Logique et Résolution:

Démonstration automatique de théorème

I – Principe d'une démonstration automatique

On cherche malgré l'indécidabilité de **LP** à avoir des procédures permettant de conclure, en un temps (seulement) fini mais non nécessairement borné, au fait qu'une expression est ou non un théorème.

On n'envisage que des formules bien formées fermées et dans ces conditions la version sémantique du méta-théorème de la déduction permet d'écrire :

Si $\{H1,\ldots, Hn\} \models B$ alors $\{H1,\ldots, Hn\text{-}1\} \models (Hn \supset B)$

et en réitérant le procédé on obtient

$\{\ \} \models H1 \supset (\ldots (Hn\text{-}1 \supset (Hn \supset B))\ldots)$

soit en utilisant une notation conjonctive des prémisses :

$\models ((H1 \wedge \ldots \wedge Hn) \supset B)$.

Plutôt que montrer cette tautologie on va préférer montrer que la négation de cette formule n'est jamais satisfaite. Autrement dit:

$\models \neg(H1 \wedge \ldots \wedge Hn \wedge \neg B)$

$\equiv$

$\{H1 \wedge \ldots \wedge Hn \wedge \neg B\} \models$ false

II Principe de Résolution

Unification

Définition

Un ensemble E de n termes $\{t_1,\ldots,t_n\}$ est unifiable s'il existe 1 substitution σ telle que $\sigma(t_1) = \ldots = \sigma(t_n)$. σ est appelée unificateur des termes de E.

Algorithme de calcul du pgu

Unifier(E,σ)

Si card(E) = 1 Alors pgu(E) = σ
Return(σ)
Sinon Soient $t_1, t_2 \in$ E tels que $t_1 \neq t_2$
Si $(\exists\tau)\ \tau(t_1) = \tau(t_2)$ Alors $\sigma \leftarrow \tau \circ \sigma$
$E \leftarrow \sigma(E)$
Unifier(E,σ)
Sinon Return(échec)

On lance l'exécution de Unifier avec l'ensemble E supposé unifiable et la "substitution nulle"

Logique et Résolution:

Démonstration automatique de théorème

Théorème 1 (résolution)

Soient C1 et C2 : 2 clauses d'une forme clausale G telles qu'elles n'aient pas de variables en commun [7]

Soient L1 et L2 : 2 ensembles de littéraux : $L1 \subseteq C1$ et $L2 \subseteq C2$,

tels que $L1 \cup \neg L2$ soit unifiable par un pgu σ

Soit la clause $Res(C1, C2) = \sigma.((C1 - L1) \cup (C2 - L2))$ appelée résolvante de C1 et C2

Alors : G et $G \cup \{Res(C1, C2)\}$ sont logiquement équivalentes

Théorème 2 (factorisation)

Soit C une clause de la forme $C = L \cup R$, où L et R sont des ensembles de littéraux, telle que : L est unifiable par un pgu σ

Soit $Fact(C) = \sigma.(\{l\} \cup R)$ appelée facteur de la clause C où $l \in L$

Alors : $C \models Fact(C)$

Théorème 3

Res et Fact sont "complètes" pour la réfutation. Autrement dit : si G est une forme clausale inconsistante (insatisfiable) alors il existe une preuve de la clause vide, au sens de la conséquence logique, à partir de G. On écrira $G \models \square$.

III Mise en oeuvre de l'algorithme de réfutation

Soient $H_1,\ldots, H_n$, B des formules bien formées. On veut démontrer

$\{H_1,\ldots, H_n\} \models B$, c'est-à-dire $\{H_1,\ldots, H_n, \neg B\} \models \square$.

Soit $G = Clause(H_1 \wedge \ldots \wedge H_n \wedge \neg B)$, on applique l'algorithme suivant :

Réfutation(G) :

Si $\square \in G$ Alors Return(succès)
Sinon Soit $P \subseteq \{(C1, C2) \mid C1, C2 \in G \cup Fact(G)\}$
Soit $S \subseteq \{Res(C1, C2) \mid (C1, C2) \in P\}$
Si $S = \varnothing$ Alors Return(échec)
Sinon Soit $T \subseteq S$, $G \leftarrow G \cup T$
Réfutation(G)

IV Stratégies de réfutation linéaires

On choisit une clause C_0 appelée clause centrale de départ.

Les clauses de G sont dites clauses d'entrée.

