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):

\[ \text{object} \sqcap \neg \text{not transparent} \sqsubseteq \exists \text{color} \]

and in ASP:

\[ \exists Y \ \text{color}(Y,X) \leftarrow \text{object}(X), \neg \text{transparent}(X). \]
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**
1. **∃-ASP**
   - What is ∃-ASP?
   - Reduct
   - ∃-Answer Set

2. **From ∃-ASP to ASP**

3. **Conclusion**
We consider classical ASP with function symbols and without disjunction

**Classical ASP**
A rule is of the form

\[ H \leftarrow B_1, \ldots, B_n, \text{ not } N_1, \ldots, \text{ not } N_s. \]

where \( H, B_1, \ldots, B_n, N_1, \ldots, N_s \) 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
- Answer Set (minimal model)

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

A rule is of the form

\[ H_1, \ldots, H_n \leftarrow B_1, \ldots, B_m, \neg(N_1^1, \ldots, N_{u_1}^1), \ldots, \neg(N_s^1, \ldots, N_{u_s}^s). \]

with \( H_1, \ldots, H_n, B_1, \ldots, B_m, N_1^1, \ldots, N_{u_1}^1, \ldots, N_1^s, \ldots, N_{u_s}^s \) atoms

and \( m, s \geq 0, n, u_1, \ldots, u_s \geq 1 \)

- \( head(r) = \{ H_1, \ldots, H_n \} \).
- \( body^+(r) = \{ B_1, \ldots, B_m \} \).
- \( body^-(r) = \{ \{ N_1^1, \ldots, N_{u_1}^1 \}, \ldots, \{ N_1^s, \ldots, N_{u_s}^s \} \} \).
Example

Let $P_U$ be an $\exists$-program

\[
P_U = \begin{cases} 
  r_0 : p(a). \\
  r_1 : l(a). \\
  r_2 : phdS(X, D), d(D) \leftarrow p(X), not (l(X), gC(X, Y)). 
\end{cases}
\]

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

Let \( P_U \) be an \( \exists \)-program

\[
P_U = \begin{cases} 
    r_0 : p(a). \\
    r_1 : l(a). \\
    r_2 : phdS(X, D), d(D) \leftarrow p(X), \neg(l(X), gC(X, Y)). 
\end{cases}
\]

We have:
\( \mathcal{V}_\forall(r) = \{X\} \), universal variables
\( \mathcal{V}_\exists(r) = \{D\} \), existential variables of the head
\( \mathcal{V}_\neg\exists(r) = \{Y\} \), existential variables of the negative body
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 $X_1, ..., X_n$ its arguments. $X_1, ..., X_n$ 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

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

\[
\text{sk}(P_U) = \begin{cases} 
  p(a), & l(a), \\
  \text{phdS}(X, sk_D^1(X)), d(sk_D^1(X)) \leftarrow p(X), \text{not} (l(X), gC(X, Y)). 
\end{cases}
\]

\[
\text{PG}(\text{sk}(P_U)) = \begin{cases} 
  p(a), & l(a), \\
  \text{phdS}(a, sk_D^1(a)), d(sk_D^1(a)) \leftarrow p(a), \text{not} (l(a), gC(a, Y))., \\
  \text{phdS}(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) \leftarrow p(sk_D^1(a)), \text{not} (l(sk_D^1(a)), gC(sk_D^1(a), Y))., \\
  \ldots 
\end{cases}
\]
Definition (Reduct)

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

$$PG(sk(P))^X = \{ head(r) \leftarrow body^+(r). | r \in PG(sk(P)), \text{ for all } N \in body^-(r) \text{ and } \text{for all ground substitution } \sigma \text{ over } L_{sk(P)}, \sigma(N) \not\subseteq X \}$$
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

\[ PG(sk(P_U))^{X_1} = \{ \]

\[ p(a), \]
\[ l(a), \]
\[ phdS(a, sk_D^1(a)), d(sk_D^1(a)) \leftarrow p(a), not (l(a), gC(a, Y))., \]
\[ phdS(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) \leftarrow p(sk_D^1(a)), not (l(sk_D^1(a)), gC(sk_D^1(a), Y))., \]
\[ \ldots \} \]
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

\[ \text{PG}\left( sk(P_U) \right)^{X_1} = \{ \]
\[ p(a) ., \]
\[ l(a) ., \]
\[ phdS(a, sk_D^1(a)) , d(sk_D^1(a)) \leftarrow p(a), \text{not} \ (l(a), gC(a, Y)) ., \]
\[ phdS(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) \leftarrow \]
\[ p(sk_D^1(a)), \text{not} \ (l(sk_D^1(a)), gC(sk_D^1(a), Y)) ., \]
\[ \ldots \} \]
Consequence operator and closure

**Definition (\(T_P\) consequence operator and \(Cn\) its closure)**

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

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

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

**Definition (\(\exists\)-Answer Set)**

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

Example

\[ X_2 = \{ p(a), l(a), phdS(a, sk^1_D(a)), d(sk^1_D(a)) \}. \]

and

\[ PG(sk(P_U))^{X_2} = \{ \]
\[ p(a), \]
\[ l(a), \]
\[ phdS(a, sk^1_D(a)), d(sk^1_D(a)) \leftarrow p(a), \]
\[ phdS(sk^1_D(a), sk^1_D(sk^1_D(a))), d(sk^1_D(sk^1_D(a))) \leftarrow p(sk^1_D(a)), \]
\[ \ldots \} \]

\[ T^0_{PG(sk(P_U))^{X_2}} = \emptyset \]
Example

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

and

\[
\begin{align*}
PG(sk(P_U))^X_2 &= \{ \\
p(a), \\
l(a), \\
phdS(a, sk_D^1(a)), d(sk_D^1(a)) &\leftarrow p(a), \\
phdS(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) &\leftarrow p(sk_D^1(a)), \\
\ldots \} \\
\end{align*}
\]

\[
T^1_{PG(sk(P_U))^X_2} = T_{PG(sk(P_U))^X_2(\emptyset)} = \{ p(a), l(a) \}
\]
∃-Answer Set

