Avantages et inconvénients de la logique propositionnelle

:-) La logique propositionnelle est *déclarative* : les éléments syntaxiques correpondent à des faits

:-) La logique propositionnelle permet d'exprimer de l'information sous forme partielle, disjonctive ou négative (contrairement à la plupart des structures ou des bases de données)

:-) La logique propositionnelle est *compositionnelle* : la signification de $B_{1,1} \wedge P_{1,2}$ est obtenue à partir des significations de $B_{1,1}$ et $P_{1,2}$

:-) La sémantique (signification) en logique propositionnelle est *indépendante du contexte* (contrairement au langage naturel par exemple)

:-(La logique propositionnelle a un pouvoir d'expressivité limité (contrairement au langage naturel par exemple)
Ex. : on ne peut pas dire "les puits provoquent des courants d'air dans les cases adjacentes" sans écrire une phrase pour chaque case

Aperçu

Logique du premier ordre

Tantdis que la logique propositionnelle suppose que le monde contienne des *faits*, la logique du 1er ordre (comme le langage naturel) suppose que le monde contienne

- *Des objets* : personnes, maisons, nombres, théories, couleurs, ...
- *Des relations* : rouge, rond, petit, ...,
  frère de, plus grand que, appartenance, inclusion, possession, ...
- *Des fonctions* : le père, le meilleur ami, le troisième élément, ...

Le monde du Wumpus en logique du premier ordre

- Une base de connaissances KB est un ensemble d'expressions du 1er ordre
- Tell(KB, e) ajoute une proposition e dans la base
- Ask(KB, S) retourne quelques/tous *substitutions* σ tels que $KB \models S\sigma$
- Exemple : au temps 5, l'agent perçoit un odeur et du courant d'air
  Tell(KB, $Percept([Smell, Breeze, None], 5)$)
  Ask($KB, \exists a Action(a, 5)$)
  la base KB répond la substitution $\{a/Shoot\}$

Représentation de l'environnement

- Perception au temps t

$$Percept([Smell/none, Breeze/none, Glitter/none], t)$$

- Actions : termes $Turn(Right)$, $Turn(Left)$, $Forward$, $Shoot$, $Grab$, $Release$
- Cases : la case (i, j) est représentée par le terme $[i, j]$.
- Relation adjacence

$$\forall x, y, a, b\ (Adjacent([x, y], [a, b]) \Leftrightarrow [a, b] \in \{[x+1, y], [x-1, y], [x, y+1], [x, y-1]\})$$

- Puits : prédicat unaire Pit
- Wumpus : constante $Wumpus$, fonction $Home(Wumpus)$ désigne la case de Wumpus
- Position de l'agent au temps t est la case s : $At(Agent, s, t)$

Représentation de l'environnement

- Règles relatives aux perceptions

$$\forall b, g, t\ (Percept([Smell, b, g], t) \implies Smelt(t))$$

$$\forall s, b, t\ (Percept([s, b, Glitter], t) \implies AtGold(t))$$

- Réflexe :

$$\forall t\ (AtGold(t) \implies Action(Grab, t))$$

- Réflex avec état interne : as-t-on déjà l'or ?

$$\forall t\ (AtGold(t) \wedge \neg Holding(Gold, t) \implies Action(Grab, t))$$

$Holding(Gold, t)$ ne peut pas être observé $\Rightarrow$ garder la trace de changements est indispensable

Déduire des propriétés cachées

- Propriété de position :

$$\forall x, t\ (At(Agent, x, t) \wedge Smelt(t) \implies Smelly(x))$$

$$\forall x, t\ (At(Agent, x, t) \wedge Breeze(t) \implies Breezy(x))$$

- Les cases proches d'un puit ont des courants d'air :
  - solution *diagnostique* : la règle infère la cause à partir de l'effet

