

Outils d'Analyse d'une Base de Règles

Swan Rocher

Université Montpellier 2

13 mai 2012

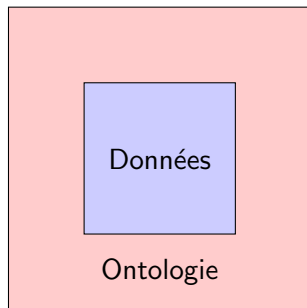
Table des matières

Table des matières

Introduction

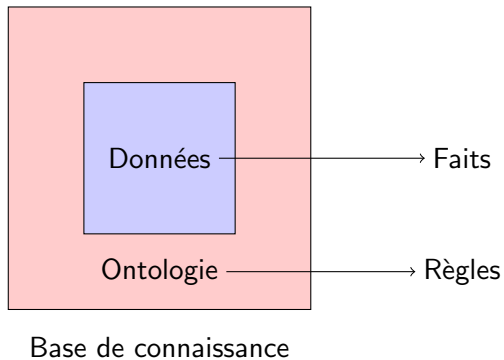
- Bases de connaissance très utilisées
- Composées d'*informations* et d'un *raisonnement* sur celles-ci
- Obtenir une réponse à une requête

Base de connaissance



Base de connaissance

Base de connaissance



Exemples

Exemple de requête sur les données uniquement

Données :

- "Jean est un des *parents* de Tom"
- "Jean est un *homme*"

Requête : "Jean est-il le père de Tom ?"

Réponse : NON.

Exemples

Exemple de requête sur les données uniquement

Données :

- "*Jean est un des parents de Tom*"
- "*Jean est un homme*"

Requête : "Jean est-il le père de Tom ?"

Réponse : NON.

Et en tenant compte d'une ontologie

Ontologie :

- "**Si** un *homme* est le *parent* de *quelqu'un*, **alors** *il* est son *père*."

Requête : "Jean est-il le père de Tom ?"

Réponse : OUI.

Réponse à une requête

- Difficulté dépendant de l'ontologie
- Possibilité de créer de nouveaux individus
- Problème non décidable de manière générale
- Nécessaire de déterminer des classes de règles

Exemple

Donnée :

- " *Jean est un homme.*"

Ontologie :

- " *Tout homme a un père.*"

Déductions :

- "Jean a un père"
- "Celui-ci a un père"
- "Ce-dernier a également un père"
- ...

Problématique

- Développement d'un outil en Java analysant une base de règles
- Construction du graphe de dépendances associé
- Détermination des classes de règles
- Etude de la décidabilité de la base
- Lecture et écriture de la base à partir et vers un fichier

Table des matières

Atome

- Prédicat : symbole relationnel d'arité donnée
- Atome : prédicat et termes associés à ses positions
- Terme : variable ou constante (pas de fonction)

Exemple

"*x* est le père de *Tom*."

père(*x*,*Tom*)

- prédicat : père
- variable *x* en position 1
- constante *Tom* en position 2

Faits

- Conjonction de k atomes
- Existentiellement fermée
- $A = \exists(atome_1 \wedge atome_2 \wedge \dots \wedge atome_k)$

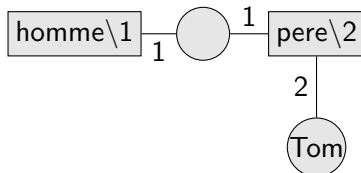
Exemple

"Il existe un homme qui est le père de Tom."
 $\exists x (\text{homme}(x) \wedge \text{pere}(x, \text{Tom}))$

Représentation graphique d'une conjonction d'atomes

- un sommet par atome étiqueté par son prédicat
- un sommet par terme étiqueté si constante
- une arête pour chaque apparition de terme dans un atome dont le poids est la position du terme

$\text{homme}(x) \wedge \text{pere}(x, \text{Tom})$



Règle

- Deux conjonctions d'atomes : une hypothèse H et une conclusion C
- $H \rightarrow C$
- Variable x soit universelle ($x \in H$) ou existentielle ($x \notin H$)
- Frontière : variable à la fois dans H et dans C ($x \in H \cap C$)

Règle

- Deux conjonctions d'atomes : une hypothèse H et une conclusion C
- $H \rightarrow C$
- Variable x soit universelle ($x \in H$) ou existentielle ($x \notin H$)
- Frontière : variable à la fois dans H et dans C ($x \in H \cap C$)

Exemple


"Tout homme a un père qui est un homme."

