Systèmes à base de règles en logique des propositions

Négation par défaut

ML Mugnier

Deux sortes de négation

Négation classique (¬)

¬B → C : « si on sait que B est faux alors C est vrai »

Pour appliquer cette règle, il **faut** avoir le fait **¬B**

Négation par défaut, par l'échec, du monde clos **(not)**

not B → C : « si rien n'indique que B est vrai alors C est vrai »

Pour appliquer cette règle, il **ne faut pas** avoir le fait **B**

Règles conjonctives avec négation par défaut

- **Fait** : atome (= symbole, littéral positif)
- **Règle** : <conjonction de littéraux> → <atome>

$A \wedge B \wedge \text{not } C \wedge \text{not } D \rightarrow E$ *aussi notée :* A, B, not C, not D → E

De façon abstraite, une règle sera aussi notée **H+, H- → C** où :

H+ est l'ensemble des littéraux positifs

H- est l'ensemble des **symboles** des littéraux négatifs

C est un littéral positif

Sur la règle de l'exemple : H+ = {A, B} et H- = {C, D}

Application d'une règle en chaînage avant

- Soit BF une base de faits et une règle R : H+, H- → C
- R est **bloquée** sur BF si $H- \cap BF \neq \emptyset$.
- R est **applicable** sur BF si
  - *(1)* elle n'est pas bloquée, et
  - *(2)* la règle positive H+ → C est applicable (autrement dit $H+ \subseteq BF$)

R : $A \wedge B \wedge \text{not } C \wedge \text{not } D \rightarrow E$

H+ = {A, B} et H- = {C, D}

BF = {A,C}	R est bloquée
BF = {A}	R n'est pas applicable car $A \wedge B \rightarrow E$ n'est pas applicable
BF = {A,B,E}	R est applicable mais son application n'est pas utile

La négation par défaut rend l'inférence non monotone

BF = {A}

R_1 : A, not B $\longrightarrow$ C

On perd la monotonie de l'inférence :
de {A, R_1} on infère C
mais de {A, B, R_1} on n'infère plus C

Selon l'**ordre d'application des règles,** on peut obtenir une base de faits saturée différente

BF = {A}

R_1 : A, not B $\longrightarrow$ C

R_2 : A $\longrightarrow$ B

Si R_1 est appliquée avant R_2 : BF* = {A,C,B}

Si R_2 est appliquée d'abord,
R_1 n'est plus applicable : BF* = {A,B}

Problème : Quelle est la sémantique d'une base de connaissances ?

Reprenons l'exemple précédent :

BF = {A}

R_1 : A, not B $\longrightarrow$ C

R_2 : A $\longrightarrow$ B

BF* = {A,B,C} ou BF* = {A,B} ?

Appliquer R_1 avant R_2 est intuitivement choquant :

on infère C car « rien n'indique que B est vrai »,
puis on s'aperçoit que B est vrai

On va imposer que **toute règle appliquée** à un moment donné **reste applicable** par la suite (en particulier sur la base de faits saturée)

Dérivation persistante, Dérivation complète

- **Dérivation** : suite d'applications de règles à partir d'une base de faits

  $BF = BF_0\ R_1\ BF_1\ R_2\ BF_2 \ldots R_i\ BF_i$

  On dit que $\mathcal{D} = (R_1, \ldots, R_i)$ est une dérivation de BF à BF_i (le résultat)

- D est persistante si aucune règle de $\mathcal{D}$ n'est bloquée sur BF^i :
  pour toute règle H+, H- → C de $\mathcal{D}$, on a $H^- \cap BF^i = \emptyset$

- D est complète si aucune règle n'est applicable de façon utile sur BF^i :
  pour toute règle H+, H- → C de $\mathcal{D}$, on a $C \in BF^i$

$BF = \{A\}$

$R_1 : A, \text{not } B \longrightarrow C$

$R_2 : A \longrightarrow B$

(R_1,R_2) menant à $BF_2 = \{A,C,B\}$: complète, pas persistante,

(R_2) menant à $BF_1 = \{A,B\}$: persistante et complète

(R_2,R_1) : n'est pas une dérivation (car R_1 pas appliquée)

Problème : Quelle est la sémantique d'une base de connaissances ? (Suite)

Base de faits saturée : résultat d'une dérivation persistante et complète

Mais …

2 dérivations persistantes et complètes peuvent mener à des résultats différents !

BF = {A}
R_1 : not B $\longrightarrow$ C
R_2 : not C $\longrightarrow$ B

BF* = {A,C} ou BF* = {A,B}

Deux approches

1) On impose des conditions sur les ensembles de règles pour avoir un **résultat unique** quelle que soit la dérivation persistante et complète

   C'est cohérent avec l'hypothèse du monde clos
   Exemple : Datalog avec négation (dit stratifié)

2) On admet qu'il y ait **plusieurs** bases de faits saturées

   C'est cohérent avec l'hypothèse du monde ouvert

   Exemple : Answer Set Programming (ASP)
   Une base de faits saturée est appelée « answer » ou « modèle stable »

Assurer l'unicité de la base de faits saturée

A, not B → C
C, not B → D
D, E → H

Ici, partant d'une BF, peut-on avoir plusieurs BF*
(par des dérivations persistantes et complètes) ?