$$\forall y\ (Breezy(y) \implies \exists x\ (Pit(x) \wedge Adjacent(x, y)))$$

  - solution *causale* : la règle infère l'effet à partir de la cause

$$\forall x, y\ (Pit(x) \wedge Adjacent(x, y) \implies Breezy(y))$$

- *Définition* du prédicat $Breezy$

$$\forall y\ Breezy(y) \equiv \exists x\ (Pit(x) \wedge Adjacent(x, y))$$

Garder une trace de changements

- Les faits existent dans une situation donnée et non éternellement, ex. $Holding(Gold, Now)$ plutôt que $Holding(Gold)$
- Le *calcul des situations* est une possibilité pour représenter les changements :
  ajouter un argument de situation à chaque prédicat non éternel
- Les *systèmes de planification* sont des systèmes de raisonnement spéciaux désignés à produire des plans (suites d'actions) plus efficace qu'un systèmes de raisonnement général.

Modus Ponens généralisé

$$\frac{(p1 \wedge p_2 \wedge \cdots \wedge p_n \implies q), \quad p'_1, \quad , p'_2, \ldots, \quad p'_n}{q\delta}$$

avec δ un unificateur tel que pour tout i, $p_i\delta = p'_i\delta$

- Modus Ponens généralisé (MPG) est utilisé avec les bases de connaissances de *clauses définies* (clauses contenant *exactement* un litéral positif)
- Toutes les variables sont supposées quantifiées universellement

Exemple d'une base de connaissances

- Soit l'énoncé suivant (conservé dans la version en anglais pour la cohérence avec la référence AIMA)
  The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
- Prouver que le Colonel West est un criminel

Représenter les connaissances

- ... it is a crime for an American to sell weapons to hostile nations :

  $$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$$

- The country Nono, an enemy of America ...

  $$Enemy(Nono, America)$$

- Nono ... has some missiles, i.e., $\exists x\ Owns(Nono, x) \wedge Missile(x)$

  $$Owns(Nono, M_1)$$
  $$Missile(M_1)$$

- ... all of its missiles were sold to it by Colonel West

  $$Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$$

- West, who is American ...

  $$American(West)$$

- Missiles are weapons :

  $$Missile(x) \implies Weapon(x)$$

- An enemy of America counts as "hostile" :

  $$Enemy(x, America) \implies Hostile(x)$$

Preuve par chaînage avant

- Algorithme de chaînage avant
  Lorsqu'un nouveau fait p est ajouté à la base de connaissances :
  pour chaque règle telle que p s'unifie avec un prémisse et si les autres prémisses sont connus,
  alors ajouter la conclusion à la base de connaissances et continuer le chaînage
- Une expression α est inférée de la base KB si elle est une conclusion dans le chaînage

Algorithme de chaînage avant

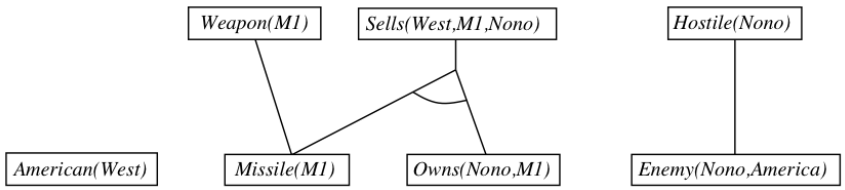
function FOL-FC-ASK(KB, α) returns a substitution or *false*

repeat until *new* is empty
  new $\leftarrow \{\}$
  for each sentence r in KB do
    $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)
    for each θ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$
        for some $p'_1, \ldots, p'_n$ in KB
      $q' \leftarrow$ SUBST(θ, q)
      if q' is not a renaming of a sentence already in KB or *new* then
        add q' to *new*
        $\phi \leftarrow$ UNIFY(q', α)
        if ϕ is not *fail* then return ϕ
  add *new* to KB
return *false*

