Intelligence Artificielle

Logiques

Emmanuel ADAM

Université Polytechnique des Hauts-De-France

UPHF/INSA HdF

Introduction

Logique classiques

- Etude en Philosophie, Logique, Linguistique, Psychologie cognitive, Intelligence Artificielle
- Informatique : théorie et traitement de l'information (Le Robert)
  - Nécessité de représenter la connaissance
  - But : manipulation par " systèmes experts "
  - Objectif : faciliter, aider la décision
- Connaître = Mémoriser + Raisonner
- Représenter = Formaliser + Inférer

Typologie de la connaissance

Les différents types de connaissances

De définition : toujours vraie

Un triangle est un polygone ayant exactement 3 côtés

Evolutive/atemporelle : peut être modifiée

Anne est étudiante à l'UVHC

Incertaine/certaine :

La Lune provient d'une collision de la Terre

Floue/précise : évaluation difficile

Les lendemains de fêtes ne sont pas très productifs

Typique/universelle : peut être contredit

Habituellement, le prof arrive en retard

Ambiguë : plusieurs significations

Nikos savait que que Abi allait gagner.

savoir / se douter ?

Problèmes de représentation

Pas de formalisme idéal

- Modalités : je crois que, je pense que, il est probable que, ...
- Evolutivité : les connaissances changent
- Typicalité et partage de propriété
- Connaissances incomplètes
- ...

$\rightsquigarrow$ pas de formalisme idéal !

Propriété de représentation

Objectif de la représentation

- Adéquation représentationnelle
- Adéquation et efficacité inférentielle
- Efficacité acquisitionnelle et extensibilité
- Simplicité (utilisable par un non informaticien)
- Connaissance explicite
- ...

Famille de connaissances

Savoir et Savoir-Faire

- Représentation déclarative
  - Les connaissances n'ont pas d'ordre
  - Connaissances indépendantes ou liées
  - Pertinence fixée a priori
  - $\bigoplus$ modularité, connaissance stockée une seule fois
- Représentation procédurale
  - Les connaissances ont un caractère opératoire
  - Pertinence définie par un programme qui traite les connaissances
  - $\bigoplus$ facilité de codage, représentation des connaissances sur des opérations

Logique propositionnelle

Eléments de définition

- Origine : de Aristote –> XIX^e (premières formalisations)
- Alphabet $\Sigma = vocabulaire = \{variables, connecteurs\}$
- $variables(atomes) = \{p, r, pepin, charlemagne, terre, \dots\}$
- $connecteurs = \{\neg, \vee, \wedge, \rightarrow, \leftrightarrow\}$
- Formule :
  $A : p \wedge q \rightarrow r$
  $A \vee B, \dots$
- domaine de valeurs $= \{V, F\}$ (en logique booléenne $\{0, 1\}$)

Logique propositionnelle

Identités remarquable et Loi de De Morgan

Propriétés. A étant défini :

- on ne peut avoir une chose et son contraire : $A \wedge \neg A = Faux$
- on peut toujours avoir une chose ou son contraire : $A \vee \neg A = Vrai$
- $\neg(C_1 \vee \ldots C_{n-1} \vee C_n) \equiv \neg(C_1 \vee \ldots C_{n-1}) \wedge \neg C_n \equiv (\neg C_1 \wedge \ldots \wedge \neg C_n)$
- $\neg(C_1 \wedge \ldots C_{n-1} \wedge C_n) \equiv \neg(C_1 \wedge \ldots C_{n-1}) \vee \neg C_n \equiv (\neg C_1 \vee \ldots \vee \neg C_n)$
- $(A \rightarrow B) \equiv (\neg A \vee B)$
- $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
- $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
- $A \wedge (A \vee B) \equiv (A)$
- $A \vee (A \wedge B) \equiv (A)$

Logique propositionnelle : Clauses

Notations équivalentes

Soit F_0 une formule, il existe F_1, F_2, F_3, F_4 équivalentes à F_0 tq :

- F_1 : n'utilise que les connecteurs $\vee, \neg$
- F_2 : n'utilise que les connecteurs $\wedge, \neg$
- F_3 : n'utilise que les connecteurs $\rightarrow, \neg$
- F_4 : n'utilise que les connecteurs $\vee, \wedge, \neg$
  - Si $F_4 = \bigwedge_{i=1}^{n} C_i = C_1 \wedge \ldots \wedge C_n$ où

