Logique, Démonstration automatique et Programmation Logique

S. Piechowiak

Université Polytechnique HAUTS-DE-FRANCE

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE

Plan du cours

- Cours
  - Logique propositionnelle
  - Éléments du calcul des prédicats / démonstration automatique
  - Programmation logique avec PROLOG

Bibliographie

- **J-P. Delahaye : «** Outils logiques pour l'Intelligence Artificielle », Eyrolles, 1988
- **A. Thayse : «** Approche logique de L'intelligence Artificielle » (4 tomes), Dunod, 1993
- **J-P. Haton et al. : «** Le Raisonnement en Intelligence Artificielle », Interéditions, 1991
- **D. Poole, A. Mackworth, R. Goebel : «** Computational Intelligence : A Logical Approach », Oxford University Press, 1998
- **L. Sterling & E. Shapiro : «** The Art of Prolog », Eds Masson 1990

- **SWI-Prolog** : disponible sur différentes plate formes : LINUX, UNIX et WINDOWS

Introduction

- **Le calcul propositionnel ou logique des propositions** a pour objet l'étude des formes de raisonnement dont la validité est indépendante de la structure des propositions composantes et résulte uniquement de leurs propriétés d'être vraies ou fausses.

- **2 aspects :**
  - **syntaxique** : on s'intéresse essentiellement à la structure des formules et aux outils «mécaniques ». A partir de règles appliquées à des formules on déduit de nouvelles formules.
  - **sémantique** : on s'intéresse au « sens *logique*» des formules. On cherche à interpréter les formules.

Système formel

- D'un point de vue syntaxique, le calcul propositionnel est un système formel.
- Un système formel est défini par:
  - les symboles qui sont utilisés
  - la syntaxe des formules
  - la donnée d'un ensemble de formules appelées axiomes
  - les méthodes qui permettent de produire / engendrer / inférer de nouvelles formules (les règles)

Système formel : définition

On appelle système formel $S = (\Sigma_S, F_S, A_S, R_S)$ la donnée de :

- un **alphabet** dénombrable Σ_S
- un sous-ensemble *récursif* $F_S \subseteq \Sigma_S{}^*$ de suites finies d'éléments de Σ_S (appelé ensemble des **formules bien formées de S**)
- un sous-ensemble récursif $A_S \subseteq F_S$ appelé ensemble des **axiomes** de S
- un ensemble fini $R_S = \{r1,\ldots,rn\}$ de **règles d'inférences**.

Système formel : notations/remarques

- La notation $f_1, f_2, \ldots, f_n \underset{ri}{\vdash} g$ signifie : on peut engendrer la formule (bien formée) *g* par application de la règle *ri* sur les formules (bien formées) *f1,…,fn* ».
- Les règles d'inférence sont également appelées règles de dérivation ou de déduction.

Système formel : exemple

Définition du système formel S1 (exemple)

- alphabet : $\Sigma_{S1} = \{1,+,=\}$
- formules bien formées : $F_{S1} = \{1^n + 1^m = 1^p$ avec $\{n,m,p\} \subseteq N^* \}$

où 1^k représente le mot 111….1 qui contient k 1

- axiomes : $A_{S1} = \{1 + 1 = 11 \}$
- règles d'inférence $R_{S1} = \{r1, r2\}$ avec :

$$1^n + 1^m = 1^p \underset{r1}{\vdash} 1^{n+1} + 1^m = 1^{p+1}$$

$$1^n + 1^m = 1^p \underset{r2}{\vdash} 1^n + 1^{m+1} = 1^{p+1}$$

Système formel : exemple

Définition du système formel S1 (exemple,suite)

Succession d'application de r1 et r2 :

$$\vdash \; 1+1=11 \vdash_{r2} 1+11=111 \vdash_{r2} 1+111=1111 \vdash_{r1} 11+111=11111$$

qu'on notera :

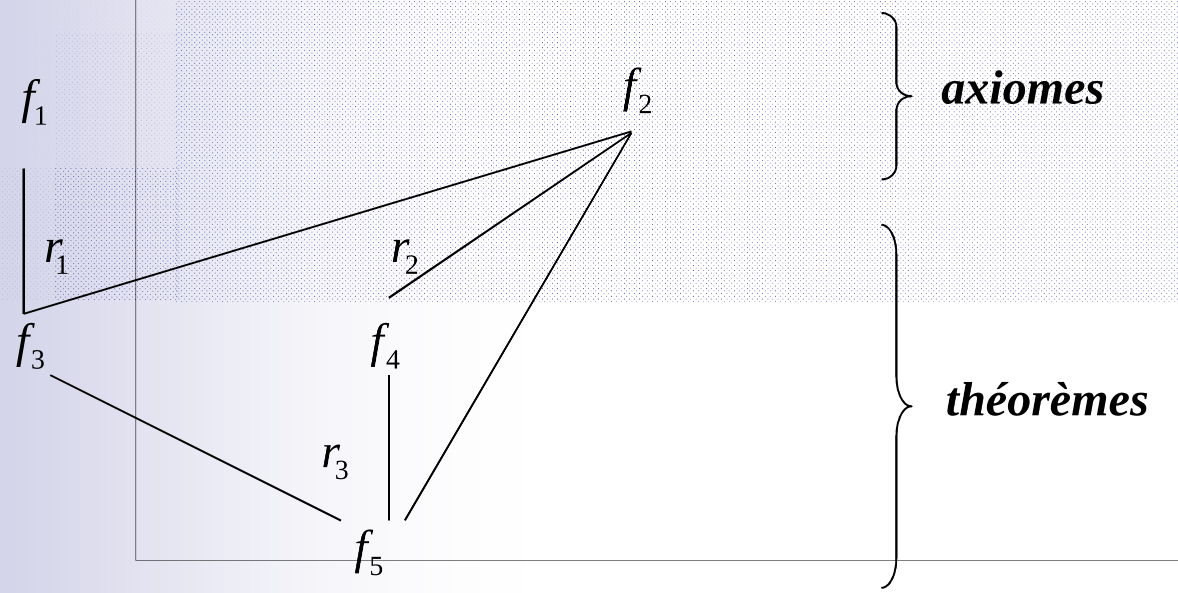
$$\begin{array}{lll} f_1 & \vdash 1+1=11 & \textit{axiome} \\ f_2 & \vdash 1+11=111 & r_2(f_1) \\ f_3 & \vdash 1+111=1111 & r_2(f_2) \\ f_4 & \vdash 11+111=11111 & r_1(f_3) \end{array}$$

