Outils d'Analyse d'une Base de Règles

Swan Rocher

Université Montpellier 2

2 mai 2012

Table des matières

1. Contexte
2. Notions de logique
3. Dépendance des règles
4. Classes de règles
5. Développement
6. Conclusion

Table des matières

1. Contexte
2. Notions de logique
3. Dépendance des règles
4. Classes de règles
5. Développement
6. Conclusion

Contexte

- Base de données très utilisées dans le monde de l'informatique
- Effectuer des requêtes sur ces bases
- Généralement, seuls des faits sont pris en compte

Exemple

Base :

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

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

Contexte

- Base de données très utilisées dans le monde de l'informatique
- Effectuer des requêtes sur ces bases
- Généralement, seuls des faits sont pris en compte
- Ajout d'un ensemble de *règles*

Exemple

Base :

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

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

Contexte

- Base de données très utilisées dans le monde de l'informatique
- Effectuer des requêtes sur ces bases
- Généralement, seuls des faits sont pris en compte
- Ajout d'un ensemble de *règles*
- Indécidable de manière générale
- Nécessaire d'ajouter des contraintes (classes de règles)
- Les contraintes influent sur les méthodes de réponse

Exemple

Base :

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

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

Problématique

- Développement d'un outil analysant une base de règles
- Construction du graphe de dépendances associé
- Détermination des classes de règles
- Décidabilité de la base
- Déduction des algorithmes à utiliser
- Lecture et écriture d'une base à partir et vers un fichier
- Langage Java

Table des matières

1. Contexte
2. Notions de logique
3. Dépendance des règles
4. Classes de règles
5. Développement
6. Conclusion

Table des matières

1 Contexte

2 Notions de logique
- Atomes
- Règles
- Requêtes

3 Dépendance des règles

4 Classes de règles

5 Développement

6 Conclusion

Atome

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

Exemple

"Tom a un père."
$\exists x$ (pere(x,Tom))

Conjonction d'atomes

- Composée de k atomes
- $A = atome_1 \wedge atome_2 \wedge ... \wedge atome_k$
- Représentation par un graphe non orienté

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

On crée :

- 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

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

Représentation graphique d'une conjonction d'atomes

On crée :

- 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

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

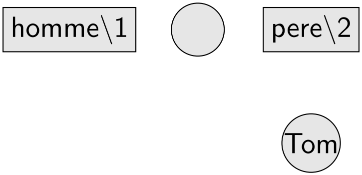
$\boxed{\text{homme}\backslash 1}$ $\boxed{\text{pere}\backslash 2}$

Représentation graphique d'une conjonction d'atomes

On crée :

- 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

$$\exists x \; (\mathrm{homme}(x) \wedge \mathrm{pere}(x, \mathrm{Tom}))$$



Représentation graphique d'une conjonction d'atomes

On crée :

- 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 ✓

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

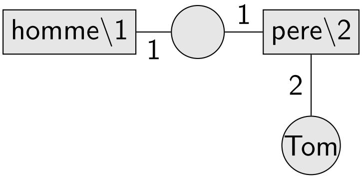


Table des matières

1. Contexte
2. Notions de logique
   - Atomes
   - Règles
   - Requêtes
3. Dépendance des règles
4. Classes de règles
5. Développement
6. Conclusion

Règle

- Deux conjonctions d'atomes : une hypothèse H et une conclusion C
- $H \to C$
- 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 règle universelle

"Tout homme est un humain."
$\forall x\ (\text{homme}(x) \rightarrow \text{humain}(x))$

Règle

- Deux conjonctions d'atomes : une hypothèse H et une conclusion C
- $H \to C$
- 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 règle universelle

"Tout homme est un humain."
$\forall x$ (homme(x) $\to$ humain(x))

Exemple règle existentielle

"Tout humain a un père qui est un homme."
$\forall x$ (humain(x) $\to \exists z$ (homme(z) $\wedge$ pere(z,Tom)))

Représentation graphique d'une règle

- Ensembles de sommets et d'arêtes identiques à une conjonction d'atomes
- 2-coloration des atomes pour différencier hypothèse et conclusion

Exemple

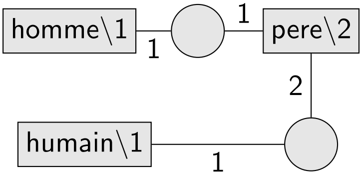
"Tout humain a un père qui est un homme."