    $C_i = \bigvee_{j=1}^{k} (lit_i) = lit_1 \vee \ldots \vee lit_k$

    lit_i litéral de la forme a ou $\neg a$ Alors les C_i sont appelées **clauses**

    et F_4 est une **forme normale conjonctive**
  - Si $F_4 = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} lit_{ij}$

    Alors F_4 est une **forme normale disjonctive**

Logique propositionnelle : Clauses de Horn

Clauses de Horn

Les clauses de Horn sont des clauses contenant *au plus* 1 *littéral positif*.
Trois types de clauses sont définis :

stricte : de la forme $p \vee \neg n_1 \vee \ldots \vee \neg n_m, m \geq 1$

positive : de la forme p

négative : de la forme $\neg n_1 \vee \ldots \vee \neg n_m, m \geq 1$

Logique propositionnelle

Axiomes et règles

- $A1 : A \rightarrow (B \rightarrow A)$
- $A2 : (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
- $A3 : (\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$
- règle de réduction : modus ponens : $A, (A \rightarrow B) \vdash B$

Logique propositionnelle : Théorèmes

Théorème

Un théorème se déduit à partir des axiomes, d'autres théorèmes et de règles de réduction (ici le modus ponens).

Ne nécessitant pas d'hypothèse initiales, le théorème s'écrit : $\vdash F$.

Démontrer que $p \rightarrow p$ est un théorème.

$A2 : (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
$F1 : ((p \rightarrow ((q \rightarrow p) \rightarrow p)) \rightarrow ((p \rightarrow (q \rightarrow p) \rightarrow (p \rightarrow p)))$
$A1 : A \rightarrow (B \rightarrow A)$
$F2 : (p \rightarrow ((q \rightarrow p) \rightarrow p))$
$F3 : ((p \rightarrow (q \rightarrow p) \rightarrow (p \rightarrow p))$
$A1 : A \rightarrow (B \rightarrow A)$
$F4 : (p \rightarrow (q \rightarrow p))$
$F5 : (p \rightarrow p)$

Partant "de rien" (des axiomes et autres théorèmes), on déduit $p \rightarrow p$.
On peut écrire $\vdash p \rightarrow p$
$p \rightarrow p$ est un théorème de la logique propositionnelle.

$A_2 : A \backslash p, B \backslash (q \rightarrow p), C \backslash p$

$A_1 : A \backslash p, B \backslash (q \rightarrow p)$
$m.p.F2, F1$

$SA_1 : A \backslash p, B \backslash q$
$m.p.F4, F3$

Logique propositionnelle : Théorèmes

Propositions

Sachant que $\vdash A \rightarrow A$:

- Proposition 1 : $\forall A\ , A \rightarrow A$
- Proposition 2 : Si $A_1, A_2, \ldots, A_{n-1} \vdash A_n \rightarrow B$
  Alors $A_1, A_2, \ldots, A_{n-1}, A_n \vdash B$
- Proposition 3 : Si $A_1, A_2, \ldots, A_{n-1}, A_n \vdash B$
  Alors $A_1, A_2, \ldots, A_{n-1} \vdash A_n \rightarrow B$

Logique propositionnelle : Théorèmes

Théorèmes

Les formules suivantes sont des théorèmes :

- $T_0 : (A \rightarrow A)$
- $T_1 : ((A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)))$
- $T_2 : (B \rightarrow ((B \rightarrow C) \rightarrow C))$
- $T_3 : (\neg B \rightarrow (B \rightarrow C))$
- $T_4 : (\neg\neg B \rightarrow B)$
- $T_5 : (B \rightarrow \neg\neg B)$
- $T_6 : ((A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A))$
- $T_7 : (B \rightarrow (\neg C \rightarrow \neg(B \rightarrow C)))$
- $T_8 : ((B \rightarrow A) \rightarrow (\neg B \rightarrow A) \rightarrow A))$

Logique propositionnelle : Statisfiabilité, tautologie

Tables de vérité

A	B	$B \rightarrow A$	$A \rightarrow (B \rightarrow A)$
F	F	V	V
F	V	F	V
V	F	V	V
V	V	V	V

