∃-ASP

F. Garreau, L. Garcia, C. Lefèvre and I. Stéphan

LERIA, University of Angers, France

ONTOLP July, 27th 2015

Introduction

Problem

Representing knowledge like

By default there exists a color for every object as long as it's not transparent

with an ontology (Description Logic) :

$$\text{object} \sqcap \text{not transparent} \sqsubseteq \exists\text{color}$$

and in ASP :

$$\exists Y\ \text{color}(Y,X) \leftarrow \text{object}(X), \text{not transparent}(X).$$

Introduction

Problem

Representing knowledge like

By default there exists a color for every object as long as it's not transparent

with an ontology (Description Logic) :

$$\text{object} \sqcap \text{not transparent} \sqsubseteq \exists \text{color}$$

and in ASP :

$$\exists Y \ \text{color}(Y,X) \leftarrow \text{object}(X), \text{not transparent}(X).$$

IMPOSSIBLE

Introduction

Two choices

- Extending Description Logic language with default negation
- Extending ASP with existential variables

Goals

- Treating ontologies in ASP (Answer Set Programming)
  - Take advantage of the ASP-Solver performance
  - Enrich the language of ontologies with default negation

Introduction

Two choices

- Extending Description Logic language with default negation (Baget et al. NMR 2014)
- Extending ASP with existential variables

Goals

- Treating ontologies in ASP (Answer Set Programming)
  - Take advantage of the ASP-Solver performance
  - Enrich the language of ontologies with default negation

1. ∃-ASP
   - What is ∃-ASP ?
   - Reduct
   - ∃-Answer Set
2. From ∃-ASP to ASP
3. Conclusion

ASP (syntax)

We consider classical ASP with function symbols and without disjunction

Classical ASP

A rule is of the form

$$H \leftarrow B_1, ..., B_n, \text{not } N_1, ..., \text{not } N_s.$$

where $H, B_1, ..., B_n, N_1, ..., N_s$ are atoms

Safety

A rule is safe if all variables of the head and the negative body of a rule also appear in its positive body

- All variables are universally quantified

Answer Set (semantic)

An ASP program is a set of ASP rules

Finding an Answer Set

- Reduct
- Answet Set (minimal model)

An ASP program can have one, several or no model (called Answer Set)

∃-ASP

∃-ASP : an extension of ASP allowing existentially quantified variables

Extend classical ASP with :

- A conjunction of atoms in head
- A conjunction of atoms for each N_i in the negative body
- Safety property relaxed to accept existential variables in head and negative body without appearing in the positive body

∃-rule (syntax)

Rule

A rule is of the form

$$H_1, \ldots, H_n \leftarrow B_1, \ldots, B_m, \text{not } (N^1_1, \ldots, N^1_{u_1}), \ldots, \text{not } (N^s_1, \ldots, N^s_{u_s}).$$

with $H_1, \ldots, H_n, B_1, \ldots, B_m, N^1_1, \ldots, N^1_{u_1}, \ldots, N^s_1, \ldots, N^s_{u_s}$ atoms and $m, s \geq 0$, $n, u_1, \ldots, u_s \geq 1$

- $head(r) = \{H_1, \ldots, H_n\}$.
- $body^+(r) = \{B_1, \ldots, B_m\}$.
- $body^-(r) = \{\{N^1_1, \ldots, N^1_{u_1}\}, \ldots, \{N^s_1, \ldots, N^s_{u_s}\}\}$.

Example

Example

Let P_U be an ∃-program

$$\mathrm{P}_U = \left\{ \begin{array}{l} r_0 : p(a). \\ r_1 : l(a). \\ r_2 : phdS(X, D), d(D) \leftarrow p(X), not\ (l(X), gC(X, Y)). \end{array} \right\}$$

The rule r_2 means that for each person X there exists a director D and X is a phD student of D, unless X is a lecturer and it exists a course given by X

Example

Example

Let P_U be an ∃-program

$$P_U = \left\{ \begin{array}{l} r_0 : p(a). \\ r_1 : l(a). \\ r_2 : phdS(X, D), d(D) \leftarrow p(X), not\ (l(X), gC(X, Y)). \end{array} \right\}$$

We have :

$\mathcal{V}_\forall(r) = \{X\}$, universal variables

$\mathcal{V}_{H\exists}(r) = \{D\}$, existential variables of the head

$\mathcal{V}_{N\exists}(r) = \{Y\}$, existential variables of the negative body

Skolemization

Skolemization of existential variables in head of rule

Definition (Skolemization)

