

Hadrien NEGROS
Swan ROCHER

Outils d'Analyse d'une Base de Règles

Table des matières

| | | |
|----------|--|-----------|
| 1 | Introduction | 2 |
| 2 | Problématique | 3 |
| 3 | Etat de l’art | 4 |
| 4 | Définitions | 5 |
| 4.1 | Prédictat | 5 |
| 4.2 | Atome | 5 |
| 4.3 | Règle | 5 |
| 4.4 | Représentation des Conjonctions d’Atomes | 6 |
| 4.5 | Implémentation du Graphe | 6 |
| 5 | Graphe de dépendances des règles | 7 |
| 5.1 | Définition | 7 |
| 5.2 | Unification de règles | 7 |
| 5.3 | Composantes fortement connexes | 12 |
| 6 | Classes de règles | 13 |
| 7 | Outils utilisés | 14 |
| 8 | Manuel | 15 |
| 9 | Conclusion | 16 |

Chapitre 1

Introduction

Requête / base de connaissance (vision globale).

Universelles / Existentielles.

Non décidabilité de la réponse à une requête dans une base contenant des existentielles.

Exemples avec des MOTS.

Chapitre 2

Problématique

Malgré le fait que de manière général, il n'existe aucun algorithme permettant de répondre à ce problème, certaines règles peuvent entrer dans des catégories (qui seront nommées *classes de règles*) qui en ajoutant des contraintes sur la forme des règles s'assurent que le problème soit décidable. Selon quelles contraintes sont satisfaites, il est nécessaire d'appliquer différentes méthodes de réponse sur différents sous-ensembles des règles.

L'objectif de ce TER est donc d'implémenter un outil permettant d'analyser une base de règles afin de construire son graphe de dépendances associés, de déterminer quelles contraintes sont satisfaites, sur quel sous-ensemble, et si la base est décidable d'en déduire quels algorithmes utiliser sur chacun d'eux. De plus, cet outil devra pouvoir charger des bases à partir de fichiers, ainsi que les y écrire, et être suffisamment modulable pour permettre l'ajout de nouvelles vérifications de contrainte.

Chapitre 3

Etat de l'art

Chapitre 4

Définitions

4.1 Prédicat

- Un prédicat noté $p \setminus n$ est un symbole relationnel d'arité n .
- On suppose que tout nom de prédicat est unique.
- On note p_i la i^{eme} position de p .
- On note P_K l'ensemble des prédicats de K .

4.2 Atome

Un atome $a = p(a_1, a_2, \dots, a_n)$ associe un terme à chaque position d'un prédicat $p \setminus n$.
On note :

- a_i le terme en position i dans a . Un terme peut être une *constante* ou une *variable*.
Une variable peut être universelle (notée $\forall - var$) ou existentielle (notée $\exists - var$).
- $dom(a) = \{a_i : \forall i \in [1, n]\}$, l'ensemble des termes de a
- $var(a)$ l'ensemble des variables de a
- $cst(a)$ l'ensemble des constantes de a

Ces notations s'étendent aux *conjonctions d'atomes*.

4.3 Règle

Une règle $R = (H, C)$ est définie par :

- H : conjonction d'atomes représentant l'hypothèse de R ,
- C : conjonction d'atomes représentant la conclusion de R .

On note :

- $dom(R) = dom(H) \cup dom(C)$
- $var(R) = var(H) \cup var(C)$
- $cst(R) = cst(H) \cup cst(C)$
- $fr(R)$ l'ensemble des variables frontières de R , avec a_i est une variable frontière ssi a_i est une variable telle que $a_i \in var(H) \cap var(C)$.
- $cutp(R) = fr(R) \cup cst(R)$ l'ensemble des points de coupure de R