- lorsque la formule F est *Vraie* pour toute interprétation (ici de A et B), on écrit $\models F$
- Les axiomes, les théorèmes sont tous des **tautologies** (toujours vrais)
- une formule F est **insatisfiable** (inconsistante) s'il n'existe pas d'interprétation I où $I(F) = Vrai$
- une formule F est **satisfiable** (consistante) s'il existe un interprétation I où $I(F) = Vrai$
- une formule F est **valide** si elle est vraie pour toute interprétation
- un problème de recherche d'assignation de variables pour satisfaire une formule est dit **problème SAT**

Logique propositionnelle : résolution

Soit $F = ((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$
$F = \neg((\neg p \vee \neg q) \vee r) \vee (\neg p \vee (\neg q \vee r))$

Vérifions que F est une tautologie par table de vérité :

p	q	r	¬	(	(	¬	p	∨	¬	q	)	∨	r	)	∨	(	¬	p	∨	(	¬	q	∨	r	)	)
T	T	T	F		F	F	T	F	F	T	T	T	T		F	T	T	F	T	T	T					
T	T	F	T		F	F	T	F	F	T	F	F	T		F	T	F	F	T	F	F					
T	F	T	F		F	F	T	T	T	F	T	T	T		F	T	T	T	F	T	T					
T	F	F	F		F	F	T	T	T	F	T	F	T		F	T	T	T	F	T	F					
F	T	T	F		T	F	T	F	T	T	T	T	T		T	F	T	F	T	T	T					
F	T	F	F		T	F	T	F	T	T	T	F	T		T	F	T	F	T	F	F					
F	F	T	F		T	F	T	T	F	T	T	T	T		T	F	T	T	F	T	T					
F	F	F	F		T	F	T	T	F	T	F	T	T		T	F	T	T	F	T	F					

F est donc vraie pour toute interprétation de p, q, r ;
on note $\models F$

2^n lignes sont nécessaires pour une formule à n variables !

Logique propositionnelle : résolution

Théorème du principe de déduction

- A est conséquence d'un ensemble de formules $\mathbb{F}$ si $\{A\} \cup \mathbb{F}$ est **satisfiable**.
- trouver une interprétation qui **SATistasse** $\{A\} \cup \mathbb{F}$.
- ou prouver que $\{\neg A\} \cup \mathbb{F}$ est **INSATisfiable (UNSAT)**
- 2^n lignes par tables de vérités (pour n atomes).
- $\Rightarrow$ utiliser un algorithme dédié.

Logique propositionnelle : Algorithme de Quine

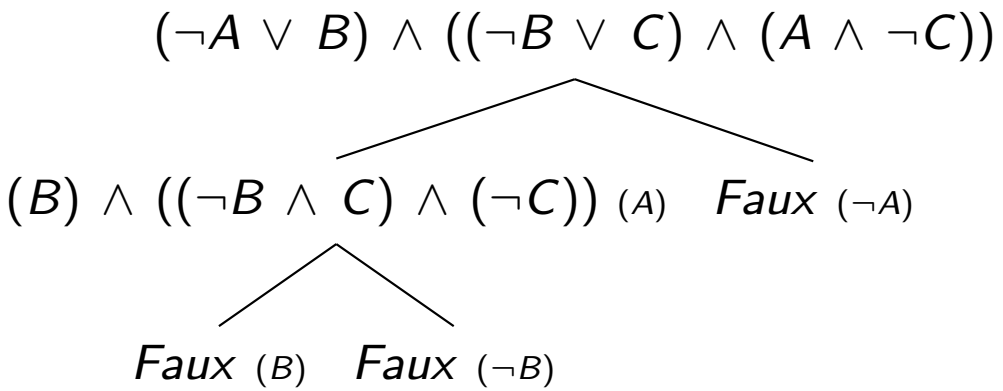
Arbre de résolution

- construction d'un arbre binaire d'une formule F constituée des atomes $p_1, \ldots, p_n$
- chaque arc est étiqueté par un atome p_i ou $\neg p_i$
- nous avons alors deux mondes, l'un où p_i est vrai, l'autre où $\neg p_i$ est vrai
- évaluation partielle de la formule à chaque nœud

Logique propositionnelle : Algorithme de Quine

Prouvez que $F = \neg((A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)))$ n'est pas valide.

