Université Paris-Sud Orsay

DEA D'INFORMATIQUE 1977-78

Démonstration automatique en langage de premier ordre et Programmation en clauses de Horn

Gérard HUET

Avril 1978

Littéraux

Nous supposons définie l'algèbre libre de termes $\mathcal{T}$, construite sur l'ensemble de variables $\mathcal{V}$ et l'ensemble d'opérateurs $\mathcal{F}$ gradué par la fonction arité α.

On se donne maintenant un ensemble $\mathcal{R} = \{R_0, R_1, \dots\}$ fini ou dénombrable disjoint de $\mathcal{V}$ et $\mathcal{F}$, & gradué par une fonction arité $\beta : \mathbb{N} \to \mathbb{N}$. Les éléments de $\mathcal{R}$ sont dits <u>prédicats</u> ou symboles de relations, et seront désignés par la suite par P, Q, R, ...

On appelle <u>littéral</u> tout mot sur $\mathcal{V} \cup \mathcal{F} \cup \mathcal{R}$ de la forme

$$R_i t_1 \dots t_{\beta_i} \quad \text{où } R_i \in \mathcal{R} \ \& \ \forall j \leq \beta_i \quad t_j \in \mathcal{T},$$

et $\mathcal{L}$ est l'ensemble des littéraux.

La plupart des propriétés de $\mathcal{T}$ s'étendent à $\mathcal{L}$. On peut définir, si $L = Rt_1 \dots t_n \in \mathcal{L}$,

$$\mathcal{V}(L) = \bigcup_{i=1}^{n} \mathcal{V}(t_i)$$

$$\theta(L) = 1 + \sum_{i=1}^{n} \theta(t_i), \dots \text{etc.}$$

Pour toute substitution σ, on définit $\sigma L = R(\sigma t_1) \dots (\sigma t_n) \in \mathcal{L}$.

Soit $E = \{L_1, \dots, L_n\} \subset \mathcal{L}$. L'ensemble E est dit <u>unifiable</u> par la substitution σ ssi $\|\sigma E\| = 1$, c.a.d. ssi $\sigma L_1 = \sigma L_2 = \dots = \sigma L_n$. Une condition nécessaire est bien sûr que tous les L_i aient même prédicat. Comme pour les termes, il existe alors un unificateur principal de E, que l'on note UP(E). Finalement, on définit

$$\mathcal{V}(E) = \bigcup_{i=1}^{n} \mathcal{V}(L_i).$$

Remarque σE est un singleton si $\sigma = UP(E)$.

On définit $\sigma(E) = \{\sigma L_1, \dots, \sigma L_n\}$. Attention, il s'agit vraiment d'un ensemble, c'est à dire qu'on peut avoir $1 \leq \|\sigma(E)\| \leq \|E\|$.

Clauses

Une <u>clause</u> est une paire $(\mathcal{P}, \mathcal{N})$, où $\mathcal{P}$ et $\mathcal{N}$ sont des ensembles finis de littéraux. On considère l'ensemble $\mathcal{C}$ des clauses.

Si $C = (\mathcal{P}, \mathcal{N})$, avec $\mathcal{P} = \{P_1, \dots, P_p\}$ et $\mathcal{N} = \{N_1, \dots, N_n\}$, on note

$$C = P_1, \dots, P_p \leftarrow N_1, \dots, N_n .$$

Les P_i sont appelés littéraux positifs, et on note $C_+ = \mathcal{P}$. De même les N_j sont appelés littéraux négatifs, et on note $C_- = \mathcal{N}$. L'ordre des littéraux dans chaque groupe n'a pas d'importance et on suppose qu'il n'y a pas de répétition. On définit

$$\mathcal{V}(C) = \mathcal{V}(\mathcal{P}) \cup \mathcal{V}(\mathcal{N}).$$

Lorsque $\mathcal{P} = \mathcal{N} = \emptyset$, on dit que C est la <u>clause vide</u>, que l'on note $\square$.

Sémantique

Soit un α-β-langage $(\mathcal{V}, \mathcal{F}, \mathcal{R})$. On appelle interprétation tout α-β-structure

$$I = (\mathcal{D}_I, \mathcal{F}_I, \mathcal{R}_I)$$

où $\mathcal{D}_I$ (domaine de l'interprétation I) est un ensemble non--vide, les éléments de $\mathcal{F}_I$ sont des fonctions totales dans $\mathcal{D}_I$ correspondant en nombre et en arité aux opérateurs dans $\mathcal{F}$, les éléments de $\mathcal{R}_I$ sont des relations dans $\mathcal{D}_I$ correspondant en nombre et en arité aux prédicats dans $\mathcal{R}$:

$$I(F_i) : \mathcal{D}_I^{\alpha_i} \longrightarrow \mathcal{D}_I$$
$$I(R_i) : \mathcal{D}_I^{\beta_i} \longrightarrow \mathcal{B} = \{0,1\} \qquad \forall i \in \mathbb{N}.$$

Une valuation ν de I est une application $\nu : \mathcal{V} \longrightarrow \mathcal{D}_I$, qu'on étend en un α-β-morphisme sur $\mathcal{T}$ et $\mathcal{L}$:

$$\nu(Ft_1 \ldots t_n) = I(F)(\nu(t_1), \ldots, \nu(t_n)) \quad \forall F \in \mathcal{F}$$
$$\nu(Rt_1 \ldots t_n) = I(R)(\nu(t_1), \ldots, \nu(t_n)) \quad \forall R \in \mathcal{R}.$$

C'est-à-dire que ν associe à chaque élément $t \in \mathcal{T}$ un élément $\nu(t)$ de $\mathcal{D}_I$ (l'objet désigné par t dans I,ν), et à chaque littéral L une valeur de vérité $\nu(L)$ dans $\mathcal{B}$. On dit que L est vrai dans (I,ν) si $\nu(L) = 1$, faux sinon. On désigne par VAL(I) l'ensemble des valuations de I.

Soit C une clause construite sur le langage $(\mathcal{V}, \mathcal{F}, \mathcal{R})$. On définit :

$$\nu(C) = \begin{cases} 1 & \text{si } (\forall N \in C_- \ \nu(N) = 1) \Rightarrow (\exists P \in C_+ \ \nu(P) = 1) \\ 0 & \text{sinon.} \end{cases}$$

En particulier, on a toujours $\nu(\square) = 0$.

On définit maintenant

$$I(C) = \begin{cases} 1 & \text{si } \forall \nu \in VAL(I) \ \ \nu(C) = 1 \\ 0 & \text{sinon,} \end{cases}$$

et on dit que C est vrai dans I si $I(C) = 1$, faux sinon.

Enfin, soit S un ensemble de clauses. on définit

$$I(S) = \begin{cases} 1 & \text{si } \forall C \in S \ \ I(C) = 1 \\ 0 & \text{sinon.} \end{cases}$$

Finalement, on dit que C est satisfiable, et on écrit SAT(C), ssi il existe une interprétation I telle que $I(C) = 1$. Sinon, on dit que C est insatisfiable, et on écrit INSAT(C). Mêmes définitions pour S. La propriété duale est la validité : C est valide ssi $I(C) = 1$ pour toute interprétation I; de même pour S.