La suite f_1, f_2, f_3, f_4 constitue une déduction de **11+111=11111** à partir de l'ensemble vide d'hypothèses. Donc : $11+111=11111$ est un théorème de $S1$

$$11+111=11111 \in T_{S1}$$

Système formel

Représentation graphique d'une déduction :



Système formel : preuve et théorème

Soit un système $S = (\Sigma_S, F_S, A_S, R_S)$

- on appelle ***déduction*** (ou *démonstration* ou *preuve*) à partir des formules $h_1,\ldots,h_n$ (***hypothèses***) toute suite de formules $f_1,\ldots,f_m$ telle que chaque f_i, $i \in \{1..m\}$ est :
  - ou bien un axiome de A_S
  - ou bien l'une des formules $h_1,\ldots,h_n$
  - ou bien fi est obtenue par l'application d'une règle de R_S sur des formules de $A_S \cup \{h_1,\ldots,h_n\} \cup \{f1,\ldots,fi-1\}$
- si $\{h_1,\ldots,h_n\} = \{\}$, les formules déduites sont appelées ***théorèmes***.

Notation : Pour exprimer que φ est un théorème de S, on notera : $\underset{S}{\vdash} \varphi$

L'ensemble des théorèmes de S est noté T_S.

Système formel : vocabulaire

- Le système S est dit ***cohérent*** s'il y a des formules de F_S qui ne sont pas des théorèmes.
- Si S possède le signe de négation $\neg$, on dira que S est ***consistant*** (ou *non contradictoire*) si pour aucune des formules de F_S on a φ et $\neg\varphi$ qui sont simultanément des théorèmes de S.
- Les axiomes sont dits ***indépendants*** si lorsqu'on retire l'un d'entre eux de A_S, on obtient moins de théorèmes.
- Si on modélise un problème par un système formel S et qu'on souhaite que T soit l'ensemble des théorèmes de S, on dit que S est ***correct*** si $T_S \subseteq T$ et on dit qu'il est *complet* si $T \subseteq T_S$.

Système formel : décidabilité

Le **problème de la décidabilité** consiste à savoir si pour toute formule φ de F_S on peut décider si oui ou non $\varphi \in T_S$.

Système formel : exercice

Soit le système formel S2:

- $\Sigma_{S2} = \{1,2,3,\ldots,n,\ldots,+,=\}$
- $F_{S2} = \{n_0+\ldots n_p = m_0\ldots m_q,\ p\geq 0,\ q\geq 0,\ n_i\in\mathbb{N}^*,\ n_j\in\mathbb{N}^*\}$
- $A_{S2} = \{1+\ldots+1 \text{ (p fois)} = p \ /\ p\in\mathbb{N}^*\}$
- $R_{S2} = \{r\}$ avec :

$$\left.\begin{cases} n_{1_0}+\ldots+n_{1_{p1}} = m_{1_0}+\ldots+m_{1_{q1}} \\ \ldots \\ n_{k_0}+\ldots+n_{k_{pk}} = m_{1_0}+\ldots+m_{k_{qk}} \end{cases}\right\} \vdash_r n_{1_0}+\ldots+n_{1_{p1}}+\ldots+n_{k_0}+\ldots+n_{k_{pk}} = m_{1_0}+\ldots+m_{1_{q1}}+\ldots+m_{1_0}+\ldots+m_{k_{qk}}$$

- Que représentent Σ_{S2} F_{S2} A_{S2} et R_{S2} ?
- Les formules suivantes sont-elles des théorèmes de S2 ?

a) **1+1+1+1+1 = 2+3** b) **2+3=5** c) **1+1+1+1=2+3** d) **2+3=3+4**

- Soit ARITH={formules arithmétiquement valides}.
  - Montrer $T_S \subseteq$ ARITH.
  - A-t-on ARITH $\subseteq T_S$
  - Ajouter une règle à Rs de manière à avoir ARITH = T_s

Système formel P0 : « calcul propositionnel »

- $\Sigma_{P0} = \{ a, b, c, e, ai, bi, ci, di, ei , p, q, \ldots\} \cup \{), (\} \cup \{ \rightarrow, \neg \}$

a, b, c, …, ei sont appelées constantes ou variables propositionnelles ou les propositions

- $F_{P0} = \{\Phi\}$. Soit Φ une formule propositionnelle, on dira que Φ est ***bien formée*** si
  - ou bien Φ est une constante propositionnelle
  - ou bien Φ est de la forme :
    - $(\alpha \rightarrow \beta)$
    - $\neg(\alpha)$

  où α et β sont des formules bien formées.