$\forall x (\text{homme}(x) \rightarrow \exists z (\text{homme}(z) \wedge \text{père}(z,x)))$

- Hypothèse : $\forall x (\text{homme}(x))$
- Conclusion : $\forall x \exists z (\text{homme}(z) \wedge \text{père}(z,x))$

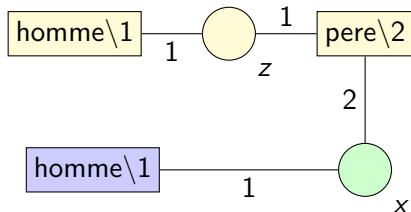
Représentation graphique d'une règle

$$\forall x (\text{homme}(x) \rightarrow \exists z (\text{homme}(z) \wedge \text{pere}(z,x)))$$

 hypothèse

 conclusion

 frontière



Requête

- Conjonction d'atomes existentiellement fermée
- Plusieurs méthodes de réponses

Exemple

"Tom a-t'il un père?"

$\exists x \text{ père}(x, \text{Tom})$

Chaînage avant

- Génération de nouveaux faits à partir des précédents et de l'ontologie
- A chaque création de fait, vérification de l'existence d'une réponse à la requête dans F
- Arrêt lorsque tous les faits sont générés ou si réponse positive

Chaînage avant

- Génération de nouveaux faits à partir des précédents et de l'ontologie
- A chaque création de fait, vérification de l'existence d'une réponse à la requête dans F
- Arrêt lorsque tous les faits sont générés ou si réponse positive

Exemple

Base :

- $F = \text{parent}(\text{Jean}, \text{Tom}) \wedge \text{homme}(\text{Jean})$
- $R = \forall x \forall y (\text{homme}(x) \wedge \text{parent}(x, y) \rightarrow \text{père}(x, y))$

Requête : $\text{père}(\text{Jean}, \text{Tom})$

Chaînage avant

- Génération de nouveaux faits à partir des précédents et de l'ontologie
- A chaque création de fait, vérification de l'existence d'une réponse à la requête dans F
- Arrêt lorsque tous les faits sont générés ou si réponse positive

Exemple

Base :

- $F = \text{parent}(\text{Jean}, \text{Tom}) \wedge \text{homme}(\text{Jean})$
- $R = \forall x \forall y (\text{homme}(x) \wedge \text{parent}(x, y) \rightarrow \text{père}(x, y))$

Requête : $\text{père}(\text{Jean}, \text{Tom})$

- Application de l'unique règle
- $x \leftarrow \text{Jean}, y \leftarrow \text{Tom}$
- $F = \text{parent}(\text{Jean}, \text{Tom}) \wedge \text{homme}(\text{Jean}) \wedge \text{père}(\text{Jean}, \text{Tom})$
- $Q \in F \rightarrow \text{réponse positive !}$

Chaînage arrière

- Réécriture de la requête via l'ontologie
- A chaque réécriture, vérification de l'existence d'une réponse à l'une de celles-ci dans F
- Arrêt lorsque celle-ci ne peut plus être réécrire ou si réponse positive

Chaînage arrière

- Réécriture de la requête via l'ontologie
- A chaque réécriture, vérification de l'existence d'une réponse à l'une de celles-ci dans F
- Arrêt lorsque celle-ci ne peut plus être réécrire ou si réponse positive

Exemple

Base :

- $F = \text{parent}(\text{Jean}, \text{Tom}) \wedge \text{homme}(\text{Jean})$
- $R = \forall x \forall y (\text{homme}(x) \wedge \text{parent}(x, y) \rightarrow \text{père}(x, y))$