• Résolution

Nous allons maintenant définir la règle d'inférence de résolution qui associe une clause $\bar{C}$ à deux clauses C et C' comme suit. Tout d'abord, si C et C' ne sont pas séparées, c'est-à-dire si $\mathcal{V}(C) \cap \mathcal{V}(C') \neq \emptyset$, alors remplacer C' par $\xi C'$ où ξ est une injection de $\mathcal{V}(C')$ dans $\mathcal{V} - \mathcal{V}(C)$. On suppose maintenant que $\mathcal{V}(C) \cap \mathcal{V}(C') = \emptyset$.

Soit $\mathcal{P} \subseteq C_+$, $\mathcal{N}' \subseteq C'_-$, $\mathcal{P}, \mathcal{N}' \neq \emptyset$, tels que $\mathcal{P} \cup \mathcal{N}'$ soit unifiable. Soit $\sigma = UP(\mathcal{P} \cup \mathcal{N}')$,

$$\bar{C}_+ = \sigma(C'_+ \cup (C_+ - \mathcal{P}))$$

$$\bar{C}_- = \sigma(C_- \cup (C'_- - \mathcal{N}')).$$

On dit que $\bar{C} = (\bar{C}_+, \bar{C}_-)$ dérive de C et C' par résolution, et on note $\bar{C} \in RES(C,C')$.

Lemme 1 (de validité) . Soient $C, C' \in \mathcal{C}$, $\bar{C} \in RES(C,C')$. Alors pour toute interprétation I telle que $I(C) = 1$ & $I(C') = 1$, on a $I(\bar{C}) = 1$.

Démonstration : soit $\bar{C} \in RES(C,C')$, avec $\mathcal{P}$, $\mathcal{N}'$ et σ comme ci-dessus. Soit I telle que $I(\bar{C}) = 0$. Donc $\exists \nu \in VAL(I)$ telle que

$$\forall L \in \sigma C_- \cup \sigma(C'_- - \mathcal{N}') \ \nu(L) = 1$$

$$\forall L \in \sigma C'_+ \cup \sigma(C_+ - \mathcal{P}) \ \nu(L) = 0.$$

Soit $M = \sigma\mathcal{P} = \sigma\mathcal{N}'$. Si $\nu(M) = 0$, alors $\nu' = \nu \circ \sigma \in VAL(I)$ est telle que $\nu'(C) = 0$, et donc $I(C) = 0$. Si $\nu(M) = 1$, alors $\nu'(C') = 0$ et donc $I(C') = 0$. $\square$

Réfutation

La règle d'inférence de résolution permet de définir une relation de déduction $\vdash$ entre clauses comme suit.

Soit S un ensemble de clauses. On dit que de S on déduit C par résolution, et on écrit $S \vdash C$, ssi $\exists C_1, \ldots, C_n \in \mathcal{C}$ tels que $C = C_n$ et

$\forall i \leq n$ • soit $C_i \in S$

• soit $\exists j,k < i \ \ C_i \in RES(C_j, C_k)$.

On dit que l'ensemble S est réfutable ssi $S \vdash \square$ (on peut déduire la clause vide de S).

Théorème. $\forall S \quad INSAT(S) \Leftrightarrow S \vdash \square$ (S est satisfiable ssi S n'est pas réfutable).

Démonstration : voir plus loin.

De ce résultat, on tire une méthode de preuve complète pour la logique de 1er ordre. Soit une théorie définie par les axiomes $A_1, \ldots, A_n$. Pour montrer que la proposition T est un théorème de la théorie, il faut montrer que la proposition

$$p = (A_1 \wedge \ldots \wedge A_n) \supset T$$

est valide, ce qui est équivalent à montrer que $\neg p$ est insatisfiable, ce qui est équivalent à montrer $\mathrm{INSAT}(\mathcal{C}(\neg p))$, ce qui est équivalent à montrer $\mathcal{C}(\neg p) \vdash \square$, i.e.

$$\mathcal{C}(A_1), \ldots, \mathcal{C}(A_n), \mathcal{C}(\neg T) \vdash \square \quad .$$

EXEMPLES

1°) Certains étudiants sérieux assistent à tous les cours d'informatique. Aucun étudiant sérieux n'assiste à un cours stupide. Donc aucun cours d'informatique n'est stupide.

On axiomatise les concepts en jeu par :

$E(x)$	x est un étudiant sérieux
$S(x)$	x est un cours stupide
$I(x)$	x est un cours d'informatique
$A(x,y)$	l'étudiant x assiste au cours y.

A1 : $\exists x\ (E(x) \wedge \forall y (I(y) \supset A(x,y)))$

A2 : $\forall x\ (E(x) \supset \forall y (S(y) \supset \neg A(x,y)))$

T : $\forall x\ (I(x) \supset \neg S(x))$

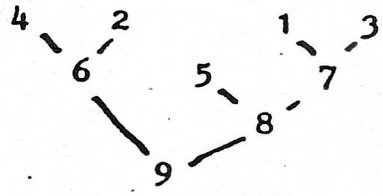
donnent l'ensemble de clauses

C1	$E(X) \leftarrow$	(A1)
C2	$A(X,y) \leftarrow I(y)$	(A1)
C3	$\leftarrow E(x), S(y), A(x,y)$	(A2)
C4	$I(Y) \leftarrow$	(T)
C5	$S(Y) \leftarrow$	(T)

d'où on obtient la réfutation suivante :

C6	$A(X,Y) \leftarrow$	RES(4,2)
C7	$\leftarrow S(y), A(X,y)$	RES(1,3)
C8	$\leftarrow A(X,Y)$	RES(5,7)
C9	$\square$	RES(6,8)

correspondant à l'arbre de preuve :



NB : on peut typer ou non typer indifféremment

2°) Dans un groupe, l'élément neutre à droite est aussi neutre à gauche. Axiomatisation :

$$\begin{cases} P(x,y,z) & : x * y = z \\ I(x) & : x^{-1} \\ E & : \text{élément neutre} \end{cases}$$

1	$P(x,I(x),E) \leftarrow$	I inverse à droite
2	$P(x,E,x) \leftarrow$	E neutre à droite
3	$P(x,y,z) \leftarrow P(u,v,x),P(v,y,w),P(u,w,z)$	associativité
4	$P(x,y,z) \leftarrow P(x,u,w),P(u,v,y),P(w,v,z)$	associativité
5	$\leftarrow P(E,A,A)$	T
6	$\leftarrow P(E,u,w),P(u,v,A),P(w,v,A)$	RES(4,5)
7	$\leftarrow P(E,v,A)$	RES(2,6)
8	$\leftarrow P(u,v,E),P(v,y,w),P(u,w,A)$	RES(3,7)
9	$\leftarrow P(A,v,E),P(v,y,E)$	RES(2,8)
10	$\leftarrow P(A,v,E)$	RES(1,9)
11	$\square$	RES(1,10)

Dans cet exemple, l'arbre de preuve est linéaire. Celui de l'exemple 1 ne l'est pas, mais il est facile de modifier la preuve de manière à le rendre linéaire.