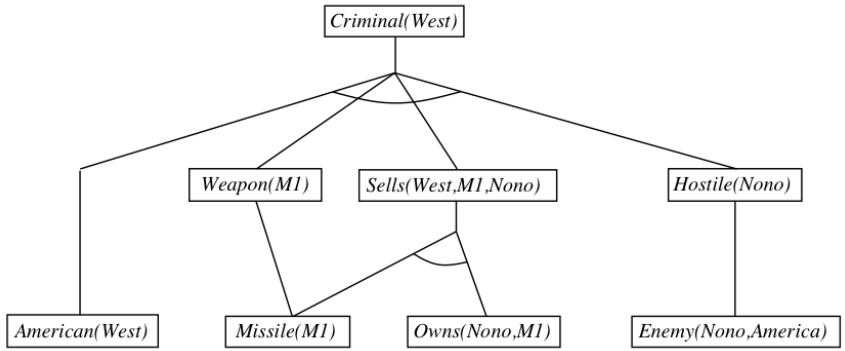
Exemple de preuve par chaînage avant

American(West) *Missile(M1)* *Owns(Nono,M1)* *Enemy(Nono,America)*

Exemple de preuve par chaînage avant



Exemple de preuve par chaînage avant



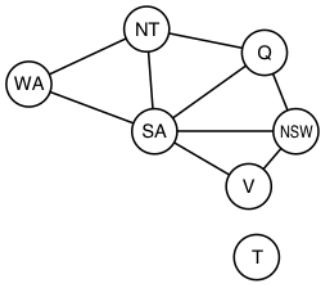
Propriétés du chaînage avant

- Correct et complet pour les clauses définies du 1er ordre
- *Datalog* = clauses définies du 1er ordre + *sans fonctions*
  Le chaînage avant se termine pour Datalog en un nombre d'itérations polynomial : au plus pn^k littéraux
- En général le chaînage avant peut ne pas s'arrêter si $KB \nvdash \alpha$
- Ceci est inévitable : le problème de déduction pour les clauses définies et semi-décidable

Efficacité du chaînage avant

- Simple observation : à l'itération k il n'est pas nécessaire de considérer les règles dont aucune prémisse est ajoutée à l'itération $k-1$
  $\Longrightarrow$ unifier les règles dont la prémisse contient un littéral nouvellement ajouté
- L'unification peut être coûteuse
- *Indexation* permet d'accéder aux faits connus en temps constant
  ex., query $Missile(x)$ retrouve $Missile(M_1)$
- Unification entre prémisses et des faits connus est NP-Difficile
- Le chaînage avant est utilisé largement dans les *bases de données déductives*

Unification est NP-Difficile



$diff(WA, NT) \wedge diff(WA, SA) \wedge$
$diff(NT, SA) \wedge diff(NT, Q) \wedge$
$diff(Q, SA) \wedge diff(Q, NSW) \wedge$
$diff(SA, NSW) \wedge diff(SA, V) \wedge$
$diff(NSW, V) \Longrightarrow Coloriable$
$diff(red, blue)$ $diff(red, green)$
$diff(blue, green)$ $diff(blue, red)$
$diff(green, blue)$ $diff(green, red)$

- *Coloriable* est inféré si et seulement si le CSP a une solution
- 3SAT est un cas spécial de CSP
- Unification est donc NP-Difficile

Preuve par chaînage arrière

- Algorithme de chaînage arrière
  Procéder à partir du but à prouver.
  Si un but s'unifie avec la conclusion d'une règle alors ajouter les prémisses de la règle à la liste des buts à prouver et continuer le chaînage
- Une expression α est inférée de la base KB si l'algorithme trouve un unificateur qui unifie les buts et les faits existant dans la base

Algorithme de chaînage arrière

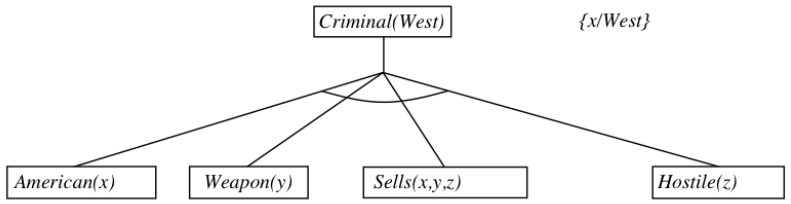
function FOL-BC-ASK($KB, goals, \theta$) returns a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query (θ already applied)
    θ, the current substitution, initially the empty substitution $\{\}$
  local variables: $answers$, a set of substitutions, initially empty

  if $goals$ is empty then return $\{\theta\}$
  $q' \leftarrow$ SUBST(θ, FIRST($goals$))
  for each sentence r in KB
    where STANDARDIZE-APART(r) = $(p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow$ UNIFY(q, q') succeeds
    $new\_goals \leftarrow [p_1, \ldots, p_n | \text{REST}(goals)]$
    $answers \leftarrow$ FOL-BC-ASK(KB, new_goals, COMPOSE(θ', θ)) $\cup$ $answers$
  return $answers$