Example

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

and

\[
\text{PG}(sk(P_U))^{X_2} = \{
\begin{align*}
p(a), & \\
l(a), & \\
\text{phdS}(a, sk_D^1(a)), d(sk_D^1(a)) & \leftarrow p(a), \\
\text{phdS}(sk_D^1(a), sk_D^1(sk_D^1(a))), d(sk_D^1(sk_D^1(a))) & \leftarrow p(sk_D^1(a)), \\
\ldots & 
\end{align*}
\}
\]

\[
T_{2}^{2}\text{PG}(sk(P_U))^{X_2} = T_{\text{PG}(sk(P_U))^{X_2}}^{2}(p(a), l(a))
\]

\[
= \{ p(a), l(a) \}(p(a), l(a), \text{phdS}(a, sk_D^1(a)), d(sk_D^1(a))
\]
Example

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

and

\[
\begin{align*}
\mathbf{PG}(sk(P_U))^{X_2} &= \{ \\
p(a), & \\
l(a), & \\
\text{phdS}(a, sk_D^1(a), d(sk_D^1(a)) & \leftarrow p(a), \\
\text{phdS}(sk_D^1(a), sk_D^1(sk_D^1(a)), d(sk_D^1(sk_D^1(a))) & \leftarrow p(sk_D^1(a)), \\
\ldots & \
\}
\end{align*}
\]

\[ Cn(\mathbf{PG}(sk(P_U))^{X_2}) = X_2 \]

Then \( X_2 \) is an \( \exists \)Answer Set of \( P_U \).
1. $\exists$-ASP
   - What is $\exists$-ASP?
   - Reduct
   - $\exists$-Answer Set

2. From $\exists$-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 $\exists$-rule

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

The normalization of such an $\exists$-rule is the set of $\exists$-rules

$$\mathbf{N}(r) =$$

$$\left\{ H_1, \ldots, H_n \leftarrow B_1, \ldots, B_m, \text{not } N_1, \ldots, \text{not } N_s, \right.$$  

$$N_1 \leftarrow N_1^1, \ldots, N_{u_1}^1, \right.$$  

$$\ldots$$  

$$N_s \leftarrow N_s^s, \ldots, N_{u_s}^s \}$$

with $N_i$ containing only universal variables.
Normalization

Example

Given

\[ P_U = \{ \begin{align*}
    r_0 & : p(a). \\
    r_1 & : l(a). \\
    r_2 & : phdS(X, D), d(D) \leftarrow p(X), \text{not } (l(X), gC(X, Y)).
\end{align*} \} \]

We have

\[ \mathbf{N}(r_2) = \{ \begin{align*}
    r_2^\dagger & : phdS(X, D), d(D) \leftarrow p(X), \text{not } p^N(X). \\
    r_2^\ddagger & : p^N(X) \leftarrow l(X), gC(X, Y).
\end{align*} \} \]

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

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

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

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

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

$$\text{Exp}(r) = \left\{ \begin{array}{c}
H_1 \leftarrow B_1, \ldots, B_m, \text{not } N_1, \ldots, \text{not } N_s, \\
\ldots \\
H_n \leftarrow B_1, \ldots, B_m, \text{not } N_1, \ldots, \text{not } N_s. \end{array} \right\}$$

The expansion of $P$ is defined as $\text{Exp}(P) = \bigcup_{r \in P} \text{Exp}(r)$.
Normalization + Skolemization + Expansion

Example

Given the rule $r_2$ of $P_U$ normalized and skolemized:

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

The expansion split $r_2$ into the two rules:

$$phdS(X, sk^1_D(X)) \leftarrow p(X), not p^N(X). \text{ and}$$
$$d(sk^1_D(X)) \leftarrow p(X), not p^N(X).$$
Correctness and completeness

Example

\[ P_U = \begin{cases} 
  r_0 : p(a). \\
  r_1 : l(a). \\
  r_2 : phdS(X, D), d(D) \leftarrow p(X), not (l(X), gC(X, Y)). 
\end{cases} \]

\[ \text{Exp}(sk(N(P_U))) = \begin{cases} 
  p(a). \\
  l(a). \\
  phdS(X, sk_D^1(X)) \leftarrow p(X), not p^N(X). \\
  d(sk_D^1(X)) \leftarrow p(X), not p^N(X). \\
  p^N(X) \leftarrow l(X), gC(X, Y). 
\end{cases} \]

The transformation is correct and complete.
## Summary

<table>
<thead>
<tr>
<th>1</th>
<th>$\exists$-ASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>• What is $\exists$-ASP?</td>
<td></td>
</tr>
<tr>
<td>• Reduct</td>
<td></td>
</tr>
<tr>
<td>• $\exists$-Answer Set</td>
<td></td>
</tr>
</tbody>
</table>

| 2 | From $\exists$-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
Thank you!