∃ASP (Answer Set Programming avec quantificateurs existentiels)

F. Garreau

JIAF 12 décembre 2013

Sommaire

Introduction (problématique)

Problématique

Coder une ontologie en DL-Lite$_R$ dans ASP

Exemple de cas qui pose problème

A $\sqsubseteq$ ∃R
Règle ASP : ∃X, R(X,Y)←A(Y)

Example

homme $\sqsubseteq$ ∃parent
à coder en ∃X, parent(X,Y)←homme(Y)

Règles existentielles

Les règles existentielles permettent de coder les DL légères

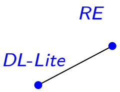
Exemple

$A \sqsubseteq \exists R$

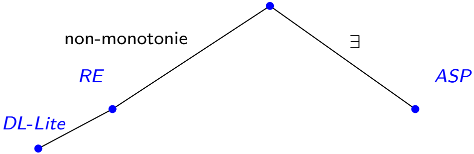
Règle existentielle : $\exists X,\ R(X,Y) \leftarrow A(Y)$

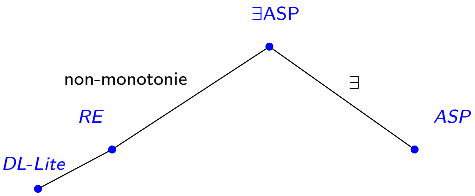
Il manque la non-monotonie

DL-Lite


RE
DL-Lite


RE
DL-Lite
ASP


non-monotonie
∃
RE
ASP
DL-Lite


∃ASP
non-monotonie
∃
RE
ASP
DL-Lite

Règles existentielles + ASP

Definition (Règle)

Une règle ∃ASP est une règle de la forme :

$$H \leftarrow B^+, \text{not } B_1^-, \ldots, \text{not } B_n^-.$$

avec H, B^+, et $B_1^-, \ldots, B_n^-$ des ensembles d'atomes
on appelle :

- H la tête
- B^+ le corps positif
- not $B_1^-, \ldots,$ not B_n^- le corps négatif

Un fait est un ensemble d'atomes

Example

R = parent(X,Y), homme(X) ← homme(Y).
F = {homme(titi),homme(toto),parent(titi,toto)}

Caractéristiques d'une ∃-règle

$$R = a(X,Y),\ b(X) \leftarrow c(Y),\ not(a(Y,Z),\ b(Z))$$

Variables universelles
Variables existentielles

Caractéristiques d'une ∃-règle

$$R = a(X,Y),\ b(X) \leftarrow c(Y),\ not(a(Y,Z),\ b(Z))$$

Variables universelles
Variables existentielles

Caractéristiques d'une ∃-règle

$$R = a(X,Y),\ b(X) \leftarrow c(Y),\ not(a(Y,Z),\ b(Z))$$

Variables universelles
Variables existentielles

Definition (Justification)

Soit F un fait, $\mathcal{V}$ un vocabulaire et I $= (\Delta, .^I)$ une interprétation de $\mathcal{V}$.
Une justification π de F dans I est une application de terms(F) $\rightarrow \Delta$ telle que :

- Si c $\in \mathcal{C}$, π(c) = c^I
- Si p(t$_1$, ..., t$_k$) $\in$ F, (π(t$_1$), ..., π(t$_k$)) $\in$ p^I

Example

$$R = \text{parent(X,Y)} \leftarrow \text{homme(Y)}.$$

$$I = \{\text{homme(titi),homme(toto),parent(titi,toto)}\}$$

Definition

Soit I une interprétation et $R = H \leftarrow B^+, \text{not } B_1^-, \ldots, \text{not } B_n^-.$ une règle ∃ASP

- Un support de R dans I est une justification de B^+ dans I
- Un support de R dans I est dit vérifié s'il s'étend à une justification de H dans I
- Un support est bloqué par I si ce support s'étend à une justification d'un B_i^- dans I

Example

$$R = \text{parent}(X,Y) \leftarrow \text{homme}(Y).$$

$$I = \{\text{homme}(\text{titi}),\text{homme}(\text{toto}),\text{parent}(\text{titi},\text{toto})\}$$

Il existe 2 supports de I dans R
$\pi(Y) = $ titi
$\pi(Y) = $ toto qui est vérifié

Definition (Modèle)

Une interprétation I est un modèle d'une règle R sur $\mathcal{V}$ ssi tous les supports non bloqués de R sont vérifiés. I est un modèle d'un programme P ssi I est modèle de tous les faits et règles du programme

Example

Soit P le programme composé de :

$$R = \mathrm{pere}(X,Y) \leftarrow \mathrm{homme}(Y).$$
$$F = \{\mathrm{homme}(\mathrm{toto})\}$$

Soit 3 interprétations :

$I = \{\mathrm{homme}(\mathrm{toto})\}$ n'est pas modèle
$I' = \{\mathrm{homme}(\mathrm{toto}), \mathrm{pere}(x_0, \mathrm{toto})\}$ modèle
$I'' = \{\mathrm{homme}(\mathrm{toto}), \mathrm{pere}(x_0, \mathrm{toto}), \mathrm{homme}(\mathrm{titi})\}$ modèle