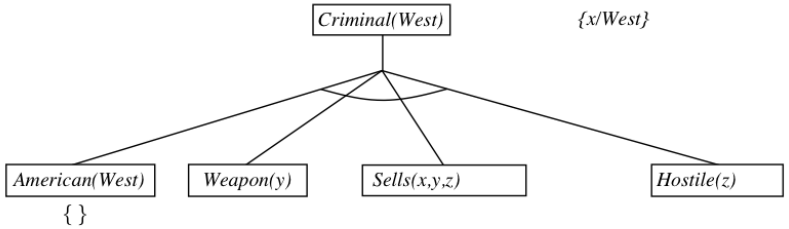
Exemple de preuve par chaînage arrière

Criminal(West)

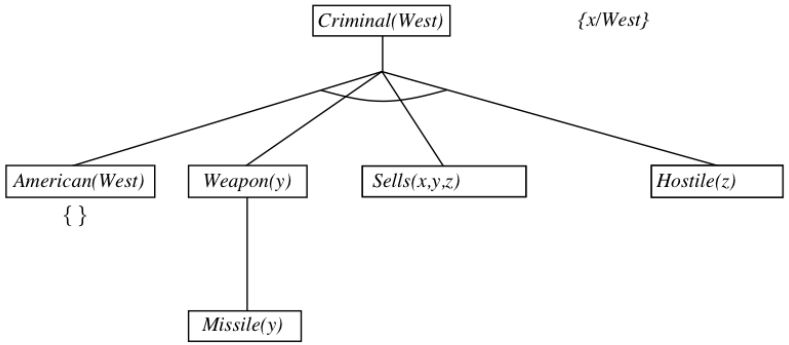
Exemple de preuve par chaînage arrière



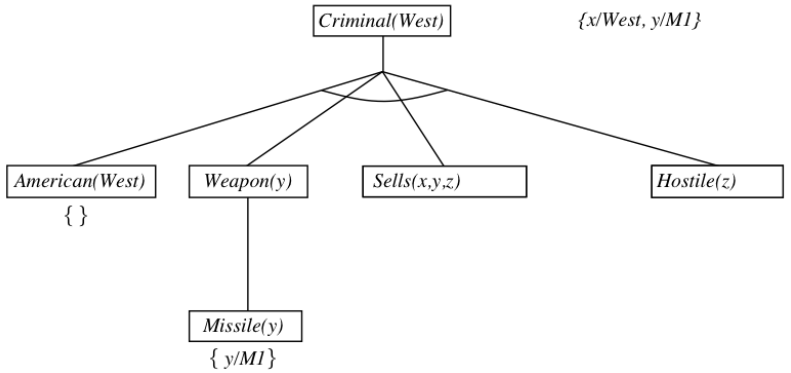
Exemple de preuve par chaînage arrière



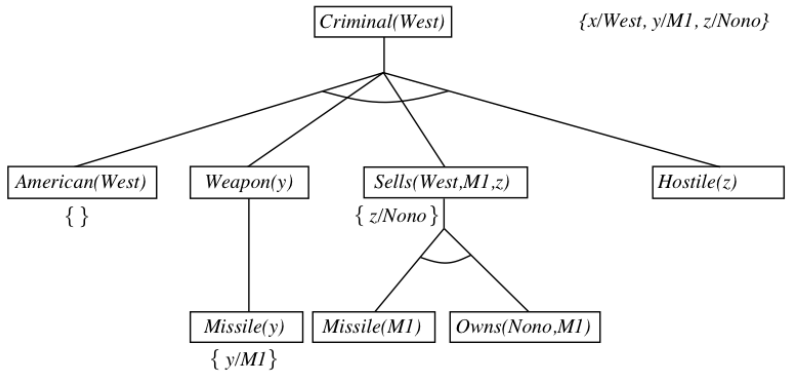
Exemple de preuve par chaînage arrière



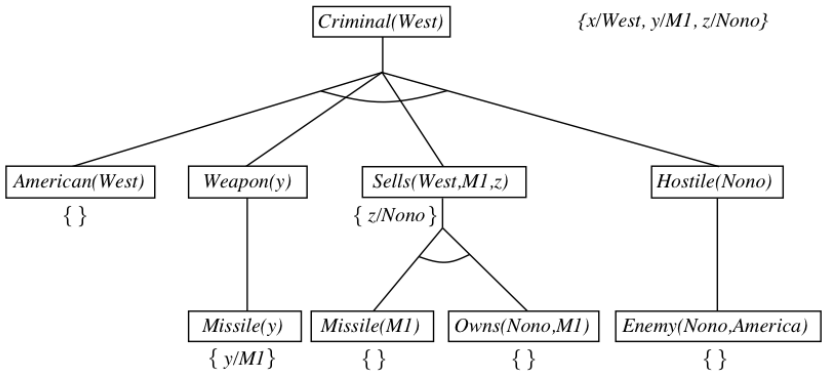
Exemple de preuve par chaînage arrière



Exemple de preuve par chaînage arrière



Exemple de preuve par chaînage arrière



Propriétés du chaînage arrière

- Algorithme de recherche en profondeur : l'espace utilisé est linéaire en taille de preuve
- Incomplet à cause de boucles infinis
  $\Longrightarrow$ vérifier si un but est déjà prouvé
- Chaînage arrière est utilisé en *programmation logique*

Résolution

$$\frac{C_1 \vee A, \quad C_2 \vee \neg B}{(C_1 \vee C_2)\theta}$$

avec $A\theta = B\theta$

- La résolution est complète pour la logique du 1er ordre
- Preuve par résolution : $KB \vdash \alpha$ si la résolution de la forme normale conjonctive de $KB \wedge \neg\alpha$ produit la clause vide

Transformer en forme normale conjonctive (1)

Toute personne qui aime tous les animaux est aimé par quelqu'un :