Requête : $Q = \text{père}(\text{Jean}, \text{Tom})$

Chaînage arrière

- Réécriture de la requête via l'ontologie
- A chaque réécriture, vérification de l'existence d'une réponse à l'une de celles-ci dans F
- Arrêt lorsque celle-ci ne peut plus être réécrire ou si réponse positive

Exemple

Base :

- $F = \text{parent}(\text{Jean}, \text{Tom}) \wedge \text{homme}(\text{Jean})$
- $R = \forall x \forall y (\text{homme}(x) \wedge \text{parent}(x, y) \rightarrow \text{père}(x, y))$

Requête : $Q = \text{père}(\text{Jean}, \text{Tom})$

- Réécriture via l'unique règle
- $\text{Jean} \rightarrow x, \text{Tom} \rightarrow y$
- $Q' = \{ Q_0 = \text{père}(\text{Jean}, \text{Tom}), \text{ } Q_1 = \text{parent}(\text{Jean}, \text{Tom}) \wedge \text{homme}(\text{Jean}) \}$
- $Q_1 \in F \rightarrow$ réponse positive !

Table des matières

Dépendance des règles

- R_i dépend de $R_j \leftrightarrow R_j$ peut amener à déclencher R_i
- Problème NP-complet (Unification de H_i avec C_j)
- Construction du graphe de dépendances associé (GRD)

Construction du graphe

Exemple

"Tout homme est un humain. Tout humain a un père qui est un homme. Si un homme est le parent d'un autre, alors il est son père. Tout père d'un individu est un de ses parents."

- $R_1 : \forall x (\text{homme}(x) \rightarrow \text{humain}(x))$
- $R_2 : \forall x (\text{humain}(x) \rightarrow \exists z (\text{homme}(z) \wedge \text{pere}(z,x)))$
- $R_3 : \forall x \forall y (\text{parent}(x,y) \wedge \text{homme}(x) \rightarrow \text{pere}(x,y))$
- $R_4 : \forall x \forall y (\text{pere}(x,y) \rightarrow \text{parent}(x,y))$

Construction du graphe

Base de règles :

- $R_1 : \forall x (\text{homme}(x) \rightarrow \text{humain}(x))$
- $R_2 : \forall x (\text{humain}(x) \rightarrow \exists z (\text{homme}(z) \wedge \text{pere}(z,x)))$
- $R_3 : \forall x \forall y (\text{parent}(x,y) \wedge \text{homme}(x) \rightarrow \text{pere}(x,y))$
- $R_4 : \forall x \forall y (\text{pere}(x,y) \rightarrow \text{parent}(x,y))$

 R_3 R_4 R_1 R_2

Construction du graphe

Base de règles :

- $R_1 : \forall x (\text{homme}(x) \rightarrow \text{humain}(x))$
- $R_2 : \forall x (\text{humain}(x) \rightarrow \exists z (\text{homme}(z) \wedge \text{pere}(z,x)))$
- $R_3 : \forall x \forall y (\text{parent}(x,y) \wedge \text{homme}(x) \rightarrow \text{pere}(x,y))$
- $R_4 : \forall x \forall y (\text{pere}(x,y) \rightarrow \text{parent}(x,y))$

R_1 peut elle se redéclencher ?

C_1 a un prédicat différent de H_1



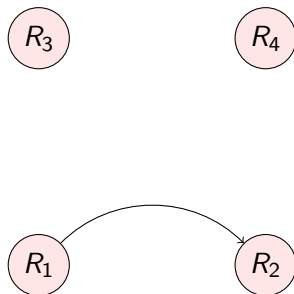
Construction du graphe

Base de règles :

- $R_1 : \forall x (\text{homme}(x) \rightarrow \text{humain}(x))$
- $R_2 : \forall x (\text{humain}(x) \rightarrow \exists z (\text{homme}(z) \wedge \text{pere}(z,x)))$
- $R_3 : \forall x \forall y (\text{parent}(x,y) \wedge \text{homme}(x) \rightarrow \text{pere}(x,y))$
- $R_4 : \forall x \forall y (\text{pere}(x,y) \rightarrow \text{parent}(x,y))$

R_1 peut amener à déclencher R_2 ?