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

Représentation graphique d'une règle

- 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 ✓
- coloration des sommets atomes en fonction de leur position

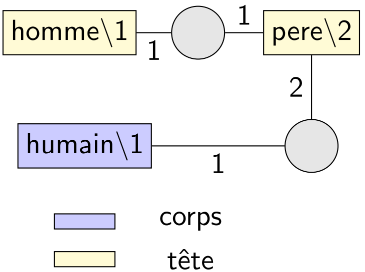
$\forall x$ (humain(x) $\rightarrow \exists z$ (homme(z) $\wedge$ pere(z,x)))



Représentation graphique d'une règle

1. un sommet par atome étiqueté par son prédicat ✓
2. un sommet par terme étiqueté si constante ✓
3. une arête pour chaque apparition de terme dans un atome dont le poids est la position du terme ✓
4. coloration des sommets atomes en fonction de leur position ✓

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



Les différents éléments d'une règle

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

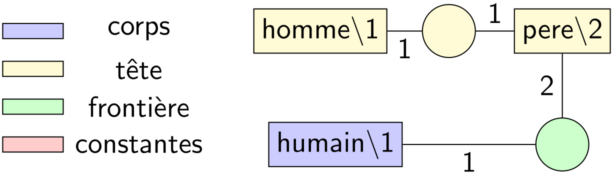


Table des matières

1 Contexte

2 Notions de logique
- Atomes
- Règles
- Requêtes

3 Dépendance des règles

4 Classes de règles

5 Développement

6 Conclusion

Base de connaissance

- Notée $K = (F, R)$
- Ensemble de faits représenté par une conjonction d'atomes F
- Ensemble de règles R

Chaînage avant

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

Exemple

Base :

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

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

Chaînage avant

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

Exemple

Base :

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

Requête : père($Jean, Tom$)

- Application de l'unique règle

Chaînage avant

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

Exemple

Base :

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

Requête : père($Jean$, Tom)

- Application de l'unique règle
- $x \leftarrow Jean, y \leftarrow Tom$

Chaînage avant

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

Exemple

Base :

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

Requête : $\text{père}(Jean, Tom)$

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

Chaînage avant

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

Exemple

Base :

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

Requête : père$(Jean, Tom)$

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

Chaînage arrière

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

Exemple

Base :

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

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

Chaînage arrière

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

Exemple

Base :

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

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

- Réécriture via l'unique règle

Chaînage arrière

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

Exemple

Base :

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

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

- Réécriture via l'unique règle
- $Jean \rightarrow x$, $Tom \rightarrow y$

Chaînage arrière

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

Exemple

Base :

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

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

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

Chaînage arrière

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

Exemple

Base :

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

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

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

Table des matières

1. Contexte
2. Notions de logique
3. Dépendance des règles
4. Classes de règles
5. Développement
6. Conclusion

Dépendance des règles

- R_1 dépend de $R_2 \leftrightarrow R_2$ peut amener à déclencher R_1
- Unification de la conclusion de R_2 avec l'hypothèse de R_1
- Construction d'un graphe de dépendances des règles
- Les sommets représentent les règles
- Il existe un arc entre R_1 et R_2 si R_2 dépend de R_1

Construction du graphe

Exemple

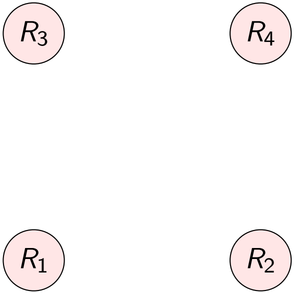
"Tout homme est un humain. Tout humain a un père qui est un homme. Si un homme est le parent d'un autre, alors il est son père. Tout père d'un homme est un de ses parents."