For each existential variable Y in the head of a rule, we replace Y by a new unique (Skolem) function symbol sk^n_Y with n its arity and $X_1, \ldots, X_n$ its arguments. $X_1, \ldots, X_n$ the variables appearing both in the positive body and in the head. (If n = 0 sk_Y is a Skolem constant symbol)

Example

$$r_2 : phdS(X, D), d(D) \leftarrow p(X), not\ (l(X), gC(X, Y)).$$

is skolemized in

$$sk(r_2) : phdS(X, sk^1_D(X)), d(sk^1_D(X)) \leftarrow p(X), not\ (l(X), gC(X, Y)).$$

Answer Set (semantic)

Skolemization not sufficient to ensure safety

- Existential variables in the negative body

Reduct needs a ground program

- A partial grounding of the program (only universal variables are grounded)
- Leads to a definite program

The partial grounding of a skolemized program can lead to an infinite program

Partial Grounding

Grounding of universal variables only

Example

$$\mathrm{sk}(\mathrm{P}_U) = \left\{ \begin{array}{ll} p(a)., & l(a)., \\ phdS(X, sk_D^1(X)), d(sk_D^1(X)) \leftarrow & p(X), not\ (l(X), gC(X, Y)). \end{array} \right\}$$

$$\mathbf{PG}(sk(P_U)) = \left\{ \begin{array}{l} p(a)., \qquad l(a)., \\ phdS(a, sk_D^1(a)), d(sk_D^1(a)) \leftarrow \\ \quad p(a), not\ (l(a), gC(a, Y))., \\ phdS(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) \leftarrow \\ \quad p(sk_D^1(a)), not\ (l(sk_D^1(a)), gC(sk_D^1(a), Y))., \\ \ldots \end{array} \right\}$$

Reduct

Definition (Reduct)

Let P be an $\exists$-program of language $\mathcal{L}_P$ and $X \subseteq \mathbf{GA}(\mathcal{L}_{sk(P)})$. The reduct of the partial ground program $\mathbf{PG}(sk(P))$ w.r.t. X is the definite partial ground program

$$\begin{array}{ll} \multicolumn{2}{l}{\mathbf{PG}(sk(P))^X =} \\ \{ & head(r) \leftarrow body^+(r). | r \in \mathbf{PG}(sk(P)), \\ & \text{for all } N \in body^-(r) \text{ and} \\ & \text{for all ground substitution } \sigma \text{ over } \mathcal{L}_{sk(P)}, \sigma(N) \not\subseteq X\} \end{array}$$

Reduct example

Example

Let

$$X_1 = \{p(a), l(a), phdS(a, sk_D^1(a)), d(sk_D^1(a)), gC(a, m)\}.$$

Then

$$\begin{array}{l} \mathbf{PG}(sk(P_U))^{X_1} = \{ \\ \quad p(a)., \\ \quad l(a)., \\ \quad phdS(a, sk_D^1(a)), d(sk_D^1(a)) \leftarrow p(a), not\ (l(a), gC(a, Y))., \\ \quad phdS(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) \leftarrow \\ \qquad p(sk_D^1(a)),\ not\ (l(sk_D^1(a)), gC(sk_D^1(a), Y))., \\ \quad \ldots\} \end{array}$$

Reduct example

Example

Let

$$X_1 = \{p(a), l(a), phdS(a, sk_D^1(a)), d(sk_D^1(a)), gC(a, m)\}.$$

Then

$$\begin{array}{l}
\mathbf{PG}(sk(P_U))^{X_1} = \{ \\
\quad p(a)., \\
\quad l(a)., \\
\quad \cancel{phdS(a, sk_D^1(a)), d(sk_D^1(a))} \leftarrow p(a), \cancel{not\ (l(a), gC(a, Y))}., \\
\quad phdS(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) \leftarrow \\
\qquad p(sk_D^1(a)),\ \cancel{not\ (l(sk_D^1(a)), gC(sk_D^1(a), Y))}., \\
\quad \ldots\}
\end{array}$$

Consequence operator and closure

Definition (T_P consequence operator and Cn its closure)

Let P be a definite partial ground program of an ∃-program of language $\mathcal{L}_P$. The operator $T_P : 2^{\mathbf{GA}(\mathcal{L}_P)} \rightarrow 2^{\mathbf{GA}(\mathcal{L}_P)}$ is defined by

$$T_P(X) = \{a | r \in P, a \in head(r), body^+(r) \subseteq X\}.$$

$Cn(P) = \bigcup_{n=0}^{n=+\infty} T_P^n(\emptyset)$ is the least fix-point of the consequence operator T_P.