Unification de C_1 avec R_1



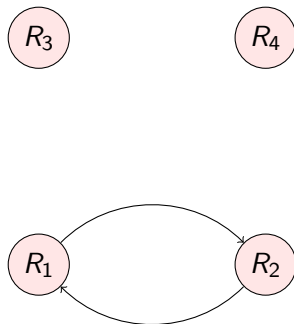
Construction du graphe

Base de règles :

- $R_1 : \forall x (\text{homme}(x) \rightarrow \text{humain}(x))$
- $R_2 : \forall x (\text{humain}(x) \rightarrow \exists z (\text{homme}(z) \wedge \text{pere}(z,x)))$
- $R_3 : \forall x \forall y (\text{parent}(x,y) \wedge \text{homme}(x) \rightarrow \text{pere}(x,y))$
- $R_4 : \forall x \forall y (\text{pere}(x,y) \rightarrow \text{parent}(x,y))$

R_2 peut amener à déclencher R_1 ?

R_2 amène l'existence d'un nouvel individu et l'hypothèse de R_1 est vérifiée pour celui ci



Construction du graphe

Base de règles :

- $R_1 : \forall x (\text{homme}(x) \rightarrow \text{humain}(x))$
- $R_2 : \forall x (\text{humain}(x) \rightarrow \exists z (\text{homme}(z) \wedge \text{pere}(z,x)))$
- $R_3 : \forall x \forall y (\text{parent}(x,y) \wedge \text{homme}(x) \rightarrow \text{pere}(x,y))$
- $R_4 : \forall x \forall y (\text{pere}(x,y) \rightarrow \text{parent}(x,y))$

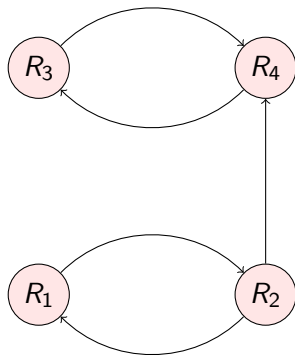


Table des matières

Etapes d'analyse

- ❶ Construire le graphe de dépendances des règles
- ❷ Vérifier si celui-ci est acyclique
- ❸ Déterminer les classes de règles
- ❹ Déterminer si l'ontologie forme un ensemble décidable
- ❺ Signaler les algorithmes à utiliser