Système formel P0

- **A_{P0} = {SA1, SA2, SA3}** : SA1, SA2 et SA3 sont des schémas d'axiomes. Ce sont des « modèles » d'axiomes. Chaque schéma d'axiome représente une infinité d'axiomes.
  - *conséquence de l'hypothèse*

    SA1 : $(A \rightarrow (B \rightarrow A))$
  - *autodistributivité de l'implication*

    SA2 : $((A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)))$
  - *contraposée partielle*

    SA3 : $((\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B))$

Dans ces schémas d'axiomes, A, B et C représentent des formules bien formées quelconque de P0.

Système formel P0

Exemples d'axiomes de A_{P0}

- à partir de SA1
  - $(a \rightarrow (b\rightarrow a))$
  - $(a \rightarrow (a\rightarrow a))$
  - $((a \rightarrow (b\rightarrow a)) \rightarrow ((a \rightarrow (b\rightarrow a)) \rightarrow (a \rightarrow (b\rightarrow a))))$
- à partir de SA2
  - $((a \rightarrow (b\rightarrow c)) \rightarrow((a\rightarrow b) \rightarrow(a\rightarrow c)))$
  - $((a \rightarrow (a\rightarrow a)) \rightarrow((a\rightarrow a) \rightarrow(a\rightarrow a)))$
  - $((a \rightarrow ((a \rightarrow (b\rightarrow a)) \rightarrow c)) \rightarrow((a\rightarrow (a \rightarrow (b\rightarrow a))) \rightarrow(a\rightarrow c)))$

    // B: $(a \rightarrow (b\rightarrow a))$
- à partir de SA3
  - $((\neg a \rightarrow \neg a) \rightarrow (a\rightarrow a))$
  - $((\neg (a \rightarrow (b\rightarrow a)) \rightarrow \neg a) \rightarrow (a\rightarrow (a \rightarrow (b\rightarrow a))))$

    // B: $(a \rightarrow (b\rightarrow a))$

Système formel P0

- Une seule règle d'inférence : le ***modus ponens***

  Notation : $A, (A \to B) \vdash_{mp} B$ ou : $\dfrac{A, (A \to B)}{B}\, mp$

- Un enchaînement d'application du mp sur des formules est une *déduction*.
- Les formules obtenues par enchaînement d'application du mp *à partir des axiomes exclusivement* sont appelées théorèmes. Dans ce cas on parle de **preuve**.

Exemple

$$\underbrace{(a \to (b \to a))}_{}, \underbrace{((a \to (b \to a)) \to ((a \to (b \to b)) \to (a \to (b \to a))))}_{} \vdash_{mp} ((a \to (b \to b)) \to (a \to (b \to a)))$$

obtenu à partir de
SA1 : (**A**→(**B**→**A**))
avec **A:a** et **B:b**

obtenu à partir de
SA1 : (**A**→(**B**→**A**))
avec **A: (a→(b→ a))** et **B: (a→(b→ b))**

Système formel P0 : démonstration

Problème de décision : pour une formule bien formée donnée φ, existe-t-il une déduction qui ne parte que des axiomes et qui aboutisse à φ ?

Autrement dit, ***peut on prouver que φ est un théorème ?***

Remarque : en général, il n'est pas facile de prouver qu'une formule est un théorème en donnant une preuve car il n'y a pas de méthode systématique qui permettent de choisir les axiomes de départ et qui précise l'enchaînement des règles.

Exercice : démontrer $\vdash_{P0} (a \rightarrow a)$

Pour cela on va donner une déduction de (a→a) à partir des axiomes.

Système formel P0 : démonstration

f1 : $((a \to ((b \to a) \to a) \to ((a \to (b \to a)) \to (a \to a)))$

axiome obtenu avec SA2 en remplaçant A par a, B par (b→a) et C par a

f2 : $(a \to ((b \to a) \to a))$

axiome obtenu avec SA1 en remplaçant A par a et B par (b→a)

f3 : $((a \to (b \to a)) \to (a \to a))$

obtenu par application de mp sur f1 et f2

f4 : $(a \to (b \to a))$

obtenu avec SA1 en remplaçant A par a et B par b

f5 : $(a \to a)$

obtenu par application de mp sur f3 et f4

CQFD

Système formel P0 : propositions

Proposition : $\forall A \in F_{P0}$ on a $\vdash (A \rightarrow A)$

Proposition : soient les formules $A_1, \ldots, A_{n-1}, A_n$, B de F_{P0}.
Si on a : $A_1, \ldots, A_{n-1} \vdash (A_n \rightarrow B)$ alors $A_1, \ldots, A_{n-1}, A_n \vdash B$

Proposition (*théorème de déduction*) : soient les formules $A_1, \ldots, A_{n-1}, A_n$, B de F_{P0}.
Si on a : A1, …, An-1, An $\vdash$ B alors $A_1, \ldots, A_{n-1} \vdash (A_n \rightarrow B)$

Exemple. Montrer: $\vdash \mathbf{(A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C))}$
Le théorème de déduction nous dit qu'il suffit de montrer : $\mathbf{(A \rightarrow (B \rightarrow C)) \vdash (B \rightarrow (A \rightarrow C))}$
Le théorème de déduction nous dit qu'il suffit de montrer : $\mathbf{(A \rightarrow (B \rightarrow C)), B \vdash (A \rightarrow C)}$
Le théorème de déduction nous dit qu'il suffit de montrer : $\mathbf{(A \rightarrow (B \rightarrow C)), B, A \vdash C}$