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

Construction du graphe

Base de règles :

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



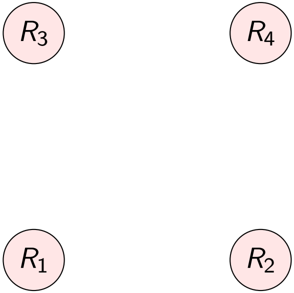
Construction du graphe

Base de règles :

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

R_1 peut elle se redéclencher ?

C_1 a un prédicat différent de H_1

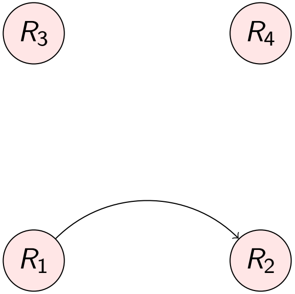


Construction du graphe

Base de règles :

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

R_1 peut amener à déclencher R_2 ?
$C1 = H2$

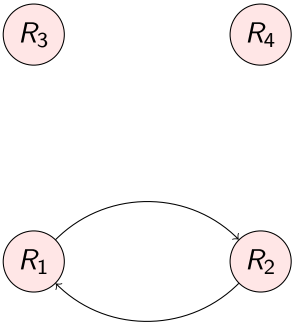


Construction du graphe

Base de règles :

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

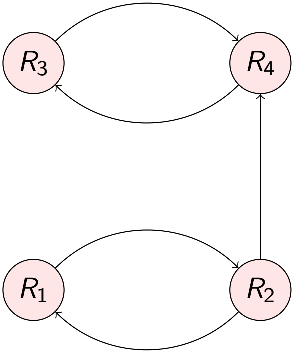


Table des matières

1. Contexte
2. Notions de logique
3. Dépendance des règles
4. Classes de règles
5. Développement
6. Conclusion

Table des matières

1. Contexte
2. Notions de logique
3. Dépendance des règles
4. Classes de règles
   - Classes abstraites
   - Classes concrètes
   - Combinaisons
5. Développement
6. Conclusion

Classes de règles abstraites

- Aucune propriété vérifiable

Classes de règles abstraites

- Aucune propriété vérifiable
- Assurent la décidabilité du problème en suivant certains algorithmes

Classes de règles abstraites

- Aucune propriété vérifiable
- Assurent la décidabilité du problème en suivant certains algorithmes
- Finite Extension Set : algorithmes de chaînage avant

Classes de règles abstraites

- Aucune propriété vérifiable
- Assurent la décidabilité du problème en suivant certains algorithmes
- Finite Extension Set : algorithmes de chaînage avant
- (Greedy) Bounded Treewidth Set : algorithmes de chaînage avant avec condition d'arrêt particulière

Classes de règles abstraites

- Aucune propriété vérifiable
- Assurent la décidabilité du problème en suivant certains algorithmes
- Finite Extension Set
- (Greedy) Bounded Treewidth Set : algorithmes de chaînage avant avec condition d'arrêt particulière
- Finite Unification Set : algorithmes de chaînage arrière

Classes de règles abstraites

- Aucune propriété vérifiable
- Assurent la décidabilité du problème en suivant certains algorithmes
- Finite Extension Set
- (Greedy) Bounded Treewidth Set : algorithmes de chaînage avant avec condition d'arrêt particulière
- Finite Unification Set : algorithmes de chaînage arrière
- Classes incomparables

Table des matières

1 Contexte

2 Notions de logique

3 Dépendance des règles

4 Classes de règles
- Classes abstraites
- Classes concrètes
- Combinaisons

5 Développement

6 Conclusion

Classes de règles concrètes

- Imposent des contraintes sur la forme des règles ou de la base
- Spécialisent les classes abstraites
- Classes pouvant être comparables

Guardée

- Un atome de l'hypothèse contient toutes les variables de celle-ci
- $\exists a \in H_i : variable(H_i) \subseteq variables(a)$
- Simple à vérifier
- $guarded(R) = \{\forall R_i \in R : R_i \text{ est gardée}\} \in GBTS$