$F = \neg(\neg(\neg A \vee B) \vee (\neg(\neg B \vee C) \vee (\neg A \vee C)))$
$F = (\neg A \vee B) \wedge ((\neg B \wedge C) \wedge (A \wedge \neg C))$



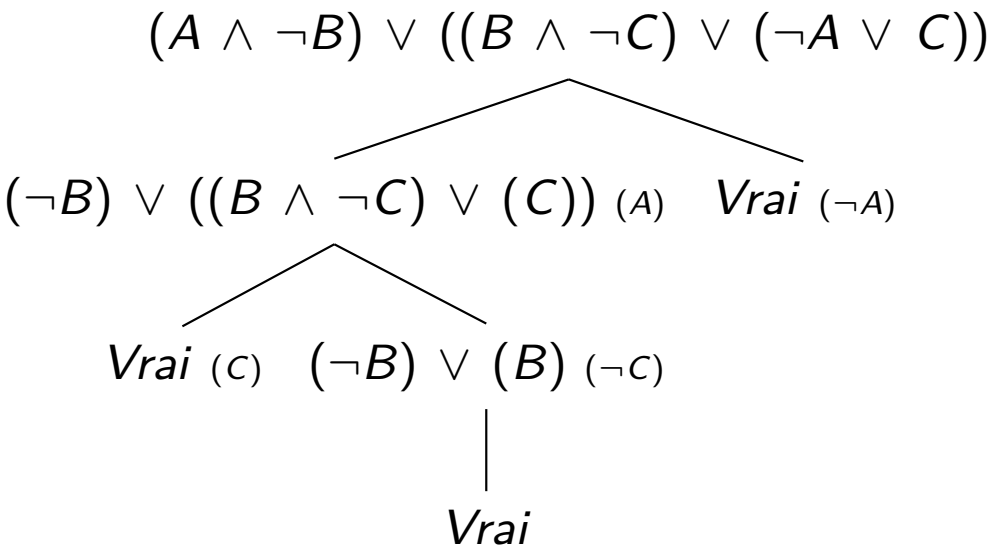
Toutes les branches (tous les "mondes") mènent à des feuilles fausses.
Il n'existe pas de modèle pour F : F est **insatisfiable**.

Donc son opposé $((A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)))$ est **valide**.

Logique propositionnelle : Algorithme de Quine

Prouvez que $F = ((A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)))$ est valide.

$F = (\neg(\neg A \vee B) \vee (\neg(\neg B \vee C) \vee (\neg A \vee C)))$
$F = ((A \wedge \neg B) \vee ((B \wedge \neg C) \vee (\neg A \vee C)))$



Toutes les branches (tous les "mondes") mènent à des succès.
Donc F est **valide**.
L'algo conclue cela sans avoir à balayer tous les cas !

Les modèles trouvés sont : $(\neg A)$, $(A \wedge C)$, $(A \wedge \neg C)$.