Voici une preuve de : $\mathbf{(A \rightarrow (B \rightarrow C)), B, A \vdash C}$

f1 : $\mathbf{(A \rightarrow (B \rightarrow C))}$	*hypothèse*
f2 : **A**	*hypothèse*
f3 : $\mathbf{(B \rightarrow C)}$	*obtenu par application de mp sur f1 et f2*
f4 : **B**	*hypothèse*
f5 : **C**	*obtenu par application de mp sur f3 et f4*

CQFD

Système formel P0 : propositions

Proposition : les formules suivantes sont des théorèmes de P_0

$$\vdash_{P0} ((A \to B) \to ((B \to C) \to (A \to C)))$$

$$\vdash_{P0} (B \to ((B \to C) \to C))$$

$$\vdash_{P0} (\neg B \to (B \to C))$$

$$\vdash_{P0} (\neg\neg B \to B)$$

$$\vdash_{P0} (B \to \neg\neg B)$$

$$\vdash_{P0} ((A \to B) \to (\neg B \to \neg A))$$

$$\vdash_{P0} (B \to (\neg C \to \neg(B \to C)))$$

$$\vdash_{P0} ((B \to A) \to ((\neg B \to A) \to A))$$

Système formel P0 : extensions vers P1

- Pour simplifier l'écriture des formules, on utilise des symboles logiques supplémentaires: $\wedge, \vee, \leftrightarrow$
- On définit donc le système formel P1 = (Σ_{P1}, F_{P1}, A_{P1}, R_{P1})

$$\Sigma_{P1} = \{ p_0, p_1, p_2, \ldots, p_n, \ldots \} \cup \{ \neg, \rightarrow, \vee, \wedge, \leftrightarrow, (,) \} = \Sigma_{P0} \cup \{ \vee, \wedge, \leftrightarrow \}$$

$$F_{P_0} \subseteq F_{P_1} = F_{P_0} \cup \{ (A \wedge B), (A \vee B), (A \leftrightarrow B) \}$$

$$A_{P_0} = A_{P_1}$$

$$T_{P_0} = T_{P_1}$$

Système formel P1 : sémantique

Définition : On appelle **interprétation** (valuation assignation) toute application *i* :

$$i : \{p_0, p_1, p_2, \ldots, p_n, \ldots\} \rightarrow \{V, F\}$$

i[x]	i[y]	i [¬ x]	i [x ∨ y]	i [x ∧ y]	i[x → y]	i[x ↔ y]
t	t	f	t	t	t	t
t	f	f	t	f	f	f
f	t	t	t	f	t	f
f	f	t	f	f	t	t

Système formel P1 : sémantique

Définitions

- On appelle **tautologie** toute formule $A \in F_{P1}$ telle que, pour toute interprétation i : i[A]=t.

  notation : $\models$ **A**

- On dit que la formule $B \in F_{P1}$ est **conséquence** de la formule $A \in F_{P1}$ si, à chaque fois que i[A]=t, alors i[B]=t.

  notation : A $\models$ **B**

- On dit que deux formules $A \in F_{P1}$ et $B \in F_{P1}$ sont **équivalentes** si A $\models$ B et B $\models$ A

  notation : **A** $\equiv$ **B**

Système formel P1 : sémantique

Définitions sur les formules

- La formule $A \in F_{P1}$ est **satisfiable** ou **consistante** s'il existe une interprétation i telle que i[A]=t. Si i existe, on dit que c'est un **modèle** de A.
- La formule $A \in F_{P1}$ est **insatisfiable** ou **inconsistant** si pour toute interprétation i, on a : i[A]=f. (*A est insatisfiable ssi ¬A est une tautologie*)

Définitions sur les ensembles de formules

- On dit que l'ensemble de formules $\mathcal{F} \subseteq F_{P1}$ est **satisfiable** ou **consistant** s'il existe une interprétation i telle que $\forall A \in \mathcal{F}$, i[A]=t. Si i existe, on dit que c'est un **modèle** de $\mathcal{F}$.
- On dit que deux ensembles de formules sont **équivalents** s'ils ont exactement le même modèle.
- On dit que l'ensemble de formules $\mathcal{F} \subseteq F_{P1}$ est **insatisfiable** ou **inconsistant** si pour toute interprétation i, $\exists A \in \mathcal{F}$ telle que i[A]=f. (il n'existe aucun **modèle** de $\mathcal{F}$).

Système formel P1 : sémantique

Résultats importants

Pour toutes formules $A \in F_{P1}$ et $B \in F_{P1}$:

- $\models A \rightarrow B \Leftrightarrow A \models B$
- $\models A \leftrightarrow B \Leftrightarrow A \equiv B$
- si $\models A$ et si $\models A \rightarrow B$ alors $\models B$
- $\models A \wedge B \Leftrightarrow \models A$ et $\models B$
- si $\models A$ ou $\models B$ alors $\models A \vee B$

$$\vdash A \Leftrightarrow \models A$$

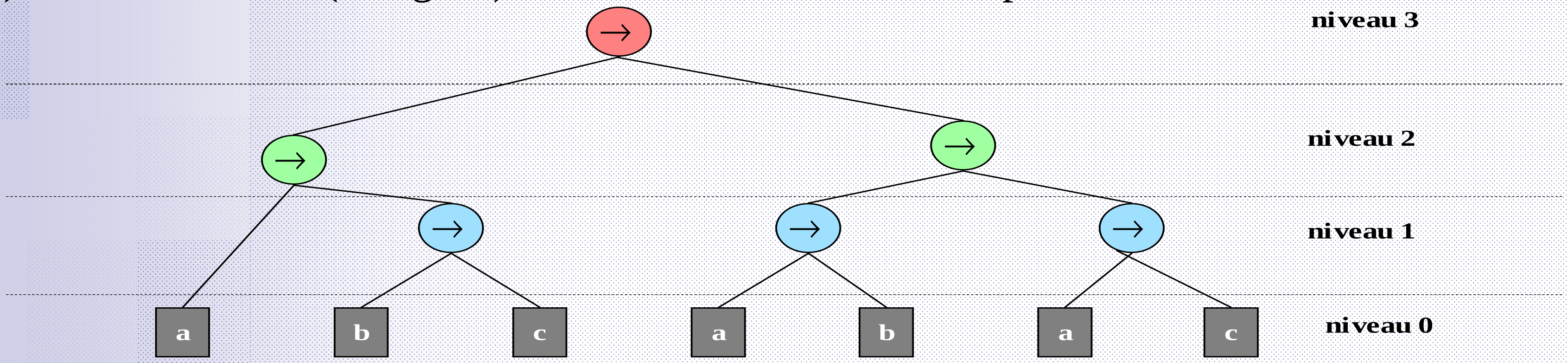
Système formel P1 : évaluation d'une expression

Méthode des tables de vérité

Soit à évaluer l'expression : $\varphi = ((a \to (b \to c)) \to ((a \to b) \to (a \to c)))$

a) Mise sous forme arborescente et calcul des niveaux

b) Table de vérité (2^n lignes) : les valeurs sont calculées par niveau.

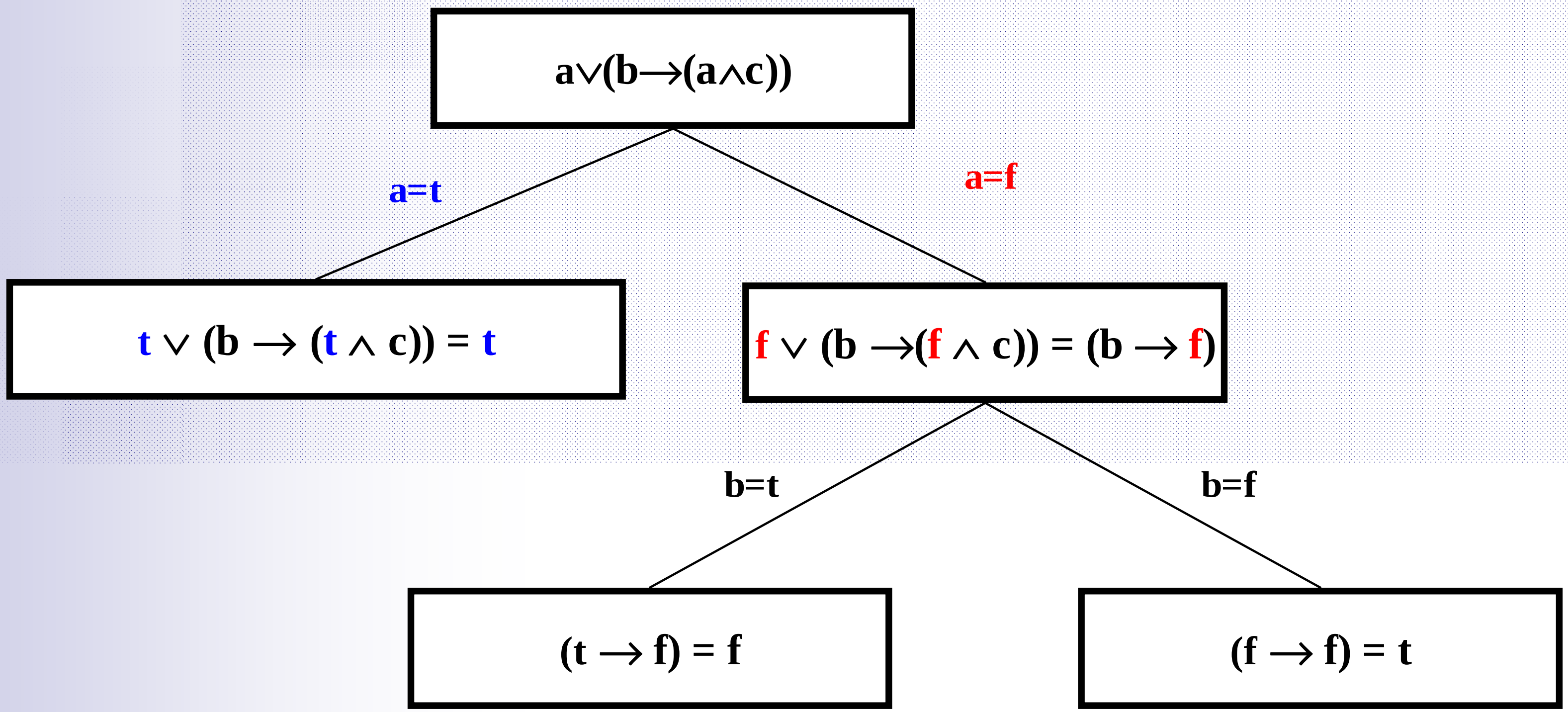


0	2	0	1	0	3	0	1	0	2	0	1	0
((a	→	(b	→	c))	→	((a	→	b)	→	(a	→	c)))
t	t	t	t	t	t	t	t	t	t	t	t	t
t	f	t	f	f	t	t	t	t	f	t	f	f
t	t	f	t	t	t	t	f	f	t	t	t	t
t	t	f	t	f	t	t	f	f	t	t	f	f
f	t	t	t	t	t	f	t	t	t	f	t	t
f	t	t	f	f	t	f	t	t	t	f	t	f
f	t	f	t	t	t	f	t	f	t	f	t	t
f	t	f	t	f	t	f	t	f	t	f	t	f