Malheureusement, ceci n'est pas toujours possible comme le montre le contre-exemple :

3°)

1	$A,B \leftarrow$
2	$A \leftarrow B$
3	$B \leftarrow A$
4	$\leftarrow A,B$

<u>Domaine de Herbrand, arbres sémantiques.</u>

Soit S un ensemble de clauses construit sur $\mathcal{V},\mathcal{F},\mathcal{R}$. On suppose que $\mathcal{F}_0 \neq \emptyset$ (si $\mathcal{F}_0 = \emptyset$, on ajoute à $\mathcal{F}$ une constante A).

On appelle <u>domaine de Herbrand</u> de S l'ensemble des termes fermés construits sur $\mathcal{F}$:

$$\mathcal{H} = \{t \in \mathcal{T} ; \mathcal{V}(t) = \emptyset\}.$$

Autrement dit, $\mathcal{H}$ est le plus petit ensemble de termes fermé par :

$$t_1,\ldots,t_n \in \mathcal{H} \ \& \ F \in \mathcal{F}_n \Rightarrow Ft_1\ldots t_n \in \mathcal{H}.$$

($\mathcal{H} \neq \emptyset$ car $\mathcal{H} \supseteq \mathcal{F}_0 \neq \emptyset$ par hypothèse).

On appelle <u>base de Herbrand</u> de S l'ensemble de tous les littéraux fermés construits sur $\mathcal{R}$ et $\mathcal{F}$:

$$\overline{\mathcal{H}} = \{Rt_1\ldots t_n ; R \in \mathcal{R}_n, t_1,\ldots,t_n \in \mathcal{H}\}.$$

Une <u>interprétation de Herbrand</u> de S consiste à interpréter S en prenant comme domaine d'interprétation $\mathcal{H}$, et en interprétant $F \in \mathcal{F}_n$ comme la fonction

$$t_1,\ldots,t_n \longmapsto Ft_1\ldots t_n \ .$$

Une telle interprétation est spécifiée de manière unique par un sous-ensemble I de $\tilde{\mathcal{H}}$: on interprète $Rt_1\ldots t_n \in \tilde{\mathcal{H}}$ par la fonction caractéristique de I :

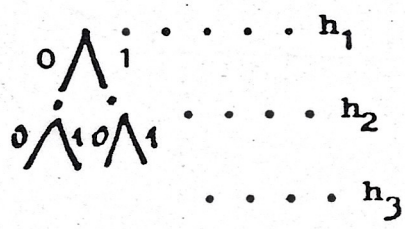
$$\begin{cases} 1 & \text{si } Rt_1\ldots t_n \in I \\ 0 & \text{sinon.} \end{cases}$$

Dans une interprétation de Herbrand, les valuations sont des substitutions. On peut donc interpréter dans I une clause C comme suit.

Soit $\Sigma = \{\sigma : \mathcal{V} \to \mathcal{H}\}$ l'ensemble des substitutions restreintes au domaine de Herbrand. On a :

$$I(C) = \begin{cases} 1 \text{ si } & \forall \sigma \in \Sigma \quad \sigma(C_-) \subseteq I \Rightarrow \sigma(C_+) \cap I \neq \emptyset \\ 0 \text{ sinon.} & \end{cases}$$

$\mathcal{F}$ et $\mathcal{K}$ étant des ensembles finis ou dénombrables, $\mathcal{H}$ et $\tilde{\mathcal{H}}$ sont des ensembles dénombrables. Considérons une énumération arbitraire de $\tilde{\mathcal{H}}$: $\tilde{\mathcal{H}} = \{h_1, h_2, \ldots, h_i, \ldots\}$ .Une interprétation de Herbrand peut être représentée par sa fonction caractéristique dans $\mathcal{B}^{\tilde{\mathcal{H}}}$, ou par sa branche de l'<u>arbre sémantique</u> :



qui décrit toutes les interprétations de Herbrand.

L'intérêt des interprétations de Herbrand provient du lemme suivant.

<u>Lemme 2</u> . Soit S un ensemble quelconque de clauses. S est satisfiable ssi S est satisfiable dans une interprétation de Herbrand de S.

Démonstration :

$\Leftarrow$ est trivial

$\Rightarrow$ Soit I une interprétation qui satisfait S. On définit une interprétation de Herbrand I' de S par

$\forall t_1, \ldots, t_n \in \mathcal{H}$, $\forall R \in \mathcal{R}_n$:

$$Rt_1 \ldots t_n \in I' \Leftrightarrow I(R)(I(t_1), \ldots, I(t_n)) = 1$$

(dans le cas où initialement $\mathcal{F}_0 = \emptyset$, et où on rajoute A à $\mathcal{F}$, alors on peut choisir I(A) comme un élément arbitraire du domaine d'interprétation $\mathcal{D}$ de I).

Par construction, I' satisfait S : à chaque σ de Σ, faire correspondre la valuation ν de I telle que

$$\forall x \in \mathcal{V} \quad \nu(x) = I(\sigma x) \ . \square$$

Remarque : ~~réduit de la fax~~ de Herbrand indifférent

EXEMPLES

Prenons $\mathcal{F} = \mathcal{F}_0 = \{A, B\}$, $\mathcal{R} = \mathcal{R}_1 = \{P, Q\}$, d'où $\mathcal{H} = \{A, B\}$ et $\overline{\mathcal{H}} = \{P(A), Q(A), P(B), Q(B)\}$.

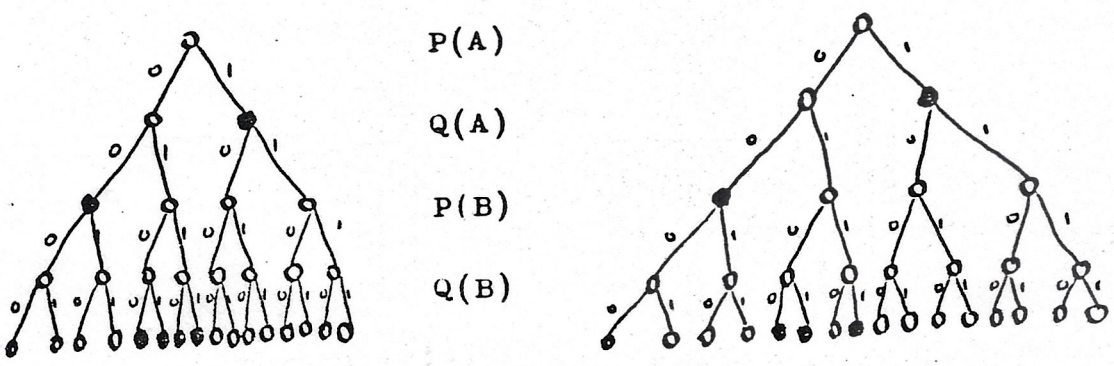
On considère les clauses :

C1	$P(x), Q(y) \leftarrow$
C'1	$P(x), Q(x) \leftarrow$
C2	$\leftarrow P(A)$
C3	$\leftarrow Q(B)$

et les ensembles de clauses

$$S = \{C1, C2, C3\}$$
$$S' = \{C'1, C2, C3\} .$$

On a pour S et S' des arbres sémantiques finis A et A'



On a représenté par des points noirs les points de l'arbre A (resp. A') correspondant à des interprétations qui rendent fausse (0) l'une des clauses de S (resp. S'). Nous allons maintenant définir précisément cette notion.