Definition (∃-Answer Set)

Let P be an ∃-program of language $\mathcal{L}_P$ and $X \subseteq \mathbf{GA}(\mathcal{L}_{sk(P)})$. X is an ∃-answer set of P if and only if $X = Cn(\mathbf{PG}(sk(P))^X)$.

∃-Answer Set

Example

$$X_2 = \{p(a), l(a), phdS(a, sk_D^1(a)), d(sk_D^1(a))\}.$$

and

$$\mathbf{PG}(sk(P_U))^{X_2} = \{$$
$$p(a).,$$
$$l(a).,$$
$$phdS(a, sk_D^1(a)), d(sk_D^1(a)) \leftarrow p(a).,$$
$$phdS(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) \leftarrow p(sk_D^1(a)).,$$
$$\ldots\}$$

$$T^0_{\mathbf{PG}(sk(P_U))^{X_2}} = \emptyset$$

∃-Answer Set

Example

$$X_2 = \{p(a), l(a), phdS(a, sk_D^1(a)), d(sk_D^1(a))\}.$$

and

$$\begin{aligned} &\mathbf{PG}(sk(P_U))^{X_2} = \{ \\ &\quad p(a)., \\ &\quad l(a)., \\ &\quad phdS(a, sk_D^1(a)), d(sk_D^1(a)) \leftarrow p(a)., \\ &\quad phdS(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) \leftarrow p(sk_D^1(a))., \\ &\quad \ldots\} \end{aligned}$$

$$\begin{aligned} T^1_{\mathbf{PG}(sk(P_U))^{X_2}} &= T_{\mathbf{PG}(sk(P_U))^{X_2}}(\emptyset) \\ &= \{p(a), l(a)\} \end{aligned}$$

∃-Answer Set

Example

$$X_2 = \{p(a), l(a), phdS(a, sk_D^1(a)), d(sk_D^1(a))\}.$$

and

$$\mathbf{PG}(sk(P_U))^{X_2} = \{$$
$$p(a).,$$
$$l(a).,$$
$$phdS(a, sk_D^1(a)), d(sk_D^1(a)) \leftarrow p(a).,$$
$$phdS(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) \leftarrow p(sk_D^1(a)).,$$
$$\ldots\}$$

$$T^2_{\mathbf{PG}(sk(P_U))^{X_2}} = T_{\mathbf{PG}(sk(P_U))^{X_2}}(\mathrm{p(a),l(a)})$$
$$= \{p(a),\ l(a)\}\{p(a),\ l(a), phdS(a, sk_D^1(a)),\ d(sk_D^1(a))\}$$

∃-Answer Set

Example

$$X_2 = \{p(a), l(a)phdS(a, sk_D^1(a), d(sk_D^1(a)))\}.$$

and

$$\begin{array}{l} \mathbf{PG}(sk(P_U))^{X_2} = \{ \\ \quad p(a)., \\ \quad l(a)., \\ \quad phdS(a, sk_D^1(a), d(sk_D^1(a))) \leftarrow p(a)., \\ \quad phdS(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) \leftarrow p(sk_D^1(a))., \\ \quad \ldots\} \end{array}$$

$Cn(\mathbf{PG}(sk(P_U))^{X_2}) = \mathrm{X}_2$

Then X_2 is an ∃Answer Set of P_U

1. ∃-ASP
   - What is ∃-ASP ?
   - Reduct
   - ∃-Answer Set
2. From ∃-ASP to ASP
3. Conclusion

From ∃-ASP to ASP

A transformation from ∃-ASP to ASP

- Equivalence between ∃-Answer Set and Answer Set of a transformed program
- Three steps
    - Normalization (removing existential variables and conjunction in negative body)
    - Skolemization (removing existential variables in head)
    - Expansion (removing conjunction of atoms in head)

Normalization

- Remove the conjunctions of atoms from negative parts of the rules
- Remove existential variables from these negative parts
- Equivalent program obtained in terms of answer sets

Normalization

Let r be an ∃-rule

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

The *normalization* of such an ∃-rule is the set of ∃-rules

$$\mathbf{N}(r) =$$
$$\{ \quad H_1, \ldots, H_n \leftarrow B_1, \ldots, B_m, not\ N_1, \ldots, not\ N_s.,$$
$$N_1 \leftarrow N_1^1, \ldots, N_{u_1}^1.,$$
$$\ldots$$
$$N_s \leftarrow N_1^s, \ldots, N_{u_s}^s.\}$$

with N_i containing only universal variables

Normalization

Example

Given

$$P_U = \left\{ \begin{array}{l} r_0 : p(a). \\ r_1 : l(a). \\ r_2 : phdS(X, D), d(D) \leftarrow p(X), not\ (l(X), gC(X, Y)). \end{array} \right\}$$