Système formel P1 : évaluation d'une expression

Algorithme de Quine

Soit à évaluer l'expression : $\varphi = a \vee (b \rightarrow (a \wedge c))$

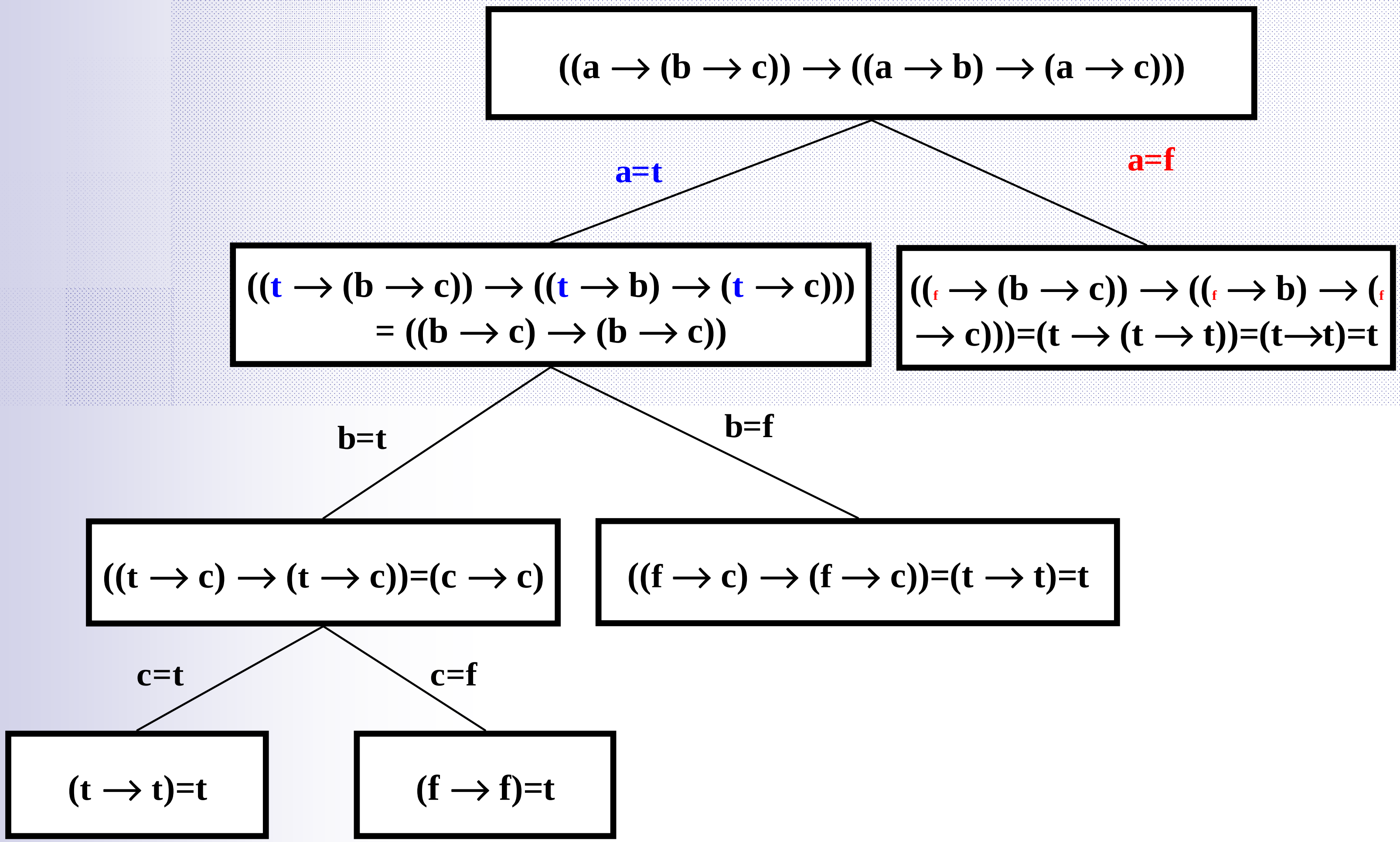


L'algorithme de Quine repose sur le même principe que les tables de vérité mais avec une évaluation plus rapide de l'expression.