Definition (∃-Answer Set)

Soit P le programme composé de :

$$R = \text{pere}(X,Y) \leftarrow \text{homme}(Y).$$
$$F = \{\text{homme}(\text{toto})\}$$

$$I_0 = \pi(F) = \{\text{homme}(\text{toto})\}$$
$$I_1 = \{\text{homme}(\text{toto}),\text{pere}(x_0,\text{toto})\}$$

Soit :

$I' = \{\text{homme}(\text{toto}),\text{pere}(x_0,\text{toto})\} = I_1$ ∃-Answer Set
$I'' = \{\text{homme}(\text{toto}),\text{pere}(x_0,\text{toto}),\text{homme}(\text{titi})\} \neq I_1$ Simple modèle

Propriété

Soit P un programme ∃ASP safe (donc un programme ASP) et I une interprétation, I est un ∃-Answer Set de P ssi I est un Answer Set de P

Réécriture

Définition de la normalisation

Normalisation ⇒ Pour traiter les variables existentielles dans un corps négatif et avoir un seul atome dans chaque corps négatif

$H_1, \ldots, H_n \leftarrow B_1, \ldots, B_m, not(N_1^1, \ldots, N_{u_1}^1), \ldots, not(N_1^s, \ldots, N_{u_s}^s)$ est normalisée en l'ensemble de règles

- $H_1, \ldots, H_n \leftarrow B_1, \ldots, B_m, not\ N_1, \ldots, not\ N_s$
- $N_1 \leftarrow N_1^1, \ldots, N_{u_1}^1$
- ...
- $N_s \leftarrow N_1^s, \ldots, N_{u_s}^s$

avec N_k nouvel atome $p(X_1, \ldots, X_v)$ avec p nouveau symbole de prédicat et $X_1, \ldots, X_v$ dans $\mathcal{V}_\forall(r)(N_1^k, \ldots, N_{u_k}^k)$

Exemple de la normalisation

Example

traitement(X,aspirine) ←
 migraine(X), not(atteint(X,Y), maladie-sang(Y)).

est normalisée en

traitement(X,aspirine) ← migraine(X), not atteint-maladie-sang(X).
atteint-maladie-sang(X) ← atteint(X,Y), maladie-sang(Y).

Définition de la skolémisation

Skolémisation $\Rightarrow$ Pour traiter les variables existentielles en tête

Example

pere(X,Y),homme(X) $\leftarrow$ homme(Y).

est skolémisée en

pere(f(Y),Y),homme(f(Y)) $\leftarrow$ homme(Y).

Définition de l'expansion

Expansion $\Rightarrow$ Pour obtenir un seul atome en tête de règle

$T_1, \ldots, T_n \leftarrow B_1, \ldots, B_m, not\ N_1, \ldots, not\ N_s$ est expansée en l'ensemble de règles

- $T_1 \leftarrow B_1, \ldots, B_m, not\ N_1, \ldots, not\ N_s$
- $\ldots$
- $T_n \leftarrow B_1, \ldots, B_m, not\ N_1, \ldots, not\ N_s$

Exemple de l'expansion

Example

$$\text{pere}(X,Y),\text{homme}(X) \leftarrow \text{homme}(Y).$$

est expansée en :

$$\text{pere}(f(Y)),Y) \leftarrow \text{homme}(Y).$$
$$\text{homme}(f(Y)) \leftarrow \text{homme}(Y).$$

Un programme ∃ASP normalisé, skolémisé et expansé est un programme ASP standard

Conclusion

Comparaison

2 façons d'appréhender l'extension ∃ASP

- Règles existentielles + non-monotonie
- Réécriture ASP avec ∃ → ASP

Même problème :

- Indécidable

Conclusion

Example

Programme P :

$$R = pere(X,Y), homme(X) \leftarrow homme(Y).$$
$$F = \{homme(titi)\}$$

∃-Answer Set infini :

$$I = \{homme(titi), pere(x_0, titi), homme(x_0), pere(x_1, x_0), homme(x_1), \ldots\}$$

Conclusion

But : Identifier des classes décidables
En règles existentielles :

- 3 classes décidables
- couvrant les DL légères

Peut-on retrouver ces 3 classes en ∃ASP ?
De nouvelles classes ?

Conclusion

Merci!

J-F. Baget, M. Leclère, M-L. Mugnier, and E. Salvat.
On rules with existential variables: Walking the decidability line.
AIJ, 2011.

C. Lefèvre and P. Nicolas.
A first order forward chaining approach for answer set computing.
LPNMR, 2009.

R. Reiter.
A logic for default reasoning.
Artificial Intelligence, 1980.

M. Gelfond and V. Lifschitz.
The stable model semantics for logic programming.
International Conference on Logic Programming, 1988.

M. Gelfond and V. Lifschitz.
Classical negation in logic programs and disjunctive databases.
New Generation Computing, 1991.