$G - \{C_0\}$ doit être satisfiable (pour la garantie d'obtention de $\square$)

Toute résolvante s'obtient à partir d'une clause centrale et d'une clause dite de bord dont le choix est restreint par les conditions suivantes :

$C_{n+1} = Res(C_n, B_n)$ où

C_n (resp. C_{n+1}) est 1 clause centrale de niveau n (resp. n+1)

et

B_n est une clause de bord telle que $B_n \in G \cup \{C_k\}_{k<n}$

Logique et Résolution:

Démonstration automatique de théorème

Propriété

Les stratégies de réfutation linéaires sont "complètes".

Exemple

$$G = \{1: C_0 = \mathbf{A} \vee \mathbf{B},\ 2: \neg\mathbf{A} \vee \mathbf{B},\ 3: \mathbf{A} \vee \neg\mathbf{B},\ 4: \neg\mathbf{A} \vee \neg\mathbf{B}\}$$

On remarque que v(**A**) = v(**B**) = v(**C**) = 0 font de C_0 une clause centrale de départ acceptable [10]

La contrainte $C_{n+1} = Res(C_n, B_n)$ où $B_n \in G \cup \{C_k\}_{k<n}$ fait que l'arbre de recherche est cloisonné.

On ne peut jamais tenter une résolution entre 2 clauses centrales dans des branches différentes.

On ne peut combiner une clause centrale qu'avec une autre en remontant dans la colonne cloisonnée ou avec une clause d'entrée.

Dans le tableau ci-après

$C_{n+1} = Res(C_n, B_n)$ est noté :

$n°(C_n) : C_n$
$+\ n°(B_n)$
$n°(C_{n+1}) : C_{n+1}$

Logique et Résolution:

Démonstration automatique de théorème

1: **A∨B**								2: **¬A∨B**							3: **A∨¬B**	4: **¬A∨¬B**
+2 5: **B**								+3 6: **A**							+4 7: ■	
+3 8: **A**				+4 9: **¬A**				+2 10: **B**				+4 11: **¬B**			inutile	
+2 =5	+4 12: **¬B**			+1 =5	+3 13: **¬B**			+3 =6	+4 14: **¬A**			+1 =6	+2 15: **¬A**			
boucle	+1 =8	+2 16: **¬A**	+5 17: □	boucle	+1 18: **A**	+2 =9	+5 19: □	boucle	+1 =10	+3 20: **¬B**	+6 21: □	boucle	+1 22: **B**	+3 =11	+6 23: □	

Logique et Résolution:

Démonstration automatique de théorème

V Stratégies de réfutation linéaires par entrées

Cas particulier des précédentes.

La combinatoire $Res(C_n, B_n)$ due à $B_n \in G \cup \{C_k\}_{k<n}$ augmente avec n.

On veut faire en sorte que la combinatoire reste d'importance fixe.

On se limite à : $B_n \in G$ (G = clauses d'entrées).

Les stratégies de réfutation linéaires par entrées ne sont pas complètes.

Dans l'exemple précédent on n'aboutit à la clause vide qu'en utilisant une clause centrale ancêtre.

Définition

Une clause de Horn est une clause qui a au plus un littéral positif.

Exemple : dans l'exemple précédent, elles sont toutes de Horn sauf $A \vee B$.

Logique et Résolution:
Démonstration automatique de théorème

Propriété

On peut même réduire très fortement le calcul des résolvantes :

$R(G) = G \cup \{Res(C1, C2) \mid C1, C2 \in G\}$

c.à.d.: on se passe de la factorisation

$Res(C1, C2) = \sigma.((C1 - L1) \cup (C2 - L2))$

où L1 = {l1}et L2 = {l2}, soit : L1 et L2 sont des singletons.

c.à.d.: on pratique la résolution binaire (effacement d'un seul littéral par clause)

Dans ces conditions, les stratégies de réfutation linéaires par entrées sont complètes pour des ensembles insatisfiables de clauses de Horn.

VI Stratégie de réfutation de Prolog

C'est une certaine mise en oeuvre d'une stratégie linéaire par entrées contrainte par :

- en principe les clauses sont des clauses de Horn, mais il faut savoir que
  - certains dialectes outrepassent cette contrainte,
  - Prolog propose des prédicats prédéfinis dont certains sont des méta-prédicats de contrôle de la recherche d'une réfutation (ex: not, assert, "cut", findall, freeze, etc.),
  - la plupart des dialectes sortent du cadre de la logique du premier ordre (ordre supérieur à 1),
- la clause centrale de départ est une clause négative de la forme clausale associée à la négation de la conjecture
- il n'y a pas vérification que G — $\{C_0\}$ est satisfiable, donc pas de garantie d'obtention de la clause vide.