Système formel P1 : évaluation d’une expression

Algorithme de Quine

Soit à évaluer l’expression : $\varphi = ((a \rightarrow (b \rightarrow c)) \rightarrow ((a \rightarrow b) \rightarrow (a \rightarrow c)))$



Système formel P1 : transformation d'une expression

On va transformer une expression en une autre expression logiquement équivalente.

formule	formule logiquement équivalente	commentaires
$\neg(\alpha \wedge \beta)$	$\neg\alpha \vee \neg\beta$	Loi de de Morgan
$\neg(\alpha \vee \beta)$	$\neg\alpha \wedge \neg\beta$	Loi de de Morgan
$(\alpha \rightarrow \beta)$	$\neg\alpha \vee \beta$	
$(\alpha \wedge \beta)$	$\neg(\neg\alpha \vee \neg\beta)$	
$(\alpha \leftrightarrow \beta)$	$(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$ $(\alpha \wedge \beta) \vee (\neg\alpha \wedge \neg\beta)$	
$(\alpha \wedge (\beta \vee \gamma))$	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$	distributivité du $\wedge$ sur le $\vee$
$(\alpha \vee (\beta \wedge \gamma))$	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$	distributivité du $\vee$ sur le $\wedge$
$\neg\neg\alpha$	α	
$(\alpha \wedge (\beta \wedge \gamma))$	$((\alpha \wedge \beta) \wedge \gamma))$	associativité du $\wedge$
$(\alpha \vee (\beta \vee \gamma))$	$((\alpha \vee \beta) \vee \gamma))$	associativité du $\vee$
$(\alpha \wedge \beta)$	$(\beta \wedge \alpha)$	commutativité du $\wedge$
$(\alpha \vee \beta)$	$(\beta \vee \alpha)$	commutativité du $\vee$
$(t \vee \alpha)$	t	
$(f \vee \alpha)$	α	
$(f \wedge \alpha)$	f	
$(t \wedge \alpha)$	α	
$(\alpha \wedge \neg\alpha)$	f	contradiction
$(\alpha \vee \neg\alpha)$	t	tautologie
$(\alpha \vee (\neg\alpha \wedge \beta))$	$(\alpha \vee \beta)$	
$(\alpha \vee (\alpha \wedge \beta))$	α	absorption
$(\alpha \wedge (\alpha \vee \beta))$	α	absorption

Système formel P1 : transformation d'une expression

Toute formule propositionnelle peut être transformer en une autre formule qui lui est logiquement équivalente et de la forme:

1. en n'utilisant que les connecteurs $\vee$ et $\neg$
2. en n'utilisant que les connecteurs $\wedge$ et $\neg$
3. en n'utilisant que les connecteurs $\rightarrow$ et $\neg$
4. sous la forme d'une conjonction de disjonctions de littéraux.
5. sous la forme d'une disjonction de conjonctions de littéraux.

Un littéral est une expression de la forme p ou $\neg$p où p est une proposition.

Système formel P1 : transformation d'une expression

Démonstration: *1. en n'utilisant que les connecteurs* $\vee$ *et* $\neg$

Il suffit d'utiliser les équivalences logiques :

$(\alpha \rightarrow \beta) = \neg\alpha \vee \beta$

$(\alpha \wedge \beta) = \neg\, \neg\, (\alpha \wedge \beta) = \neg\, (\neg\alpha \vee \neg\, \beta)$

exemple : transformons la formule $(\neg a \rightarrow (b \rightarrow\ (c \vee \neg d\ )))$

$= (\neg\, \neg\, a \vee (b \rightarrow\ (c \vee \neg d\ )))$

$= (\neg\, \neg\, a \vee (\neg\, b \vee (c \vee \neg d\ )))$

$= (a \vee \neg\, b \vee c \vee \neg d\ )$

Système formel P1 : transformation d'une expression

Démonstration: *2. en n'utilisant que les connecteurs ∧ et ¬*

Il suffit d'utiliser les équivalences logiques :

$(\alpha \rightarrow \beta) = \neg\alpha \vee \beta = = \neg (\neg\neg\alpha \wedge \neg \beta) = \neg (\alpha \wedge \neg \beta)$

$(\alpha \vee \beta) = \neg \neg (\alpha \vee \beta) = \neg (\neg\alpha \wedge \neg \beta)$

exemple : transformons la formule $(\neg a \rightarrow (b \rightarrow (c \vee \neg d)))$

$= (\neg a \rightarrow (b \rightarrow \neg(\neg c \wedge \neg \neg d))) = (\neg a \rightarrow (b \rightarrow \neg(\neg c \wedge d)))$

$= (\neg a \rightarrow \neg(b \wedge \neg \neg(\neg c \wedge d))) = (\neg a \rightarrow \neg(b \wedge (\neg c \wedge d)))$

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

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

Système formel P1 : transformation d'une expression

Démonstration: *3. en n'utilisant que les connecteurs $\rightarrow$ et $\neg$*

Il suffit d'utiliser les équivalences logiques :

$(\alpha \wedge \beta) = \neg(\neg\alpha \vee \neg\beta) = \neg(\alpha \rightarrow \neg\beta)$

$(\alpha \vee \beta) = (\neg\alpha \rightarrow \beta)$

exemple : transformons la formule $(\neg a \rightarrow (b \rightarrow (c \vee \neg d)))$

$= \ldots$

Système formel P1 : transformation d'une expression

Démonstration: ***4. sous la forme d'une conjonction de disjonctions de littéraux (un littéral est une expression de la forme p ou ¬p où p est une proposition).*** **Cette forme est appelée Forme Normale Conjonctive FNC**

$$\bigwedge_i \left(\bigvee_j \left(l_j \right) \right) \qquad avec\ l_j = p_j\ ou\ l_j = \neg p_j$$

clauses

Exemple.