Logique propositionnelle : preuve par réfutation

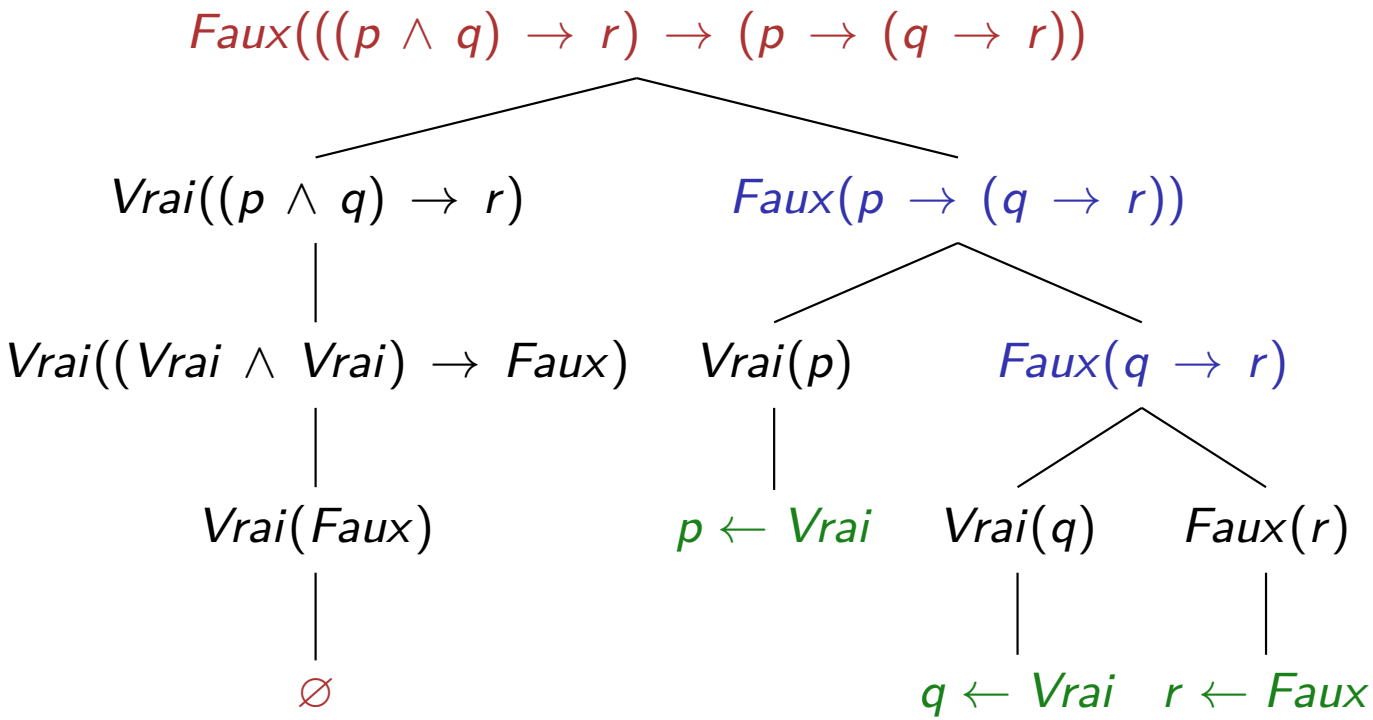
Algorithme de Réduction

- preuve par réfutation (ou falsification)
- Pour toute formule F de la forme $A \rightarrow B$:
  on tente de vérifier si elle peut être fausse :
  $A \rightarrow B$ **est faux ssi** $A = Vrai \wedge B = Faux$
  - Créer un nœud pour évaluer $Vrai(A)$
  - Créer un nœud pour évaluer $Faux(B)$
- jusqu'à obtention d'un modèle ou d'une contradiction

Logique propositionnelle : résolution

Algorithme de Réduction

Démontrer que $((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$ est valide
Démontrer que l'on ne peut avoir
$Faux((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$



Il est impossible que ce soit faux (contradiction), donc c'est vrai !

Logique propositionnelle : Résolvante

Algorithme par Résolvante

- soit F une formule d'un *forme normale conjonctive* :
  $F = \bigwedge_{i=1}^{n} C_i$ avec $C_i = \bigvee_{j=1}^{k} p_j$
- Soient C_i et C_j telles que $p \in C_i$ et $\neg p \in C_j$
- Soit $R_{ij} = C_i \setminus \{p\} \cup C_j \setminus \{\neg p\}$
- alors F et $F \cup R_{ij}$ sont équivalents
- R_{ij} est appelée *résolvante* de C_i et C_j

Logique propositionnelle : Résolvante

Exemple de résolvante

$F_1 = \{(p \vee q \vee r), (\neg p \vee t)\}$
Choix de résolvante par p : $R_1 = (p \vee q \vee r) \setminus p \cup (\neg p \vee t) \setminus \neg p$
$R_1 = (q \vee r \vee t)$
$F_1' = \{(p \vee q \vee r), (\neg p \vee t), (q \vee r \vee t)\}$ est équivalent à F_1

$F_2 = \{(\neg a \vee b), (a)\}$
Choix de résolvante par a : $R_2 = (\neg a \vee b) \setminus \neg a \cup (a) \setminus a$
$R_2 = (b)$
$F_2' = \{(\neg a \vee b), (a), (b)\}$ est équivalent à F_2

Logique propositionnelle : algorithme de résolution

La formule doit être une forme normale conjonctive ET les clauses doivent être des clauses de Horn.