Définitions

Chaque noeud d'un arbre sémantique est désigné par un mot $u \in \mathcal{B}^*$ (occurrence de ce noeud). On appelle interprétation de Herbrand partielle associée au noeud d'occurrence u l'ensemble $IP(u) \subset \overline{\mathcal{H}} \times \mathcal{B}$ défini par

$$\begin{cases} IP(\varepsilon) = \emptyset \\ IP(u0) = IP(u) \cup \{(h_k, 0)\} \\ IP(u1) = IP(u) \cup \{(h_K, 1)\} \end{cases} \quad \text{où } k = |u| + 1$$

On dit que u <u>réfute</u> la clause C lorsque

$$\exists \sigma \in \Sigma \quad \forall N \in C_- \quad (\sigma N, 1) \in IP(u)$$
$$\& \forall P \in C_+ \quad (\sigma P, 0) \in IP(u)$$

En particulier, la racine Λ réfute la clause vide $\square$, et ne réfute pas d'autre clause.

On dit que u est <u>noeud d'échec</u> pour l'ensemble de clauses S lorsque

a) $\exists C \in S$ u réfute C

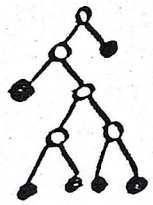
b) $\forall v \prec u$ (v ancêtre de u) $\nexists C' \in S$ v réfute C'.

Si u est noeud d'échec pour S, alors I(S) = 0 pour toute interprétation de Herbrand I prolongeant IP(u), c'est-à-dire telle que

$$\forall L \quad (L,0) \in IP(u) \Rightarrow L \notin I \ , \ (L,1) \in IP(u) \Rightarrow L \in I.$$

On appelle <u>arbre sémantique fermé</u> pour S un arbre sémantique initial dont les feuilles (noeuds terminaux) sont exactement les noeuds d'échec pour S.

EXEMPLE. Pour l'ensemble S ci-dessus, on obtient l'arbre sémantique fermé :



Par contre, S' ne possède pas d'arbre sémantique fermé, à cause de la feuille 0110 qui correspond à l'interprétation de Herbrand $\{Q(A),P(B)\}$ qui satisfait S'.

<u>Théorème 1</u> (Herbrand 1930)

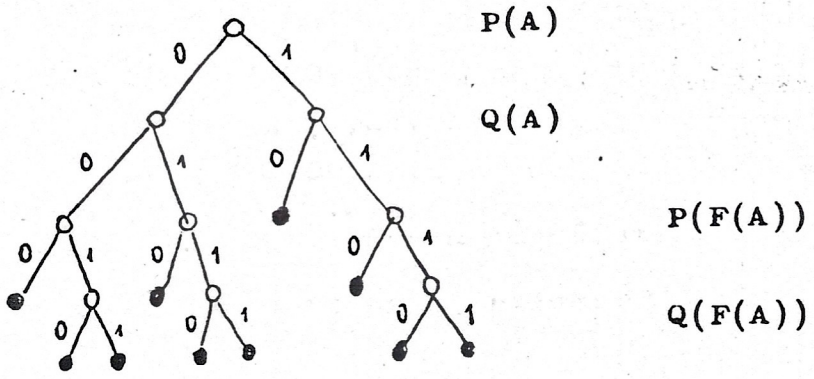
Soit S un ensemble quelconque de clauses. S est insatisfiable ssi S possède un arbre sémantique fermé pour S fini.

<u>Démonstration</u> : soit un arbre sémantique fermé pour S. Par le lemme de König, cet arbre est fini ssi il ne possède pas de branche infinie. S'il est fini, alors S est faux dans toute interprétation de Herbrand de S, et donc insatisfiable par le lemme 2. S'il est infini, une branche infinie détermine une interprétation de Herbrand qui satisfait S, et réciproquement. $\square$

EXEMPLE Soit

$$S = \{Q(x) \leftarrow P(x),\ P(F(y)) \leftarrow ,\ \leftarrow Q(F(y))\}$$
$$\mathcal{H} = \{A, F(A), F(F(A)), \ldots\}$$
$$\overline{\mathcal{H}} = \{P(A), Q(A), P(F(A)), Q(F(A)), P(F(F(A))), \ldots\}\ .$$

S est insatisfiable, et un arbre sémantique fermé pour S est



Le théorème de Herbrand nous donne la possiblité d'écrire une procédure de semi-décision pour la validité d'une proposition p, en énumérant un arbre sémantique fermé pour $\mathcal{C}(\neg p)$. Il y a trois cas :

- on trouve en un temps fini un arbre sémantique fermé (donc fini) : p est valide;
- $\overline{\mathcal{H}}$ est fini et on trouve un arbre sémantique fini dont une des feuilles n'est pas noeud d'échec : p est non-valide;
- la procédure ne s'arrête pas : p est non-valide.

Les premiers programmes de démonstration automatique étaient axés sur ce principe. Les programmes échouaient à cause de l'explosion combinatoire formidable qui en découle. En particulier, la longueur de preuve dépend de manière cruciale de lénumération de $\overline{\mathcal{H}}$ choisie.

Nous allons voir maintenant comment on peut interpréter le théorème de Herbrand comme un théorème de complétude pour la règle de résolution, ce qui nous permet de définir des procédures de démonstration automatique plus efficaces.

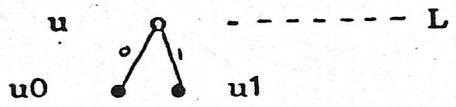
<u>Définition</u>. On appelle <u>noeud d'inférence</u> pour S un noeud dont les deux successeurs sont noeuds d'échec pour des clauses de S.

<u>Lemme 3</u> (de suffisance).

Soit S un ensemble fini de clauses insatisfiable. Alors S est réfutable.