Logique et Résolution:
Démonstration automatique de théorème

- la résolution est binaire sans factorisation,
- le littéral effacé de la clause centrale parente est le premier littéral négatif dans l'ordre d'écriture de ses littéraux
- il est remplacé par les littéraux non effacés d'une clause de bord (c'est-à-dire une clause d'entrée) en conservant leur ordre respectif d'écriture :

$$Res(\neg l_1 \vee \neg l_2 \vee \ldots \vee \neg l_n,\ l_1 \vee \neg h_2 \vee \ldots \vee \neg h_m) = \neg h_2 \vee \ldots \vee \neg h_m \vee \neg l_2 \vee \ldots \vee \neg l_n$$

- les clauses d'entrées (qui forment le programme Prolog) sont toujours examinées dans le même ordre : l'ordre d'écriture de ce programme
- la plupart des dialectes obligent les clauses à se regrouper en suites, de faits (atomes) ou de règles (clauses de Horn), utilisant un même nom de prédicat,
- la recherche d'une clause vide se fait en profondeur (et non pas niveau par niveau, c-à-d: en largeur)
- Prolog recherche toutes les clauses vides (et pas seulement la première).

Nota Bene : Pour certaines de ces raisons Prolog n'est pas une stratégie complète

VII Raisonnement naturel

Exprimer les assertions suivantes sous forme de fbf :

« Un malade aime tous les docteurs »
« Aucun malade n'aime les charlatans »

Que peut-on déduire de ces assertions ?

VII Raisonnement naturel

Exprimer les assertions suivantes sous forme de fbf :

« Un malade aime tous les docteurs »
« Aucun malade n'aime les charlatans »

On peut déduire que: « Aucun docteur n'est charlatan »

VII Raisonnement naturel

Exprimer les assertions suivantes sous forme de fbf :

« Un malade aime tous les docteurs »
« Aucun malade n'aime les charlatans »

On peut déduire que: « Aucun docteur n'est charlatan »

Comment ?

Fabriquer des formules (fbf)

Mettre ces fbf sous forme normale prénexe conjonctive, puis sous forme Clausale

Utiliser une stratégie de réfutation pour prouver cette déduction

Logique et Résolution:

Démonstration automatique de théorème

Formules

« Un malade aime tous les docteurs »
H1: $((\exists x)\ M(x) \wedge ((\forall y)\ D(y) \supset A(x,y)))$

« Aucun malade n'aime les charlatans »
H2: $((\forall x)\ M(x) \supset ((\forall y)\ C(y) \supset \neg A(x,y)))$

« Aucun docteur n'est charlatan »
B: $((\forall x)\ D(x) \supset \neg C(x))$

Négation de B
$\neg B : \neg((\forall x)\ D(x) \supset \neg C(x)) \equiv ((\exists x)\ D(x) \wedge C(x))$

$\{H1 \wedge \ldots \wedge Hn \wedge \neg B\} \models \square$

Logique et Résolution:

Démonstration automatique de théorème

Formes clausales

H1: $((\exists x)\ M(x) \wedge ((\forall y)\ D(y) \supset A(x,y)))$
$\{M(a), \neg D(y) \vee A(a,y)\}$

H2: $((\forall x)\ M(x) \supset ((\forall y)\ C(y) \supset \neg A(x,y)))$
$\{\neg M(x) \vee \neg C(z) \vee \neg A(x,z)\}$

$\neg B : ((\exists x)\ D(x) \wedge C(x))$
$\{D(b), C(b)\}$

Ensemble inconsistant G de formes clausales

$$G = \{M(a), \neg D(y) \vee A(a,y), \neg M(x) \vee \neg C(z) \vee \neg A(x,z), D(b), C(b)\}$$

Démonstration automatique de théorème

Démonstration à la manière de Prolog

$b1=M(a)$, $b2=\neg D(y) \vee A(a,y)$, $b3=D(b)$, $b4=C(b)$

$C0= \neg M(x) \vee \neg C(z) \vee \neg A(x,z)$	combiné avec b1	$s=(a\|x)$
$C1= \neg C(z) \vee \neg A(a,z)$	combiné avec b4	$s'=(b\|z)$
$C2= \neg A(a,b)$	combiné avec b2	$s''=(b\|y)$
$C3= \neg D(y)$	combiné avec b3	$s'''= \varepsilon$

$C4 = \square$