Guardée

- Un atome de l'hypothèse contient toutes les variables de celle-ci
- $\exists a \in H_i : variable(H_i) \subseteq variables(a)$
- Simple à vérifier
- $guarded(R) = \{\forall R_i \in R : R_i \text{ est gardée}\} \in GBTS$

Exemple

$R_3 : \forall x \forall y\ (\text{parent}(x,y) \wedge \text{homme}(x) \rightarrow \text{père}(x,y))$
Garde : parent(x,y)

Frontière gardée

- Un atome de l'hypothèse contient toutes les variables de la frontière
- $\exists a \in H$: frontière$(R) \subseteq variables(a)$
- Seule la frontière influe sur l'application d'une règle
- Généralisation des règles gardées
- $fr - guarded(R) = \{\forall R_i \in R : R_i$ a une frontière gardée$\} \in GBTS$

Frontière gardée

- Un atome de l'hypothèse contient toutes les variables de la frontière
- $\exists a \in H$: frontière$(R) \subseteq variables(a)$
- Seule la frontière influe sur l'application d'une règle
- Généralisation des règles gardées
- $fr - guarded(R) = \{\forall R_i \in R : R_i$ a une frontière gardée$\} \in GBTS$

Exemple

$\forall x \forall y (p(x) \wedge q(y) \rightarrow \exists z (r(y, z)))$
Garde-frontière : $q(y)$

Frontière de taille 1

- La frontière de la règle est de taille 1
- Spécialisation des règles à frontière gardée
- Utiles pour les notions d'héritage
- $fr-1(R) = \{\forall R_i \in R : |\text{frontière}(R_i)| = 1\} \in GBTS$

Frontière de taille 1

- La frontière de la règle est de taille 1
- Spécialisation des règles à frontière gardée
- Utiles pour les notions d'héritage
- $fr-1(R) = \{\forall R_i \in R : |\text{frontière}(R_i)| = 1\} \in GBTS$

Exemple

$R_2 : \forall x\ (\text{humain}(x) \rightarrow \exists z\ (\text{homme}(z) \wedge \text{père}(z,x)))$
$\text{frontière}(R_2) = \{x\}$

Hypothèse atomique

- L'hypothèse de la règle ne contient qu'un seul atome
- Spécialisation des règles gardées
- $ah(R) = \{\forall R_i = (H_i, C_i) \in R : |H_i| = 1\} \in GBTS \cap FUS$

Hypothèse atomique

- L'hypothèse de la règle ne contient qu'un seul atome
- Spécialisation des règles gardées
- $ah(R) = \{\forall R_i = (H_i, C_i) \in R : |H_i| = 1\} \in GBTS \cap FUS$

Exemple

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

Domaine restreint

- Les atomes de la conclusion contiennent soit toutes les variables de l'hypothèse, soit aucune
- $\forall a_j \in R_i(variables(H_i) \subseteq variables(a_j)) \vee (variables(H_i) \cap variables(a_j) = \emptyset)$
- $dr(R) = \{\forall R_i \in R : R_i$ a un domaine restreint $\} \in FUS$

Domaine restreint

- Les atomes de la conclusion contiennent soit toutes les variables de l'hypothèse, soit aucune
- $\forall a_j \in R_i(variables(H_i) \subseteq variables(a_j)) \vee (variables(H_i) \cap variables(a_j) = \emptyset)$
- $dr(R) = \{\forall R_i \in R : R_i \text{ a un domaine restreint } \} \in FUS$

Exemple

$R_2 : \forall x\ (\text{humain}(x) \rightarrow \exists z\ (\text{homme}(z) \wedge \text{père}(z{,}x)))$

- $variables(homme(z)) \cap variables(H_2) = \emptyset$
- $variables(H_2) \subseteq variables(pere(z, x))$

Déconnectée

- Frontière vide
- Spécialisation des règles de domaine restreint
- Une seule application nécessaire
- Partage possible de constantes entre l'hypothèse et la conclusion
- $disc(R) = \{\forall R_i \in R : \text{frontière}(R_i) = \emptyset\} \in FUS$