4.4 Représentation des Conjonctions d'Atomes

Une conjonction d'atomes $K = k_1 \wedge k_2 \wedge \dots \wedge k_n$ avec $k_i = q(t_1, t_2, \dots, t_p)$ peut être représentée par le graphe non orienté $G_K = (V_K, E_K, \omega)$ avec V_K son ensemble de sommets, E_K , son ensemble d'arêtes et ω une fonction de poids sur les arêtes construits de la manière suivante :

$$V_K = A_K \cup T_K \text{ avec } A_K = \{i : k_i \in K\} \text{ et } T_K = \{t_j \in \text{dom}(K)\}$$

$$E_K = \{(i, t_j) : \forall k_i \in K, \forall t_j \in \text{dom}(k_i)\}$$

$$\omega(i, t_j) = j : \forall (i, t_j) \in E_K$$

On remarque que quelques propriétés sont directement induites par cette représentation :

- G_K admet une bipartition de ses sommets, en effet toutes les arêtes ont une extrémité dans A_K et l'autre dans T_K , or par construction $A_K \cap T_K = \emptyset$.

4.5 Implémentation du Graphe

Par la suite on supposera que le graphe G_K est implémenté par deux ensembles de sommets (les atomes et les termes), ainsi que par une liste de voisinage pour chaque sommet. De plus, chaque sommet peut avoir connaissance de son type de contenu parmi : *atome*, *terme*, *constante*, $\forall - \text{var}$, $\exists - \text{var}$ en temps constant.

On note n (resp. t) le nombre d'atomes (resp. de termes) dans K .

Opérations (et temps d'exécution associé) :

- Parcourir les sommets *atomes* : n .
- Parcourir tous les sommets : $n + t$.
- Accéder au i^{eme} voisin d'un sommet : *constant*.

Chapitre 5

Graphe de dépendances des règles

Le graphe de dépendances de règles est une représentation d'une base de règles très intéressante. En effet il permet de regarder rapidement quelles règles pourront éventuellement être déclenchées après l'application d'une règle donnée.

De plus il permet de déterminer l'appartenance à certaines classes de règles, et le calcul de ses composantes permet de "découper" la base de manière à effectuer les requêtes de manières différentes selon celles-ci.

5.1 Définition

Le graphe de dépendances des règles associé à une base de règles B_R est défini comme le graphe orienté $GRD = (V_{GRD}, E_{GRD})$ avec :

- $V_{GRD} = \{R_i \in B_R\}$,
- $E_{GRD} = \{(R_i, R_j) : \exists \text{ un bon unificateur } \mu : \mu(C_i) = \mu(H_j)\}$.

Intuitivement, on crée un sommet par règle et on relie R_i à R_j si R_i "peut amener à déclencher" R_j (R_j dépend de R_i). La notion d'unificateur est abordée dans la section suivante.

5.2 Unification de règles

Afin de pouvoir construire ce graphe, il faut donc pouvoir déterminer si une règle peut en déclencher une autre, c'est à dire s'il existe un unificateur entre la conclusion de la première et l'hypothèse de la seconde. Tout d'abord, l'unification est définie, s'en suit un algorithme permettant de vérifier si un tel unificateur existe, puis la correction de celui-ci ainsi que ses complexités.

5.2.1 Définitions

Substitution

Une substitution de taille n d'un ensemble de symboles X dans un ensemble de symboles Y est une fonction de X vers Y représentée par l'ensemble de couples suivants (avec $n \leq |X|$) :

- $s = \{(x_i, y_i) : \forall i \in [1, n] \ x_i \in X, y_i \in Y, \forall j \neq i \ x_i \neq x_j\}$
- $s(x_i) = y_i \ \forall i \in [1, n]$
- $s(x_i) = x_i \ \forall i \in [n+1, |X|] \ x_i \in X$

Unificateur logique

Un unificateur logique entre deux atomes a_1 et a_2 est une substitution μ telle que :

- $\mu : \text{var}(a_1) \cup \text{var}(a_2) \rightarrow \text{dom}(a_1) \cup \text{dom}(a_2)$
- $\mu(a_1) = \mu(a_2)$

Cette définition s'étend aux conjonctions d'atomes.