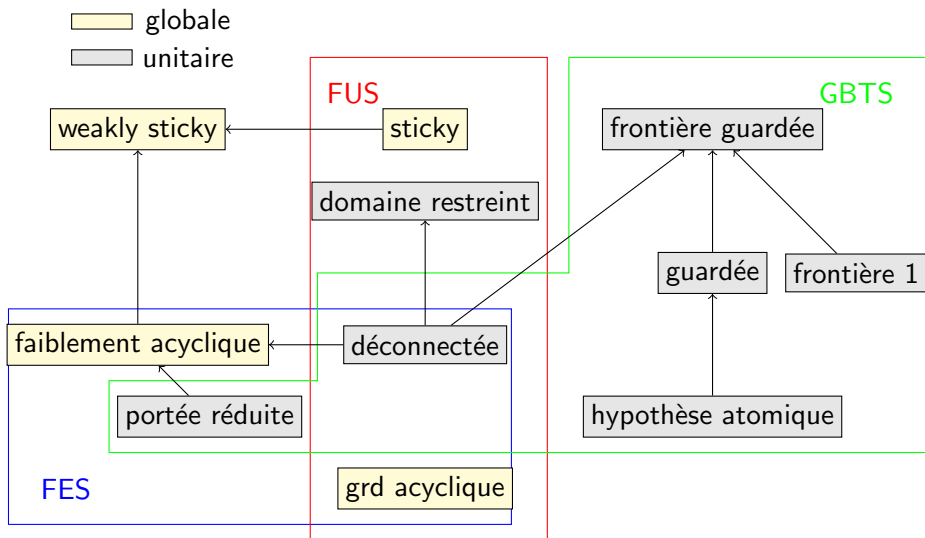
Classes de règles abstraites

- Réponse à une requête en temps fini en suivant certains algorithmes
- Finite Extension Set : algorithmes de chaînage avant
- (Greedy) Bounded Treewidth Set : algorithmes de chaînage avant avec condition d'arrêt particulière
- Finite Unification Set : algorithmes de chaînage arrière
- Impossible de vérifier si une règle appartient à une classe abstraite

Classes de règles concrètes

- Spécialisations des classes abstraites
- Application sur chacune des règles indépendamment ou sur un ensemble
- Complexités différentes
- Vérification de propriétés sur la frontière, la position des variables, ...

Classes concrètes



Portée réduite

- Aucune variable existentielle
- $variables(R) \subseteq variables(H)$
- $FES \cap GBTS$

Exemple

$R_5 : \forall x \forall y \forall z (m\hat{e}meFamille(x,y) \wedge m\hat{e}meFamille(y,z) \rightarrow m\hat{e}meFamille(x,z))$

Frontière gardée

- Un atome de l'hypothèse contient toutes les variables de la frontière
- $\exists a \in H : \text{frontière}(R) \subseteq \text{variables}(a)$
- Seule la frontière influe sur une nouvelle application d'une règle
- *GBTS*

Exemple

$\forall x \forall y (p(x) \wedge q(y) \rightarrow \exists z (r(y, z)))$

Garde-frontière : $q(y)$

Hypothèse atomique

- L'hypothèse de la règle ne contient qu'un seul atome
- Spécialisation des règles gardées
- Utiles pour les notions d'héritage
- $|H| = 1$
- $GBTS \cap FUS$

Exemple

$R_1 : \forall x \text{ (homme}(x) \rightarrow \text{humain}(x))$

Etapes d'analyse

- ① Construire le graphe de dépendances des règles
- ② Vérifier si celui-ci est acyclique
- ③ Déterminer les classes de règles
- ④ Déterminer si l'ontologie forme un ensemble décidable
- ⑤ Signaler les algorithmes à utiliser

Réponse à une requête en temps fini

- Graphe de dépendances de règles sans circuit
- L'ensemble des règles étiqueté par une même classe
- Sinon : calcul du graphe orienté des composantes fortement connexes
- Détermination des classes de chaque composante
- Vérification de la propriété de *précédence*

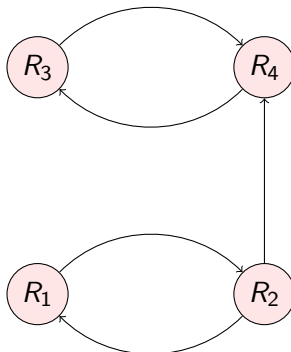
Précédence

- Graphe des composantes fortement connexes : un sommet par composante, un arc entre C_i et C_j si une règle de C_i peut déclencher une règle de C_j
- Composante C_i *précède* C_j si aucun arc de C_j vers C_i
- Notée $C_i \triangleright C_j$
- Décidable si $FES \triangleright GBTS \triangleright FUS$

Exemple

Exemple

- $R_1 : \forall x (\text{homme}(x) \rightarrow \text{humain}(x))$
- $R_2 : \forall x (\text{humain}(x) \rightarrow \exists z (\text{homme}(z) \wedge \text{pere}(z,x)))$
- $R_3 : \forall x \forall y (\text{parent}(x,y) \wedge \text{homme}(x) \rightarrow \text{pere}(x,y))$
- $R_4 : \forall x \forall y (\text{pere}(x,y) \rightarrow \text{parent}(x,y))$