<u>Démonstration</u> : considérons un arbre sémantique A fermé pour S fini (dont l'existence est assurée par le th. 1). A possède

au moins un noeud d'inférence u (sinon il existerait une branche infinie). Soient u0 et u1 les successeurs de u. Ce sont des noeuds d'échec pour des clauses C0 et C1 de S. Soit L le littéral dans $\mathcal{H}$ correspondant au noeud u :



On suppose que C0 et C1 ont leurs variables séparées. Considérons tout d'abord C0. Pour que u0 soit noeud d'échec pour C0 et que u ne le soit pas, il faut que C0 soit tel qu'il existe σ_0 et $\emptyset \neq \mathcal{P}^0 \subseteq C0_+$ tels que

$$\forall P \in \mathcal{P}^0 \quad \sigma_0 P = L \quad \& \quad u \text{ réfute } C = (\sigma_0(C0_+ - \mathcal{P}^0), \sigma_0 C0_-)$$

De même pour C1, il faut qu'il existe σ_1 et $\emptyset \neq \mathcal{N}^1 \subseteq C1_-$ tels que

$$\forall N \in \mathcal{N}^1 \quad \sigma_1 N = L \quad \& \quad u \text{ réfute } C' = (\sigma_1 C1_+, \sigma_1(C1_- - \mathcal{N}^1)).$$

Comme C0 et C1 sont supposées séparées, on peut considérer la substitution $\sigma_0 \cup \sigma_1$ qui unifie $\mathcal{P}^0 \cup \mathcal{N}^1$. Soit $\rho = UP(\mathcal{P}^0 \cup \mathcal{N}^1)$, $\bar{C}$ le résolvant correspondant de C0 et C1. Par définition de UP, il existe une substitution η dans Σ telle que

$$\eta \cdot \rho = \sigma_0 \cup \sigma_1$$

Par construction, η réfute $\bar{C}$ en u. Il existe donc dans A un noeud $\bar{u} \lesssim u$ qui est noeud d'échec pour $\bar{C}$. Soit $\bar{A}$ l'arbre obtenu en coupant A en $\bar{u}$. L'arbre $\bar{A}$ est un arbre sémantique fermé pour $S \cup \{\bar{C}\}$ strictement plus petit que A. On peut donc en itérant ce procédé, se ramener en un nombre fini d'étapes à l'arbre sémantique fermé trivial, ce qui correspond à la dérivation de la clause vide $\square$ par résolution à partir de S. $\square$

<u>Théorème 2</u> (d'adéquation) (Robinson, 1965)

$$\forall S \quad INSAT(S) \Leftrightarrow S \vdash \square$$

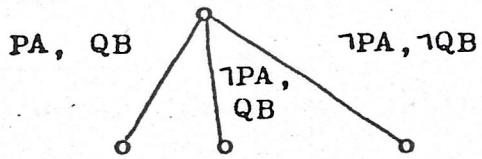
<u>Démonstration</u> : découle des lemmes 1 et 3.

REMARQUES

La notion d'arbre sémantique , et la preuve du théorème d'adéquation fondée sur cette notion sont dues à Hayes et Kowalski.

Par cette méthode, il est possible de prouver la complétude de diverses règles d'inférence. On peut pour cela généraliser la notion d'arbre sémantique. Il n'est pas nécessaire en

particulier que l'arbre représente une énumération uniforme de $\mathcal{H}$ niveau par niveau. Il est possible d'examiner plusieurs littéraux de $\mathcal{H}$ simultanément en un noeud, comme par exemple



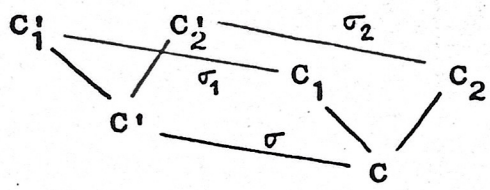
Mais il est nécessaire que deux conditions soient remplies :

1) chaque noeud a un degré de branchement fini, et la disjonction des conditions qui étiquettent ses branches est une tautologie du calcul des propositions.
2) toute branche infinie énumère complètement $\overline{\mathcal{H}}$. Autrement dit, par le lemme de König, $\forall L \in \overline{\mathcal{H}}$, $\exists k$ tel que toute branche de longueur k contienne L ou $\neg L$. (cette condition est très souvent imparfaitement définie dans la littérature).

L'analyse de la preuve du théorème d'adéquation ci-dessus montre qu'on a pu se ramener à une preuve de complétude dans le domaine de Herbrand, c'est-à-dire dans le calcul propositionnel grâce à deux résultats fondamentaux ;

- le théorème de Herbrand, qui peut s'exprimer sous la forme: un ensemble fini S de clauses est insatisfiable ssi il existe un ensemble fini S' de clauses instances fermées de celles de S qui soit insatisfiable propositionnellement.
- un lemme d'ouverture (lifting lemma) ;

si $C'_1 = \sigma_1 C_1$ et $C'_2 = \sigma_2 C_2$, pour tout $C' \in RES(C'_1, C'_2)$ il existe $C \in RES(C_1, C_2)$ et σ tels que $C' = \sigma C$.



Ceci donne une méthode générale pour prouver la complétude d'une restriction de la règle de résolution, en se ramenant au cas des clauses fermées.

Soit P une propriété des dérivations par résolution, qui soit invariante par instanciation : $P(D) \Leftrightarrow P(D')$, où D' est une instanciation de D, c'est-à-dire que toute clause à un noeud de D' est une instance de la clause au noeud correspondant de D (les arbres de D et D' sont isomorphes). Alors si

mais l'instanciation n'est pas forcément uniforme

on prouve

$(*)_P$ - $\forall S'$ ensemble fini de clauses fermées

$$\mathrm{INSAT}(S') \Leftrightarrow \exists D'\ P(D')\ \&\ S' \vdash_{D'} \square$$

alors la propriété est vraie en retirant l'hypothèse "fermées". C'est-à-dire que P est une restriction complète de la résolution.

EXERCICE : en faire la preuve complètement.

ésolution binaire

On appelle <u>facteur</u> d'une clause C toute clause C' obtenue en contractant deux ou plusieurs littéraux de même signe à l'aide de leur unificateur principal.

Plus précisément, si $\emptyset \neq \mathcal{L} \subset C_+$ est unifiable ~~d'unifié L~~, avec $\sigma = \mathrm{UP}(\mathcal{L})$, soit $C' = \cancel{(\sigma(C_+ - \mathcal{L}) \cup \{L\},\ \sigma C_-)}$. On dit que C' est σC un facteur de C. Demême en remplaçant C_+ par C_-. Il est clair que pour toute interprétation I, si $I(C) = 1$ alors $I(C') = 1$, ce qui valide la règle d'inférence de <u>factorisation</u> :

$$C \vdash_F C' \quad \text{où } C' \text{ est un facteur de } C.$$

L'intérêt de cette règle réside dans le fait que toute résolution peut se décomposer en deux factorisations et une résolution binaire :

<u>Résolution binaire</u>

$$C, C' \vdash_{RB} \sigma((C_+ - \{L_+\}) \cup C'_+,\ C_- \cup (C'_- - \{L'_-\})),$$
$$C, C' \text{ séparés},\ \sigma = \mathrm{UP}\{L_+, L'_-\}.$$

Toute résolution de C et C' est une résolution binaire d'un facteur de C et d'un facteur de C'. Ceci s'appuie sur le fait qu'une clause est son propre facteur et qu'on peut unifier $\mathcal{P} \cup \mathcal{N}'$ en unifiant $\mathcal{P}$ et $\mathcal{N}'$ séparément par $\sigma = \mathrm{UP}(\mathcal{P})$ et $\sigma' = \mathrm{UP}(\mathcal{N}')$, puis en rassemblant par $\mathrm{UP}(\mathcal{P} \cup \mathcal{N}') = \mathrm{UP}(\sigma\mathcal{P}, \sigma'\mathcal{N}')$, l'unificateur principal calculant un sup de termes.