Unificateur de conclusion atomique

Un unificateur de conclusion atomique est un unificateur logique $\mu = \{(x_i, t_i) : \forall i \in [1, n]\}$ entre l'hypothèse d'une règle $R_1 = (H_1, C_1)$ et la conclusion atomique d'une règle $R_2 = (H_2, C_2)$ et est défini de la manière suivante :

- $\mu : \text{fr}(R_2) \cup \text{var}(H_1) \rightarrow \text{dom}(C_2) \cup \text{cst}(H_1)$
- $\forall (x_i, t_i) \in \mu \text{ si } x_i \in \text{fr}(R_2) \text{ alors } t_i \in \text{cutp}(R_2) \cup \text{cst}(H_1)$

Bonne unification atomique

Un bon unificateur de conclusion atomique est un unificateur de conclusion atomique $\mu = \{(x_i, t_i) : \forall i \in [1, n]\}$ entre un sous ensemble Q de l'hypothèse d'une règle $R_1 = (H_1, C_1)$ et la conclusion d'une règle atomique $R_2 = (H_2, C_2)$ tel que :

$\forall (x_i, t_i) \in \mu : \text{si } x_i \in H_1 \setminus Q \text{ alors } t_i \text{ n'est pas une } \exists - \text{var}$

Un tel ensemble Q est appelé un *bon ensemble d'unification atomique* de l'hypothèse de R_1 par la conclusion de R_2 . On note que Q est donc défini comme suit :

- $Q \subseteq H_1$
- $\forall \text{ position } i \text{ de } \exists - \text{var dans } C_2, \forall \text{ atome } a \in Q, \text{ si } a_i \in \text{var}(H_1) \text{ alors } \forall \text{ atome } b \in H_1 : \text{si } \exists b_j \in b : a_i = b_j, \text{ alors } b \in Q$

Bon ensemble d'unification atomique minimal

Un *bon ensemble d'unification atomique minimal* Q de H_1 par C_2 enraciné en a est défini tel que :

- Q est un bon ensemble d'unification atomique de H_1 par C_2
- $a \in Q$
- $|Q| = \min(|Q_i| : Q_i \text{ est un bon ensemble d'unification atomique de } H_1 \text{ par } C_2 \text{ et } a \in Q_i)$

5.2.2 Algorithmes

Vérifier qu'une règle atomique R_i peut déclencher R_j consiste donc à trouver un bon unificateur atomique entre R_i et R_j . Dans cette section, un algorithme permettant de répondre à ce problème est détaillé.

Dans la suite, les règles sont supposées représentées par des graphes (tels que définis en 4.4) et à conclusion atomique.

Le premier algorithme fait appel aux deux suivants de manière à déterminer si il existe au moins un unificateur entre les deux conjonctions d'atomes. En première phase, il vérifie l'existence d'unificateurs avec chaque atome de manière indépendante. S'ensuit une extension à partir des atomes préselectionnés, et dès qu'un bon ensemble d'unification est entièrement unifié, l'algorithme s'arrête en répondant avec succès.

Algorithm 1 Unification

Require: H_1 : conjonction d'atomes, $R = (H_2, C_2)$: règle à conclusion atomique

Ensure: succès si C_2 peut s'unifier avec H_1 , i.e. si $\exists H \subseteq H_1, \mu$ une substitution : $\mu(H_1) = \mu(C_2)$, échec sinon

```

1  ▷ Précoloration
2  for all sommet atome  $a \in H_1$  do
3      if  $UnificationLocale(a, R) \neq \text{échec}$  then
4           $couleurLogique[a] \leftarrow \text{noir}$ 
5      else
6           $couleurLogique[a] \leftarrow \text{blanc}$ 
7      end if
8  end for
9  ▷ Initialisation du tableau contenant les positions des variables existentielles de  $C_2$ 
10  $E \leftarrow \{i : c_i \text{ est une } \exists - \text{var de } C_2\}$ 
11 ▷ Extension des ensembles
12 for all sommet atome  $a \in H_1 : couleurLogique[a] = \text{noir}$  do
13     if  $Q \leftarrow Extension(H_1, a, couleurLogique, E) \neq \text{échec}$  then
14         if  $UnificationLocale(Q, R) \neq \text{échec}$  then
15             return succès
16         end if
17     end if
18      $couleurLogique[a] \leftarrow \text{blanc}$ 
19 end for
20 return échec