Exemple

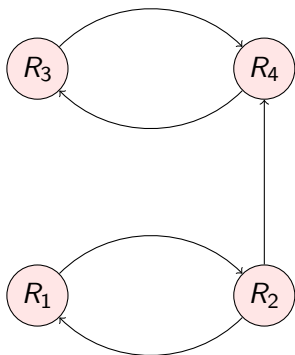


FIGURE: GRD

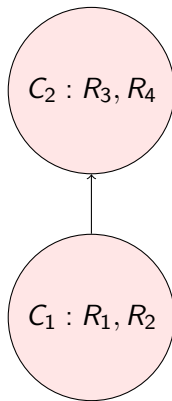


FIGURE: Composantes fortement connexes

Exemple

- $C_1 \triangleright C_2$
- Si C_1 est *FUS* uniquement, C_2 doit l'être également
- Si C_1 est *GBTS* uniquement, C_2 ne peut pas être *FES*

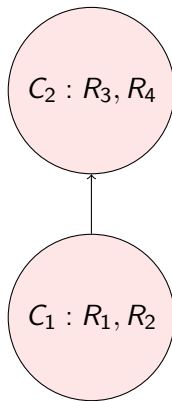


FIGURE: Composantes fortement connexes

Exemple

$C_1 :$

- $R_1 : \forall x (\text{homme}(x) \rightarrow \text{humain}(x))$
- $R_2 : \forall x (\text{humain}(x) \rightarrow \exists z (\text{homme}(z) \wedge \text{pere}(z,x)))$
- Hypothèses atomiques donc *FUS*
- Gardées donc *GBTS*

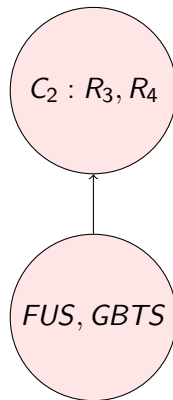


FIGURE: Composantes fortement connexes

Exemple

C_2 :

- $R_3 : \forall x \forall y (\text{parent}(x,y) \wedge \text{homme}(x) \rightarrow \text{pere}(x,y))$
- $R_4 : \forall x \forall y (\text{pere}(x,y) \rightarrow \text{parent}(x,y))$

Classes :

- Portée réduite donc *FES*
- Domaine restreint donc *FUS*

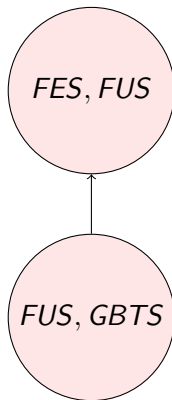


FIGURE: Composantes fortement connexes

Exemple

Algorithmes à utiliser :

- C_1 : chaînage avant ou arrière
- C_2 : chaînage arrière

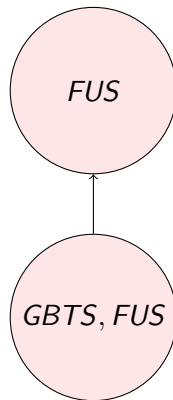


FIGURE: Composantes fortement connexes

Table des matières

Table des matières

Gestion du projet

- Gestionnaire de versions : git (github.com)
- Réunions presque hebdomadaires avec les encadrants
- Communication par courriels

Problèmes rencontrés

- Domaine nouveau
- Disparition d'un membre du groupe

Contributions

- Lecture de nombreux articles
- Développement d'une bibliothèque pour les graphes
- Conception d'un algorithme d'unification
- Construction du graphe de dépendances des règles
- Calcul des classes concrètes
- Combinaison des classes abstraites
- Etude de la décidabilité
- Lecture et écriture d'une base à partir et vers un fichier
- Sortie PostScript minimale pour la visualisation des différents graphes

Perspectives

- Reconnaissance de nouvelles classes de règles concrètes
- Combinaison des classes de règles en fonction des complexités
- Implémentation d'une interface graphique

Merci de votre attention

Avez-vous des questions ?