On peut donc montrer que p est valide en obtenant une réfutation de $\mathcal{C}(\neg p)$ par résolution binaire et factorisation. La factorisation peut également s'exprimer comme composition de factorisations binaires.

Subsomption

Soient une clause C' et un ensemble de clauses S tels que il existe $C \in S$ et une substitution σ telle que $\sigma C \subseteq C'$. Alors $\{C'\} \cup S$ est insatisfiable ssi S est insatisfiable. Autrement dit, on peut supprimer C' lorsqu'elle est subsumée (filtrée) par une clause C de S.

Suppression des tautologies

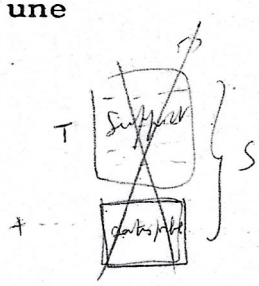
De même si C' est une tautologie, c'est-à-dire si $C'_+ \cap C'_- \neq \emptyset$.

Principe de pureté

Si C' possède un littéral L' qui ne s'unifie avec aucun littéral L de signe opposé pour aucune clause C de S, on peut également supprimer C', qui ne peut contribuer à une réfutation de S.

Stratégies de démonstration automatique

- Recherche diagonale .
- Top-down / Bottom-up .
- Ordonnancement par mérite : poids, profondeur, unit.



Quelques restrictions complètes de la règle de résolution

a) Ensemble de support

Soit S un ensemble de clauses. On dit que $T \subseteq S$ est un ensemble de support de S si S-T est satisfiable. La résolution avec ensemble de support T ne considère pas pour résolution deux clauses de S-T.

Complétude de la résolution avec ensemble de support

Si S est un ensemble fini de clauses, $T \subset S$, tels que

$$INSAT(S) \ \& \ SAT(S-T)$$

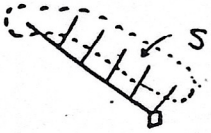
alors il existe une déduction de $\square$ à partir de S ne résolvant pas entre elles les clauses de S-T.

Démonstration : voir Chang et Lee.

Application : Resolution négative - $T = \{C \in S \mid C_+ = \emptyset\}$

b) Résolution linéaire

Bien qu'on ne puisse imposer les résolutions complètement linéaires, i.e. $C, C' \vdash \bar{C} \Rightarrow C \in S$ ou $C' \in S$



on peut se restreindre aux résolutions pseudo-linéaires :

$C, C' \vdash \bar{C} \Rightarrow C \in S$ ou $C' \in S$ ou C ancêtre de C' ou C' ancêtre de C.

La preuve de la complétude de cette restriction se trouve également dans Chang et Lee.

c) Hyper-résolution

Soit P_1 la propriété

$$C_- = \emptyset \text{ lorsque } C, C' \vdash \bar{C} .$$

Théorème 3. $\forall S$ fini $\mathrm{INSAT}(S) \Leftrightarrow S \vdash_{P1} \square$

Par la méthode ci-dessus, il suffit de prouver ce théorème pour S ensemble de clauses fermées, ce qui suit de

Lemme 4. Soit S ensemble fini de clauses fermées, avec INSAT(S) et $\square \notin S$. Alors il existe un P1-résolvant de clauses de S qui n'appartient pas à S.

Démonstration : Soit $\mathcal{L}(S)$ l'ensemble des littéraux apparaissant dans S, et

$$\mathrm{POS} = \{ C \in S \ ; \ C_- = \emptyset \}.$$

$\mathrm{POS} \neq \emptyset$, car sinon l'interprétation de Herbrand $\lambda L.0$ satisferait S. De plus SAT(POS), par exemple par $\lambda L.1$. Soit I une interprétation de Herbrand satisfiant POS et telle que

$$p = \mathrm{card}\{ L \in \mathcal{L}(S) \ ; \ I(L) = 1 \} \text{ soit minimum (on a } p>0).$$

$\mathcal{L}(S) \cap I$

$I(C) = 0$ pour certaines clauses de S - POS. Soit C une telle clause, telle de plus que $q = \mathrm{card}\, C_-$ soit minimum (on a $q>0$). Soit $L \in C_-$. Il existe une clause $C' \in \mathrm{POS}$ telle que

$$L \in C'_+ \ \& \ \forall M \in C'_+ - \{L\} \ I(M) = 0$$

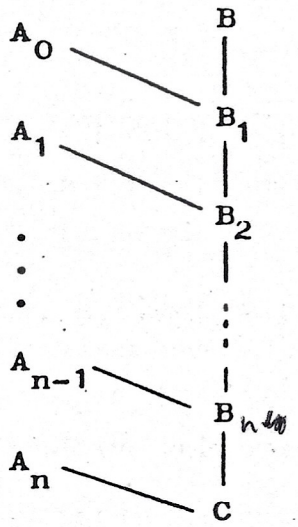
(sinon, $I' = I[L \leftarrow 0]$ remplirait les conditions pour I avec $p' < p$). Alors $\bar{C} = (C - \{L\}) \cup (C' - \{L\})$ est un P1-résolvant de C et C', et donc de clauses de S. Montrons que $\bar{C} \notin S$:

- si $\bar{C} = \square$ $\bar{C} \notin S$ par hypothèse,
- si $\bar{C} \neq \square$ & $\bar{C}_- = \emptyset$ alors $\bar{C} \notin \mathrm{POS}$ car $I(\bar{C}) = 0$ et donc $\bar{C} \notin S$.
- sinon $\bar{C}$ n'est pas dans S-POS, car $I(\bar{C}) = 0$ et $\bar{q} < q$, et donc $\bar{C} \notin S$. $\square$

REMARQUE : on peut remplacer ci-dessus P1 par N1

$$\text{N1} \qquad C'_+ = \emptyset \text{ lorsque } C, C' \vdash \bar{C} .$$

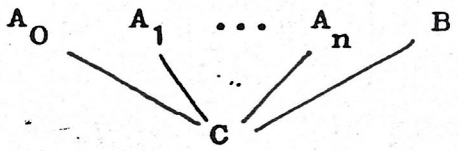
Toute P1-déduction de S peut être décomposée en segments :



où $A_{0-} = A_{1-} = \ldots = A_{n-} = C_- = \emptyset$

$B_- \neq \emptyset$ & $B \in S$

De plus l'ordre des A_i n'a pas d'importance. C'est-à-dire qu'on peut sélectionner les littéraux des B_i à résoudre dans un ordre arbitraire, ou même qu'on peut effectuer une déduction composée de pas multiples :



On dit que C est obtenu par <u>hyper-résolution</u> de $A_0, \ldots, A_n, B$.