$\forall x(\forall y Animal(y) \Longrightarrow Loves(x,y)) \Longrightarrow \exists y Loves(y,x)$

- Elimination d'implication

$$\forall x(\neg\forall y(\neg Animal(y) \vee Loves(x,y)) \vee \exists y Loves(y,x))$$

- Entrer $\neg$: $\neg\forall x p \equiv \exists x \neg p, \quad \neg\exists x p \equiv \forall x \neg p$:

$$\forall x(\exists y \neg(\neg Animal(y) \vee Loves(x,y)) \vee \exists y\ Loves(y,x))$$
$$\forall x(\exists y(Animal(y) \wedge \neg Loves(x,y)) \vee \exists y\ Loves(y,x))$$

Transformer en forme normale conjonctive (2)

- Renommer des variables : chaque quantificateur une variable différente

$$\forall x(\exists y(Animal(y) \wedge \neg Loves(x,y)) \neg\exists z\ Loves(z,x))$$

- Skolemisation : forme plus générale de l'instanciation existentielle. Chaque variable existentielle est remplacée par une *fonction de Skolem* des variables quantifiées universellement :

$$\forall x((Animal(F(x)) \wedge \neg Loves(x,F(x))) \vee Loves(G(x),x))$$

- Supprimer les quantificateurs universels :

$$(Animal(F(x)) \wedge \neg Loves(x,F(x))) \vee Loves(G(x),x)$$

- Distribuer $\wedge$ sur $\vee$:

$$(Animal(F(x)) \vee Loves(G(x),x)) \wedge (\neg Loves(x,F(x)) \vee Loves(G(x),x))$$

Exemple de résolution