Déconnectée

- Frontière vide
- Spécialisation des règles de domaine restreint
- Une seule application nécessaire
- Partage possible de constantes entre l'hypothèse et la conclusion
- $disc(R) = \{\forall R_i \in R \text{ :frontière}(R_i) = \emptyset\} \in FUS$

Exemple

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

Universelle

- Aucune variable existentielle
- Couramment utilisées
- $rr(R) = \{\forall R_i \in R : variables(R_i) \subseteq variables(H_i)\} \in FES \cap GBTS$

Universelle

- Aucune variable existentielle
- Couramment utilisées
- $rr(R) = \{\forall R_i \in R : variables(R_i) \subseteq variables(H_i)\} \in FES \cap GBTS$

Exemple

R_5 : $\forall x \forall y \forall z$ (mêmeFamille(x,y) $\wedge$ mêmeFamille(y,z) $\rightarrow$ mêmeFamille(x,z))

Faiblement acyclique

- Contrainte sur l'ensemble des règles
- Nécessite l'usage d'une nouvelle structure : le graphe de dépendances des positions
- $wa(R) \in FES$

Faiblement acyclique

- Contrainte sur l'ensemble des règles
- Nécessite l'usage d'une nouvelle structure : le graphe de dépendances des positions
- $wa(R) \in FES$

Exemple de non faible acyclicité

R_1 : $\forall x$ (homme(x) $\rightarrow$ humain(x))
R_2 : $\forall x$ (humain(x) $\rightarrow$ $\exists z$ (homme(z) $\wedge$ père(z,x)))

Faiblement acyclique

- Contrainte sur l'ensemble des règles
- Nécessite l'usage d'une nouvelle structure : le graphe de dépendances des positions
- $wa(R) \in FES$

Exemple de non faible acyclicité

$R_1 : \forall x\ (\text{homme}(x) \rightarrow \text{humain}(x))$
$R_2 : \forall x\ (\text{humain}(x) \rightarrow \exists z\ (\text{homme}(z) \wedge \text{père}(z,x)))$

Exemple de faible acyclicité

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

Sticky

- Contrainte sur l'ensemble des règles
- Marquage des variables
- Une variable marquée ne doit pas apparaître plusieurs fois dans l'hypothèse d'une règle
- $sticky(R) \in FES$

Sticky

- Contrainte sur l'ensemble des règles
- Marquage des variables
- Une variable marquée ne doit pas apparaître plusieurs fois dans l'hypothèse d'une règle
- $sticky(R) \in FES$

Exemple

R_3 : $\forall x \forall y$ (parent(x,y) $\wedge$ homme(x) $\rightarrow$ père(x,y))
R_4 : $\forall x \forall y$ (père(x,y) $\rightarrow$ parent(x,y))

Weakly sticky

- Généralisation de faiblement acyclique et de sticky
- Les variables marquées ne doivent pas apparaître plusieurs fois dans l'hypothèse ou ne pas être dans une position de rang infini
- $ws(R) \notin FES \cup GBTS \cup FUS$

Graphe de dépendances des règles acyclique

- L'ensemble des règles forment un graphe de dépendances sans circuit
- $aGRD(R) \in FES \cup FUS$

Graphe de dépendances des règles acyclique

- L'ensemble des règles forment un graphe de dépendances sans circuit
- $aGRD(R) \in FES \cup FUS$

Exemple

R_2 : $\forall x$ (humain(x) $\rightarrow \exists z$ (homme(z) $\wedge$ père(z,x)))
R_4 : $\forall x \forall y$ (père(x,y) $\rightarrow$ parent(x,y))