```

Le deuxième algorithme est utilisé pour le calcul des bons ensembles d'unification à partir d'un atome racine. Tant qu'aucune erreur n'est détectée il "avale" les atomes voisins aux termes en position "existentielles".

Algorithm 2 Extension

Require: H_1 : conjonction d'atomes, $a \in H_1$: sommet atome racine, $couleurLogique$: tableau de taille égal au nombre d'atomes dans H_1 tel que $couleurLogique[a] = noir$ ssi $UnificationLocale(a, R) = succès$, E : ensemble des positions des variables existentielles

Ensure: Q : bon ensemble d'unification minimal des atomes de H_1 construit à partir de a s'il existe, échec sinon.

```
1  ▷ Initialisation du parcours
2  for all sommet atome  $a \in H_1$  do
3      if  $couleurLogique[a] = noir$  then
4          for all sommet terme  $t \in voisins(a)$  do
5               $couleur[t] = blanc$ 
6          end for
7           $couleur[a] = blanc$ 
8      end if
9  end for
10  $couleur[a] \leftarrow noir$ 
11  $Q \leftarrow \{a\}$  ▷ conjonction d'atomes à traiter
12  $attente \leftarrow \{a\}$  ▷ file d'attente du parcours
13 while  $attente \neq \emptyset$  do
14      $u \leftarrow haut(attente)$ 
15     if  $u$  est un atome then
16         for all  $i \in E$  do
17              $v \leftarrow voisin(u, i)$ 
18             if  $v$  est une constante then
19                 return échec
20             else if  $couleur[v] = blanc$  then
21                 ▷  $v$  est une  $\forall$  - var non marquée par le parcours
22                  $couleur[v] \leftarrow noir$ 
23                  $attente \leftarrow attente \cup \{v\}$ 
24             end if
25         end for
26     else
27         ▷  $u$  est un terme
28         for all  $v \in voisins(u)$  do
29             if  $couleurLogique[v] = blanc$  then
30                 return échec
31             else
32                 if  $couleur[v] = blanc$  then
33                      $couleur[v] \leftarrow noir$ 
34                      $attente \leftarrow attente \cup \{v\}$ 
35                      $Q \leftarrow Q \cup \{v\}$ 
36                 end if
37             end if
38         end for
39     end if
40 end while ▷ Fin du parcours
41 return  $Q$ 
```

Remarque :

La phase d'initialisation du parcours pourrait simplement parcourir tous les sommets de H_1 et mettre leur couleur à blanc. En pratique, cette solution serait sans doute plus efficace, mais dépendrait donc du nombre de sommets total dans H_1 . Ce qui en théorie amènerait la complexité de cette boucle en $\mathcal{O}(\text{nombre d'atomes} \times \text{arite max de } H_1)$. Or ici, la complexité ne dépend pas de cette arité max, mais uniquement de l'arité du prédicat de la conclusion C_2 .

Le dernier algorithme est celui qui teste réellement si il existe un unificateur entre une conclusion atomique, et un bon ensemble d'unification atomique minimal. Il est appelé une première fois pour tester les atomes de la conjonction séparément, et permettre une préselection des atomes (qui vont servir de racine). Durant la dernière phase (lorsqu'il est appelé sur les ensembles étendus), s'il trouve un unificateur, celui-ci assure que la règle peut amener à déclencher la conjonction.