Clauses de Horn

Une clause $P \leftarrow N$ est de Horn si $\mathrm{card}(P) \leq 1$.

Propriété sémantique fondamentale : si S est un ensemble de clauses de Horn, et si I_1 et I_2 sont des interprétations de Herbrand qui satisfont S, alors $I_1 \cap I_2$ est une interprétation qui satisfait S. Mieux, il existe une interprétation de Herbrand minimale satisfaisant S : $\bigcap \{ I;\ I(S) = 1 \}$.

Théorème 4. Soit S un ensemble de clauses de Horn insatisfiable. Alors il il existe une réfutation de S par résolution telle que

PU : si $C, C' \vdash \bar{C}$ alors $C_- = \emptyset$.

(positive unit resolution).

Démonstration : par le théorème 3 (PU est P1).

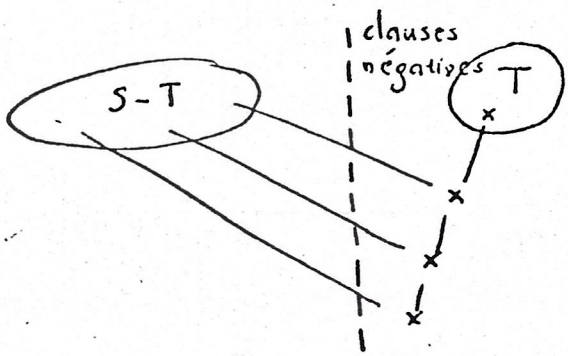
Théorème 5. Soit S un ensemble de clauses de Horn insatisfiable. Alors il existe une réfutation de S par résolution telle que :

NI : si $C, C' \vdash \bar{C}$ alors $C \in S, C_- \neq \emptyset$ et $C'_+ = \emptyset$ C_+

(negative input resolution).

Démonstration : par le dual du théorème 3, en remplaçant P1 par N1.

Toutes les clauses engendrées par la N1-résolution sont entièrement négatives et les arbres de preuve sont de la forme



clauses initiales négatives

Le théorème 4 permet de faire des réfutations "descendantes", le théorème 5 des réfutations "montantes".

On peut montrer de plus que les littéraux dans les clauses peuvent être ordonnés de manière à toujours réduire par résolution le premier littéral négatif d'une clause (Kuehner). Enfin, Henschen et Wos ont montré que l'on peut se restreindre à la résolution binaire, la factorisation n'étant pas nécessaire.

REMARQUES : pour les clauses non de Horn, input et unit sont des restrictions non-complètes (voir l'exemple ci-dessus). Mais on peut montrer qu'il existe une réfutation input ssi il existe une réfutation unit ssi les clauses peuvent être renommées en clauses de Horn.

La combinaison des deux stratégies n'est pas complète, même pour les clauses de Horn, comme le montre l'exemple

$$\left\{ \begin{array}{ll} R \leftarrow & \leftarrow P, Q, S \\ Q \leftarrow R & P \leftarrow Q, S \\ S \leftarrow R & \end{array} \right\}$$

Se limiter aux clauses de Horn revient à supprimer une certaine forme de non-déterminisme. Ceci permet d'avoir des méthodes de preuves complètes plus efficaces que dans le cas général. Ce gain peut être évalué précisément dans le cas du calcul des propositions.

Théorème 6. Soit p une proposition du calcul des propositions, S l'ensemble des clauses de $\mathcal{C}(\neg p)$, n = card(S). On peut décider en temps n^2 si p est une tautologie.

Démonstration. Ordonnons S de manière arbitraire, et divisons S en PU ($C \in PU \Leftrightarrow C_- = \emptyset$) et NU = S - PU. On définit l'algorithme :

itérer *si* PU = $\emptyset$: *écrire* ('NON TAUTOLOGIE') ; *sortir fsi* ;
soit u la première clause de PU ;
si u n'a de résolvant avec aucune clause de NU :
PU ← PU - {u}
sinon : soit v la première clauses de NU telle que
$\exists w \quad u,v \vdash_R w$;
NU ← NU - {v} ;
si w = $\square$: *écrire*('TAUTOLOGIE') ; *sortir*
sinsi $w_- = \emptyset$: PU ← PU $\cup$ {w}
sinon NU ← NU $\cup$ {w}
fsi
fsi
fitérer;

La complètude de l'algorithme découle du théorème 4, la suppression de v étant due à sa "subsomption" par w. Il est facile de voir qu'on fait au plus n^2 itérations.

En revanche, on ne connaît pas d'algorithme déterministe polynômial pour le calcul des propositions complet (ce problème est le prototype des problèmes NP-complets). Jones et Laeser ont montré que le problème est P-complet pour les clauses de Horn.

PROLOG

PROLOG est un langage de programmation développé par Colmerauer à Marseille, fondé sur le langage des listes de clauses de Horn ordonnées. L'exécution d'un programme PROLOG consiste à essayer de réfuter l'ensemble du programme et des clauses questions. L'interpréteur est un programme de démonstration automatique dont le principe s'inspire du théorème 5 (à peu de choses près).

Exemples de programmation en clauses de Horn d'après Kowalski.

1°) Arithmétique

PLUS(0,x,x) ←
PLUS(S(x),y,S(z)) ← PLUS(x,y,z)

TIMES(0,x,0) ←
TIMES(S(x),y,u) ← TIMES(x,y,v),PLUS(y,v,u)

FACT(0,S(0)) ←
FACT(S(x),u) ← FACT(x,v),TIMES(S(x),v,u)

Exemples de questions :

← PLUS(S(S(0)),S(S(0)),u)	2 + 2 = ?
← TIMES(S(S(0)),S(0),x)	2 x 1 = ?
← TIMES(x,S(S(0)),S(S(S(S(0)))))	? x 2 = 4

On peut effectuer des divisions grâce au programme de multiplication !