Schéma récapitulatif

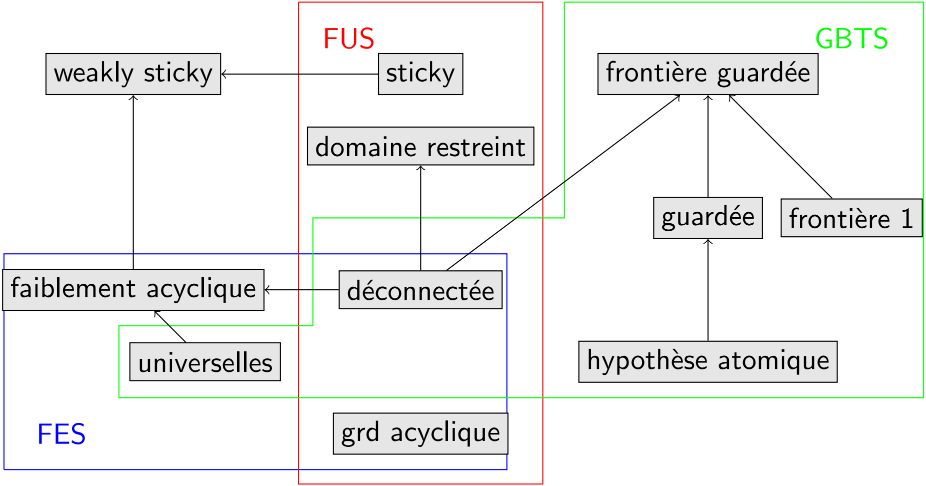


Table des matières

1 Contexte

2 Notions de logique

3 Dépendance des règles

4 Classes de règles
- Classes abstraites
- Classes concrètes
- Combinaisons

5 Développement

6 Conclusion

Décidabilité de l'ensemble de règles

- Ensemble des règles étiqueté par une classe concrète
- Calcul du graphe orienté des composantes fortement connexes
- Détermination des classes concrètes de chaque composante
- Attribution d'étiquettes abstraites pour chaque composante
- Combinaison de celles-ci

Précédence

- Composante C_i *précède* C_j si aucun arc de C_j vers C_i
- Notée $C_i \triangleright C_j$
- Décidable si $FES \triangleright GBTS \triangleright FUS$

Table des matières

1. Contexte
2. Notions de logique
3. Dépendance des règles
4. Classes de règles
5. Développement
6. Conclusion

Spécifications

- Langage Java

Structures de données

- Différents types de graphes
- Structure générique et algorithmes
- Sommets et arcs de types paramétrables
- Règles : graphe biparti non orienté (prédicats et termes, position du terme)
- Dépendances des règles : graphe orienté (règles, aucun)
- Composantes fortement connexes : graphe orienté (ensembles de sommets, aucun) sans circuit
- Dépendances des positions : graphe orienté (positions des prédicats, spécial ou non)

Analyseur

- Divisé en deux parties
- Détermination des classes concrètes
- Combinaison des classes abstraites
- Règles à conclusion atomique (sans perdre de généralité)

Détermination des classes concrètes

- Possibilité d'ajouter de nouveaux tests
- Renvoient des étiquettes
- Indiquent les classes abstraites satisfaites

Combinaison des classes abstraites

- Vérification de l'ensemble des règles
- Parcours du graphe des composantes à partir des sources
- Etiquettes des classes abstraites valuées
- Attribue à chaque composante l'étiquette la plus *petite* compatible
- Décidable si tous les sommets sont étiquetés
- Liste des types d'algorithmes à utiliser

Fichiers

- Lecture et écriture via un format interne
- Conversion des fichiers Datalog (.dtg)
- Sortie PostScript pour l'affichage des graphes

Table des matières

1. Contexte
2. Notions de logique
3. Dépendance des règles
4. Classes de règles
5. Développement
6. Conclusion

Gestion du projet

- Gestionnaire de versions : git
- Réunions fréquentes avec les encadrants
- Problèmes :
  - Domaine nouveau
  - Disparition d'un membre du groupe

Contributions

- Lecture et écriture d'une base à partir et vers un fichier
- Mise en place d'un algorithme d'unification
- Construction du graphe de dépendances des règles
- Calcul des classes concrètes
- Combinaison des classes abstraites
- Etude de la décidabilité

Perspectives

- Combinaison des classes de règles en fonction des complexités
- Interface graphique