La formule $\varphi = (a \vee b) \wedge (a \vee \neg b) \wedge (\neg a) \wedge (c \vee d \vee b \vee \neg c)$

possède les 4 clauses : $(a \vee b)$, $(a \vee \neg b)$, $(\neg a)$ et $(c \vee d \vee b \vee \neg c)$

Système formel P1 : transformation d'une expression

Démonstration: ***5. sous la forme d'une disjonction de conjonctions de littéraux (un littéral est une expression de la forme p ou ¬p où p est une proposition).*** **Cette forme est appelée Forme Normale Disjonctive FND**

$$\bigvee_i \left(\bigwedge_j (l_j) \right) \quad avec\ l_j = p_j\ ou\ l_j = \neg p_j$$

Exemple.

La formule $\varphi = (a \vee b) \wedge (a \vee \neg b) \wedge (\neg a) \wedge (c \vee d \vee b \vee \neg c)$ possède les 4 clauses : $(a \vee b)$, $(a \vee \neg b)$, $(\neg a)$ et $(c \vee d \vee b \vee \neg c)$

Système formel P1 : satisfiabilité

Rappel : La formule $A \in F_{P1}$ est **satisfiable** s'il existe une interprétation i telle que i[A]=t.

Autrement dit, on cherche à savoir si la formule A peut être vraie.

L'algorithme de Davis & Putnam répond à cette question.

Cet algorithme suppose que la formule A soit mise sous ***forme clausale*** (ensemble de clauses).

Système formel P1 : satisfiabilité

exemple1 : la formule Ψ est-elle satisfiable ?

$\Psi = (h \vee \neg h \vee g \vee a) \wedge g \wedge (\neg g \vee d \vee \neg e) \wedge (\neg g \vee d \vee e) \wedge (\neg g \vee a \vee \neg b) \wedge (a \rightarrow b) \wedge (\neg a \vee b \vee c) \wedge (a \vee \neg b \vee \neg c)$

sous forme normale conjonctive on a :

$\Psi = (h \vee \neg h \vee g \vee a) \wedge g \wedge (\neg g \vee d \vee \neg e) \wedge (\neg g \vee d \vee e) \wedge (\neg g \vee a \vee \neg b) \wedge (\neg a \vee b) \wedge (\neg a \vee b \vee c) \wedge (a \vee \neg b \vee \neg c)$

sous forme clausale on a :

$\{ (h \vee \neg h \vee g \vee a), g, (\neg g \vee d \vee \neg e), (\neg g \vee d \vee e), (\neg g \vee a \vee \neg b), (\neg a \vee b), (\neg a \vee b \vee c), (a \vee \neg b \vee \neg c) \}$

Système formel P1 : Algo Davis & Putnam

fonction DP(entrée: CL, un ensemble de clauses) : booléen;
début
si CL = { }
alors renvoyer(vrai)
sinon si ⊥ (la clause vide) ∈ CL
alors renvoyer(faux)
sinon appliquer tant que c'est possible les règles suivantes :

R1: **supprimer toutes les clauses qui sont des tautologies**

R2: **si une clause est réduite à un seul littéral L, supprimer toutes les clause contenant L et supprimer toutes les occurrences de Lc des autres clauses.**

R3: **si un littéral L apparaît dans les clauses CL1, CL2, …, CLN et que Lc n'apparaît dans aucune clause alors supprimer les clauses CLi.**

R4: **si une clause C1 est "incluse" dans une clause C2 ,(c'est à dire que tous les littéraux de C1 apparaissent dans C2) alors supprimer C2.**

R5: **si toutes les clauses de CL contiennent soit L soit Lc, alors scinder CL en CX et CY :**

- **CX, est constitué en supprimant de CL toutes les clauses qui contiennent L et en supprimant toutes les occurrences de Lc**
- **CY, est constitué en supprimant de CL toutes les clauses qui contiennent Lc et en supprimant toutes les occurrences de L**

renvoyer(DP(CX) ∧ DP(CY))

fin

Système formel P1 : satisfiabilité

exemple1 : la formule Ψ est-elle satisfiable ?

$\Psi = (h \lor \neg h \lor g \lor a) \land g \land (\neg g \lor d \lor \neg e) \land (\neg g \lor d \lor e) \land (\neg g \lor a \lor \neg b) \land (a \rightarrow b) \land (\neg a \lor b \lor c) \land (a \lor \neg b \lor \neg c)$

sous forme clausale on a :

$\{ (h \lor \neg h \lor g \lor a), g, (\neg g \lor d \lor \neg e), (\neg g \lor d \lor e), (\neg g \lor a \lor \neg b), (\neg a \lor b), (\neg a \lor b \lor c), (a \lor \neg b \lor \neg c) \}$

application de la règle 1

$\{ g, (\neg g \lor d \lor \neg e), (\neg g \lor d \lor e), (\neg g \lor a \lor \neg b), (\neg a \lor b), (\neg a \lor b \lor c), (a \lor \neg b \lor \neg c) \}$

application de la règle 2

$\{ (d \lor \neg e), (d \lor e), (a \lor \neg b), (\neg a \lor b), (\neg a \lor b \lor c), (a \lor \neg b \lor \neg c) \}$

application de la règle 3

$\{ (a \lor \neg b), (\neg a \lor b), (\neg a \lor b \lor c), (a \lor \neg b \lor \neg c) \}$

application de la règle 4

$\{ (a \lor \neg b), (\neg a \lor b) \}$

application de la règle 5

$CX = \{ b \}$

$CY = \{ \neg b \}$

application de la règle 2

$\{ \}$ et $\{ \}$

on obtient donc vrai (la formule initiale est satisfiable).