Algorithm 3 UnificationLocale

Require: H_1 : conjonction d'atomes, $R = (H_2, C_2)$: règle à conclusion atomique
Ensure: succès si C_2 peut s'unifier avec H_1

```

1  ▷ Vérification des prédicats
2  for all atome  $a \in H_1$  do
3      if  $\text{prédicat}(a) \neq \text{prédicat}(C_2)$  then
4          return échec
5      end if
6  end for
7   $u \leftarrow \emptyset$   ▷ substitution
8  for all terme  $t_i \in C_2$  do
9      ▷ def :  $a_i$  = terme de  $a$  en position  $i$ 
10      $E \leftarrow \{a_i : \forall \text{ atome } a \in H_1\}$ 
11     if  $t_i$  est une constante then
12         if  $\exists v \in E : v$  est une constante et  $v \neq t_i$ , ou  $v$  est une  $\exists$  – variable then
13             return échec
14         else
15              $u \leftarrow \{(v, t_i) : v \in E \text{ et } v \neq t_i\}$ 
16         end if
17     else if  $t_i$  est une  $\exists$  – variable then
18         if  $\exists v \in E : v$  est une  $\exists$  – variable et  $v \neq t_i$ , ou  $v$  est une constante then
19             return échec
20         else
21              $u \leftarrow \{(v, t_i) : v \in E \text{ et } v \neq t_i\}$ 
22         end if
23     else
24         if  $\exists v_1, v_2 \in E : v_1 \neq v_2$  et  $v_1, v_2$  ne sont pas des  $\forall$  – variables then
25             return échec
26         else if  $\exists c \in E : c$  est une constante then
27              $u \leftarrow \{(v, c) : v \in E \cup \{t_i\} \text{ et } v \neq c\}$ 
28         else
29              $u \leftarrow \{(v, t_i) : v \in E \text{ et } v \neq t_i\}$ 
30         end if
31     end if
32      $H_1 \leftarrow u(H_1)$ 
33      $C_2 \leftarrow u(C_2)$ 
34 end for
35 return succès
  
```

5.2.3 Correction

5.2.4 Complexités

On note :

- n = nombre d'atomes dans H_1
- p = arité de C_2

- t = nombre de termes "colorables" dans H_1
 - m = nombre d'arêtes "suivables" dans H_1
- On remarque que dans le pire des cas on a :
- $t = n \times p$
 - $m = n \times p$

Temps

- UnificationLocale (pire des cas) = $\mathcal{O}(np)$
- Extension (pire des cas) = $\mathcal{O}(np)$
- Unification (pire des cas) = $\mathcal{O}(n^2p)$

Extension

On effectue simplement un parcours en largeur à partir d'un sommet donné qui peut éventuellement s'arrêter plus tôt qu'un parcours classique. La complexité en temps dans le pire des cas est donc au plus la même, c'est à dire linéaire au nombre de sommets plus le nombre d'arcs. Le graphe représentant la conjonction d'atomes H_1 possèdent $n \times ariteMax(H_1)$ arêtes et $n \times (ariteMax(H_1) + 1)$ sommets. On sait donc que $C_{Extension}^{temps} = \mathcal{O}(n \times ariteMax(H_1))$.

Deux cas :

i) Découverte d'un sommet atome :

parcours classique = \hookrightarrow on récupère tous les voisins blancs

parcours extension = \hookrightarrow on récupère tous les voisins blancs de couleur logique noire (en effet arrêt immédiat si un voisin de couleur logique noire est découvert.

ii) Découverte d'un sommet variable :

pas de différence

On a donc que l'algo ne suit que les sommets atomes pouvant être unifiés localement. C'est à dire (entre autres) que leur prédicat est égal à $predicat(C_2)$.

...

Espace

5.3 Composantes fortement connexes

Chapitre 6

Classes de règles

Chapitre 7

Outils utilisés

Chapitre 8

Manuel

Chapitre 9

Conclusion