We have

$$\begin{array}{l} \mathbf{N}(r_2) = \{ \\ \quad r_2^{\dagger} : phdS(X, D), d(D) \leftarrow p(X), not\ p^N(X). \\ \quad r_2^{\ddagger} : p^N(X) \leftarrow l(X), gC(X, Y).\} \end{array}$$

and $\mathbf{N}(P_U) = \{r_0, r_1, r_2^{\dagger}, r_2^{\ddagger}\}$.

Skolemization

Skolemization of the normalized program

- To remove existential variables in head of the rules

Normalization + Skolemization

Example

$$\mathbf{N}(P_U) = \{$$
$$r_0 : p(a).$$
$$r_1 : l(a).$$
$$r_2^{\dagger} : phdS(X, D), d(D) \leftarrow p(X), not\ p^N(X).$$
$$r_2^{\ddagger} : p^N(X) \leftarrow l(X), gC(X, Y).\}$$

$$sk(\mathbf{N}(P_U)) = \{$$
$$r_0 : p(a).$$
$$r_1 : l(a).$$
$$r_2^{\dagger} : phdS(X, sk_D^1(X)), d(sk_D^1(X)) \leftarrow p(X), not\ p^N(X).$$
$$r_2^{\ddagger} : p^N(X) \leftarrow l(X), gC(X, Y).\}$$

Expansion

Expansion of the normalized then skolemized program

- To remove conjunction of atoms in head of the rules

Expansion

Definition

Let P be a Skolemized normalized program and $r \in P$:

$$H_1, \ldots, H_n \leftarrow B_1, \ldots, B_m, \textit{not } N_1, \ldots, \textit{not } N_s.$$

The *expansion* of such a rule is the set defined by :

$$\begin{aligned}
&\mathbf{Exp}(r) = \\
&\{ \quad H_1 \leftarrow B_1, \ldots, B_m, \textit{not } N_1, \ldots, \textit{not } N_s., \\
&\qquad \ldots \\
&\qquad H_n \leftarrow B_1, \ldots, B_m, \textit{not } N_1, \ldots, \textit{not } N_s.\}
\end{aligned}$$

The expansion of P is defined as $\mathbf{Exp}(P) = \bigcup_{r \in P} \mathbf{Exp}(r)$.

Normalization + Skolemization + Expansion

Example

Given the rule r_2 of P_U normalized and skolemized :

$$phdS(X, sk_D^1(X)), d(sk_D^1(X)) \leftarrow p(X), not\ p^N(a).$$

The expansion split r_2 into the two rules :

$$phdS(X, sk_D^1(X)) \leftarrow p(X), not\ p^N(X). \text{ and}$$
$$d(sk_D^1(X)) \leftarrow p(X), not\ p^N(X).$$

Correctness and completeness

Example

$$P_U = \left\{ \begin{array}{l} r_0 : p(a). \\ r_1 : l(a). \\ r_2 : phdS(X, D), d(D) \leftarrow p(X), not\ (l(X), gC(X, Y)). \end{array} \right\}$$

$$\begin{array}{l} \mathbf{Exp}(sk(\mathbf{N}(P_U))) = \{ \\ \quad p(a). \\ \quad l(a). \\ \quad phdS(X, sk_D^1(X)) \leftarrow p(X), not\ p^N(X). \\ \quad d(sk_D^1(X)) \leftarrow p(X), not\ p^N(X). \\ \quad p^N(X) \leftarrow l(X), gC(X, Y).\} \end{array}$$

The transformation is correct and complete

1. ∃-ASP
   - What is ∃-ASP ?
   - Reduct
   - ∃-Answer Set
2. From ∃-ASP to ASP
3. Conclusion

Conclusion

- First step to formalize ASP allowing the use of existential variables
- Well suited to integrate ontologies and rules in a unique formalism
- The translation allows us to use any ASP solver
- A front-end of the solver ASPeRiX already implemented
  - uses an on-the-fly grounding
  - dealing with variables in a more efficient way

Future work

- In depth comparison with other formalisms
- Parallel work
  - existential rules extended with non-monotonic negation (Baget et al. NMR 2014)
- Dealing efficiently with conjunctive queries
  - not obvious due to non-monotonic aspect and inconsistencies in ASP

Future work

Thank you !