procedure RESOLUTION(F)
  $fin \leftarrow Faux$
  while $\neg fin \wedge \varnothing \notin F$ **do**
    $fin \leftarrow Vrai$
    choisir $P \in F$, tq $P = (p)$ (clause positive)
    if $P \neq \emptyset$ **then**
      choisir $C \in F$, tq $\neg p \in C$
      if $C \neq \emptyset$ **then**
        $R \leftarrow \text{RESOLVANTE}(P, C) = C \setminus (\neg p)$
        $F \leftarrow (F \setminus C) \cup R$
        $fin \leftarrow Faux$
      end if
    end if
  end while
  if $\varnothing \in F$ **then return** "inconsistance"
  else return "consistance"
  end if
end procedure

Logique propositionnelle : résolution

Clauses de Horn

Il faut prouver l'inconsistance de $F = \{\neg p \vee r, \neg r \vee s, p, \neg s\}$

Prenons $P = p$ et $C = (\neg p \vee r)$, Alors $R = \varnothing \cup \{(r)\} = \{(r)\}$
$F = F \setminus C \cup R = \{\neg r \vee s, p, \neg s, r\}$

Prenons $P = r$ et $C = (\neg r \vee s)$, Alors $R = \varnothing \cup \{(s)\} = \{(s)\}$
$F = F \setminus C \cup R = \{p, \neg s, r, s\}$

Prenons $P = s$ et $C = (\neg s)$, Alors $R = \varnothing \cup \varnothing = \{\varnothing\}$
$F = F \setminus C \cup R = \{s, p, \varnothing, r\}$

$\varnothing \in F$ donc F est inconsistant

Logique propositionnelle : Davis & Putman

Algorithme de Davis & Putman

- soit F une formule d'un *forme normale conjonctive* :
  $F = \bigwedge_{i=1}^{n} C_i$ avec $C_i = \bigvee_{j=1}^{k} lit_{ij}$
- On note *lit* un littéral, et lit_c son complémentaire (sa négation)
- $\square$, la clause vide est insatisfiable
- $\varnothing$, l'ensemble vide de toute clause est satisfiable

Logique propositionnelle : Davis & Putman

Règles de Davis & Putman

Appliquer les règles suivantes jusqu'à ce qu'aucune ne puisse plus s'appliquer ; lorsque plusieurs règles sont applicables, on choisit celle de plus petit numéro :

- *R*1 Enlever les tautologies (clause contenant *lit* et lit_c)
- *R*2 Si l'une des clauses ne possède qu'un seul littéral *lit*, enlever toutes les clauses contenant *lit* et enlever dans les autres clauses toutes les occurrences de lit_c.
- *R*3 Si *lit* apparaît dans certaines clauses et que lit_c n'apparaît pas, enlever toutes les clauses contenant *lit*.
- *R*4 Si une clause C_i a tous ses littéraux présents dans une clause C_j, ôter C_j.
- *R*5 Si *lit* et lit_c sont présents dans des clauses, remplacer celui-ci par deux ensembles de clauses :
  - un ensemble obtenu en enlevant toutes les clauses contenant *lit* et toutes les occurrences lit_c
  - un ensemble obtenu en enlevant toutes les clauses contenant lit_c et toutes les occurrences *lit*

Logique propositionnelle : Davis & Putman

Algorithme de Davis & Putman

Exemple : prouver $\models ((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$

$\Rightarrow \{((p \wedge q) \rightarrow r)\} \models (p \rightarrow (q \rightarrow r))$?

$\Rightarrow \{((p \wedge q) \rightarrow r), p\} \models (q \rightarrow r)$?

$\Rightarrow \{((p \wedge q) \rightarrow r), p, q\} \models r$?