Non

Ensemble de règles semi-positif :
Les symboles qui apparaissent en conclusion de règles ne peuvent pas être niés

Soit la règle R : H+, H- → C

- Si H- ∩ BF ≠ ø : R est bloquée
- Sinon, on peut oublier H- (R ne sera jamais bloquée) et se ramener à une règle positive H+ → C

Assurer l'unicité de la base de faits saturée (suite)

- **Ensemble de règles stratifié :** les règles sont partitionnées en un ensemble totalement ordonné de strates :
  - Chaque strate est un ensemble semi-positif
  - Si une règle de la strate i contient *not A* en hypothèse, alors toute règle qui a *A* en conclusion est dans une strate $j < i$
  - Si une règle de la strate i contient *A* en hypothèse, alors toute règle qui a *A* en conclusion est dans une strate $j \leq i$

La saturation est effectuée par ordre croissant des strates :

- À l'étape 1, on sature la base de faits initiale avec les règles de la strate 1
- À une étape $i > 1$, on sature la base de faits calculée à l'étape i-1 avec les règles de la strate i

Bonnes propriétés des ensembles de règles stratifiables

Propriété 1 : Toute dérivation qui suit une stratification est persistante

Propriété 2 : Si un ensemble de règles est **stratifiable**, alors **toutes** ses stratifications sont équivalentes : à partir d'une base de faits BF quelconque, toutes les dérivations qui suivent une stratification produisent la **même** base de faits saturée BF*

Propriété 3 : Si un ensemble de règles est **stratifiable**, alors quelle que soit la base de faits BF, la saturation de BF par n'importe quelle **dérivation persistante et complète** produit le **même résultat**

Outil utile : Graphe de précédence des symboles

- Sommets : symboles des conclusions des règles
- Arc (p,q) si p apparaît en hypothèse d'une règle H+,H- → q
  - Arc positif si $p \in H+$
  - Arc négatif si $p \in H-$

<u>Propriété 4</u> : Un ensemble de règles est **stratifiable** si et seulement si son graphe de précédence n'admet **aucun circuit avec un arc négatif**

Calcul d'une stratification

Si le graphe de précédence des symboles n'a aucun circuit avec un arc négatif :

1. On calcule ses composantes fortement connexes *(cfc)*
   (elles ne contiennent que des circuits positifs)
2. On calcule le graphe des cfc :
   Sommets : les cfc
   Arcs : (Ci,Cj) si une règle de Ci a un arc vers une règle de Cj
   Ce graphe définit un ordre partiel sur les cfc
3. On range les cfc en strates de façon compatible avec cet ordre partiel :
   si arc (Ci,Cj) positif, alors strate(Ci) ≤ strate(Cj)
   si arc (Ci,Cj) négatif, alors strate(Ci) < strate(Cj)
4. On affecte à chaque règle la strate de son symbole de conclusion

Answet Set Programming (voir TP)

- Tous les ensembles de règles sont admis
- On peut avoir une ou plusieurs saturations (answers, modèles stables)

- En plus des règles à conclusion positive, on peut avoir des contraintes négatives :

  $H \rightarrow \bot$ (où $\bot$ est le symbole « toujours faux »)

- Dans ce cas, un modèle stable doit aussi satisfaire chaque contrainte $H \rightarrow \bot$, c'est-à-dire ne doit pas satisfaire H

- Une base de connaissances peut n'avoir aucun modèle stable : on dit qu'elle est insatisfiable

$\text{not } p \rightarrow p$

p
$p \rightarrow q$
$p \land q \rightarrow \bot$

Exemple (ASP avec Clingo)

```
omnivore(chaperon).
mange(chaperon, chevre).

plante(Y) :- omnivore(X), mange(X,Y).
animal(Y) :- omnivore(X), mange(X,Y).
:- plante(X), animal(X).
```

insatisfiable

```
omnivore(chaperon).
mange(chaperon, chevre).

plante(Y) :- omnivore(X), mange(X,Y), not animal(Y).
animal(Y) :- omnivore(X), mange(X,Y), not plante(Y).

:- plante(X), animal(X). % ne sert à rien ici
```

2 modèles stables

CONCLUSION

Approche 1 : assurer l'unicité de la base de faits saturée

(par une dérivation persistante et complète)

- stratification de l'ensemble des règles
- mais tous les ensembles de règles ne sont pas stratifiables

Approche 2 : admettre plusieurs bases de faits saturées

(par des dérivations persistantes et complètes)

- processus de saturation plus complexe
- selon le problème modélisé :
  - chaque base de faits saturée est une solution
  - ou on considère l'intersection des bases de faits saturées