← TIMES(x,y,S(S(0)))	trouver les (x,y) tels que xy = 2
← FACT(x,S(S(0)))	?! = 2
← FACT(x,S(S(S(0))))	?! = 3 (!)
	pas de réponse = satisfiable

2°) Manipulation de listes

MEMBER(x,CONS(x,y)) ←
MEMBER(x,CONS(y,z)) ← MEMBER(x,z)

* APPEND(NIL,x,x) ←
* APPEND(CONS(x,y),z,CONS(x,u)) ← APPEND(y,z,u)

DELETE(x,CONS(x,y),y) ←
DELETE(x,CONS(y,z),CONS(y,u)) ← DELETE(x,z,u)

Exemples de questions

← MEMBER(B,CONS(A,CONS(B,CONS(C,NIL))))
← MEMBER(x,CONS(A,CONS(B,CONS(C,NIL))))
← APPEND(CONS(A,NIL),CONS(B,CONS(C,NIL)),x)

3°) Tri

```
*   LE(0,x,T) ←
*   LE(S(x),0,F) ←
*   LE(S(x),S(y),u ) ← LE(x,y,u)

    PERM(NIL,NIL) ←
    PERM(z,CONS(x,y)) ← DELETE(x,z,z'),PERM(z',y)

    ORD(NIL) ←
    ORD(CONS(x,NIL)) ←
    ORD(CONS(x,CONS(y,z))) ← LE(x,y,T),ORD(CONS(y,z))
```

Un programme de tri très simple (et très inefficace) pourrait s'écrire :

```
SORT(x,y) ← PERM(x,y),ORD(y)
```

La procédure QUICKSORT s'obtient en ajoutant aux clauses marquées * ci--dessus les clauses suivantes :

```
SORT(NIL,NIL) ←
SORT(CONS(x,y),w) ← PART(x,y,u,v),SORT(u,u'),SORT(v,v'),
                    APPEND(u',CONS(x,v'),w)

PART(x,NIL,NIL,NIL) ←
PART(x,CONS(x, ),u,CONS(y,v)) ← PART(x,y,u,v),LE(x,y,T)
PART(x,CONS(y,z),CONS(y,u),v) ← PART(x,y,u,v),LE(x,y,F)
```

4°) Analyse syntaxique

On représente une grammaire (simplifiée) du français par l'ensemble des clauses

```
PHRASE(x,y) ← SUBSTANTIF(x,u),VERBE(u,v),SUBSTANTIF(v,y)
SUBSTANTIF(x,y) ← ARTICLE(x,u),NOM(u,y)
SUBSTANTIF(x,y) ← ARTICLE(x,u),NOM(u,v),ADJECTIF(v,y)
```

et un dictionnaire (abrégé) par

```
ARTICLE(x,y) ← L'(x,y)
ARTICLE(x,y) ← LA(x,y)
NOM(x,y) ← STUPIDITE(x,y)
NOM(x,y) ← INTELLIGENCE(x,y)
ADJECTIF(x,y) ← NATURELLE(x,y)
ADJECTIF(x,y) ← ARTIFICIELLE(x,y)
VERBE(x,y) ← ENCOURAGE(x,y)
```

On peut maintenant analyser la phrase :

```
"LA STUPIDITE NATURELLE ENCOURAGE L'INTELLIGENCE ARTIFICIELLE"
1  2         3          4          5  6           7            8
```

en ajoutant l'ensemble de clauses :

LA(1,2) ←
STUPIDITE(2,3) ←
NATURELLE(3,4) ←
.....
ARTIFICIELLE(7,8) ←

et en déclenchant l'analyse par la clause : ← PHRASE(1,8).

EXERCICES.

1) Vérifiez que les méthodes de déduction montante (resp. descendante) correspondent à l'analyse de la phrase montante (resp. descendante).
2) Enrichissez la grammaire ci-dessus, et le dictionnaire, de manière à pouvoir analyser complètement la langue française. Si vous n'y arrivez pas, aidez-vous de méthodes heuristiques, de techniques d'apprentissage, ...etc. Si vous n'y arrivez toujours pas, méditez sur la phrase ci-dessus.

REFERENCES

Andreka H. & Nemeti I.
The generalized completeness of Horn predicate-logic as a programming language.
DAI RR 21, March 76, U. of Edinburgh

Bibel W.
Predicative programming
Report Institut für Informatik, Technische Universität, München

Clark K. & Tärnlund S.
A first-order theory of data and programs
IFIP 77 pp 939-944

Colmerauer A.
Programmation en logique du premier ordre
Groupe d'intelligence artificielle, Marseille-Luminy, Nov 75

Colmerauer A.
Les grammaires de métamorphose
Groupe d'intelligence artificielle, Marseille-Luminy, Nov 75

Chang C. & Lee R.
Symbolic logic and mechanical theorem proving
Academic Press, 1973

Eder G.
A PROLOG-like interpreter for non-Horn clauses
DAI RR 26, Sept 76, U. of Edinburgh

Van Emden M.
Verification conditions as representations for programs
RR CS-76-03 U. of Waterloo, Jan 76.

van Emden M.
A proposal for an imperative complement to PROLOG
RR CS-76-39 U. of Waterloo, Aug 76

van Emden M.
Programming with resolution logic
Machine Intelligence 8

van Emden M.
Computation and deductive information retrieval

van Emden M. & Kowalski R.
The semantics of predicate logic as a programming language.
JACM 23,4 Oct 76 pp 733-742

Galvin F.
Horn sentences
Annals of Math. Logic 1,4 (1970) pp 389-422

Hayes P.
Semantic trees
Ph D thesis, Edinburgh U. 1973

Hayes P.
Computation and deduction
Proc. 2nd MFCS Symposium (1973) pp 105-118

Hayes P.
Simple and structural redundancy in non-deterministic computation

Henschen L.
Semantic resolution for Horn sets
4th IJCAI, Tbilisi 1975.

Henschen L & Wos L.
Unit refutations and Horn sets
JACM 21,4 Oct 74 pp 590-605

Horn A
On sentences which are true of direct unions of algebras
the journal of Symbolic Logic 16,1 March 1951.

Jones N. & Laaser W.
Complete problems for deterministic polynomial time
TCS 3 (1977) pp 105-117

Kowalski R.
Predicate logic as programming language
DCL memo 70, Nov 73, U. of Edinburgh

Kowalski R.
Logic for problem solving
DCL memo 75, March 74, U. of Edinburgh

Kowalski R.
A proof procedure using connection graphs
JACM 22,4, Oct 75, pp 572-595

Kuehner D.
Some special-purpose resolution systems
Machine Intelligence 7 pp 117-128

Lewis H.
Renaming a set of clauses as a Horn set
JACM 25,1, Jan 78. pp 134-135.

Meloni H.
PROLOG : mise en route de l'interpréteur et exercices
Groupe d'intelligence artificielle, Marseille-Luminy, 1976.

Robinson J.A.
A machine-oriented logic based on the resolution principle
JACM 12,1 Jan 65 pp 23-41

Robinson J.A.
Automatic deduction with hyper-resolution
Int. J. of Computer Math. 1 (1965) pp 227-234

Roussel P.
PROLOG : manuel de référence et d'utilisation
Groupe d'intelligence artificielle, Marseille-Luminy, Sept 75.

Tärnlund S.
Logic information processing
Report TRITA-IBADB-1034 1975-11-29
U. of Stockholm

Tärnlund S.
Horn clause computability
Report TRITA-IBADB-1030 1976-05-29
U. of Stockholm

Warren D.
Warplan : a system for generating plans
DCL memo 76, June 74, U. of Edinburgh.

Warren D.
Logic programming and compiler writing
DAI RR 44, Sept 77, U. of Edinburgh