$\Rightarrow$ on cherche à prouver l'**inconsistance** de

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

$\Rightarrow$ transformation en clauses : $F = \{(\neg(p \wedge q) \vee r), (p), (q), (\neg r)\}$

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

$\Rightarrow$ Application de $R2$ ($\neg r$ isolé) : $F = \{(\neg p \vee \neg q), (p), (q)\}$

$\Rightarrow$ Application de $R2$ (q isolé) : $F = \{(\neg p), (p)\}$

$\Rightarrow$ Application de $R2$ (p isolé) : $F = \{\square\}$

$\Rightarrow F$ est inconsistant donc $\{((p \wedge q) \rightarrow r), (p), (q)\} \models r$ est vrai

donc $((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$ est vrai

Logique propositionnelle : Davis, Putman, Logeman, Loveland

Règles de DPLL

L'algorithme de D.P.L et L est basé sur l'algo de Davis et Putman et l'algorithme de Quine.

procedure DPLL(F)
  if $F = C$ and C coherent **then**
    return true
  end if
  if $\square \in F$ **then**
    return false
  end if
  for all $C \in F$ **do**
    remove C_i where $C \in C_i$
  end for
  for all $C \in F, C = (lit)$ **do**
    remove C_i where $lit \in C_i$
    remove $\neg lit$ in C_i where $\neg lit \in C_i$
  end for
  choose lit_i
  return $DPLL(F \wedge lit_i) \vee DPLL(F \wedge \neg lit_i)$
end procedure

Logique propositionnelle : DPLL

Algorithme DPLL

Exemple : $\Rightarrow$ on cherche à prouver l'**inconsistance** de
$F = \{(\neg p \vee \neg q \vee r), (p), (q), (\neg r)\}$
$\Rightarrow$ traitement de (p) : $F = \{(\neg q \vee r), (q), (\neg r)\}$
$\Rightarrow$ traitement de (q) : $F = \{(r), (\neg r)\}$
$\Rightarrow$ traitement de (r) : $F = \{()\}$
$F = \{(\square)\}$
$\Rightarrow F$ est inconsistant donc $\{((p \wedge q) \rightarrow r), (p), (q)\} \models r$ est vrai

donc $((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$ est vrai

Logique propositionnelle : DPLL

Algorithme DPLL

Exemple : $\Rightarrow$ on cherche à prouver l'**inconsistance** de
$F = \{(a \vee b \vee \neg d), (d \vee c), (\neg a \vee b), (\neg a \vee \neg c), (\neg b \vee c), (\neg c \vee a)\}$
$\Rightarrow$ séparation par (a) et $(\neg a)$: $F1 = \{(d \vee c), (b), (\neg c), (\neg b \vee c)\}$
$F2 = \{(b \vee \neg d), (d \vee c), (\neg b \vee c), (\neg c)\}$
Traitement de $F1$
- retrait de (b) : $F1 = \{(d \vee c), (\neg c), (c)\}$
- retrait de (c) : $F1 = \{()\}$
- $\square \in F1$, donc $F1$ insatisfiable

Traitement de $F2$
- retrait de $(\neg c)$: $F2 = \{(b \vee \neg d), (d), (\neg b)\}$
- retrait de (d) : $F2 = \{(b), (\neg b)\}$
- retrait de (b) : $F2 = \{()\}$
- $\square \in F2$, donc $F2$ insatisfiable

$F = F1 \cup F2$

$F1$ et $F2$ insatisfiables $\Rightarrow$ F est insatisfibable, i.e. inconsistant

Logique propositionnelle : résolutions

Plusieurs algorithmes

- un **problème SAT** (de satisfaction booléenne) peut donc être résolu selon différentes méthodes, algorithmes pour prouver la vérité ou l'insconsistance d'une formule
- table de vérité ; vérifie si F est logiquement valide : $\models F$
- raisonnement par déduction, à partir d'axiomes et de règles d'inférence ; vérifie si F est prouvable : $\vdash F$
- algorithme de Quine
- algorithme de réduction (par réfutation)
- algorithme de Davis & Putman
- algorithme par clauses de Horn
- ...
- c'est un problème **NP complet**

Logiques

Intérêt au temps du Machine Learning ?

- Les logiques (classique, des prédicats, modales, temporelles) s'appliquent à des problèmes de SATisfaction
- Elles ont pour objectif de fournir des solutions, des réponses précises et définitives à des problèmes à partir de règles et d'une description précise du problème
- La solution peut être justifiée et expliquée
- Au contraire, une solution proposée par un algo basé sur le machine learning donnera une solution "probable" sans justification autre que "habituellement ça fonctionne bien"
- Pour les problèmes de planification (ex. trajectoires), vérification (ex. circuits intégrés), . . ., la logique est donc préférée