Upward Refinement for Conceptual Blending in Description Logic — An ASP-based Approach and Case Study in $\mathcal{EL}^{++}$ —

Roberto Confalonieri$^1$  Manfred Eppe$^{1,2}$  Marco Schorlemmer$^1$
Oliver Kutz$^3$  Rafael Peñaloza$^3$  Enric Plaza$^1$

$^1$IIIA-CSIC, Barcelona, Spain
$^2$International Computer Science Institute, Berkeley, USA
$^3$Free University of Bozen-Bolzano, Italy

ONTOLP 2015, 25 July, 2015, Buenos Aires, Argentina
Outline

1. Introduction
2. Computational Conceptual Blending Overview
3. Upward Refinement Operator for $\mathcal{EL}^{++}$
4. Generalisation as Search Problem in Answer Set Programming
5. Conclusion and Future Works
Introduction

- **COINVENT Project goals:**
  - Computationally feasible model of *conceptual blending*
    - Cognitive theory described by Fauconnier and Turner [1998, 2002].
      - The universal creative engine of human thinking.
    - Model concept invention
  - Symbolic approach, based on formal logic (CASL, OWL)
  - Applications areas: Mathematics, Music, and Computer Icon Design

- **This paper goals:**
  - Define an upward refinement operator for $\mathcal{EL}^{++}$
  - Use ASP to generate and search for $\mathcal{EL}^{++}$ concept generalisations
Introduction

- COINVENT Project goals:
  - Computationally feasible model of conceptual blending
    - Cognitive theory described by Fauconnier and Turner [1998, 2002].
      The universal creative engine of human thinking.
    - Model concept invention
  - Symbolic approach, based on formal logic (CASL, OWL)
  - Applications areas: Mathematics, Music, and Computer Icon Design

- This paper goals:
  - Define an upward refinement operator for $\mathcal{EL}^{++}$
  - Use ASP to generate and search for $\mathcal{EL}^{++}$ concept generalisations
Creating Icons by Conceptual Blending

“Give me an icon with meaning Preview-Document”

Conceptual Blending Engine

Designer

protégé

ICON LIBRARY
OWL
Conceptual Blending Engine

“Give me an icon with meaning Preview-Document”

Conceptual Blending Engine

Amalgamation OWL2ASP

ASP2OWL Evaluation

ICON LIBRARY OWL

Confalonieri, et al  Upward Refinement for $\mathcal{EL}^{++}$ in ASP  25/07/2015 4 / 23
Computational Model of Conceptual Blending - Amalgamation

- Amalgamation originates from the notion of *amalgam* Ontañón and Plaza [2010] in case-based reasoning.
- It applies to any language $\mathcal{L}$ such that $\langle \mathcal{L}, \sqsubseteq \rangle$ is a poset.

![Diagram]

- An *amalgam* of two input concepts is a new concept that combines parts from the original descriptions:
  - Find *Generic Space* ($G$) of input concepts (commonalities) and try to combine non-common elements in $I_1$ and $I_2$.
  - Often, input concepts $I_1$ and $I_2$ cannot be combined directly (inconsistency or insatisfaction of some properties).
  - Input concepts have to be first *generalised* into $I'_1$ and $I'_2$.
  - $I'_1$ and $I'_2$ can be finally blended to obtain a ‘good’ $B$. 

Confalonieri, et al.
Creating Icons by Conceptual Blending

Generic Space

Input 1

Input 2

Blend

Confalonieri, et al
Modeling Computer Icons in $\mathcal{EL}^{++}$

$N_C = \{\text{Icon, Sign, Document, MagnifyingGlass, Pen, HardDisk}\}$
$N_r = \{\text{hasSign, isAbove, isLeft, isRight, isBelow, isSpatialRelation}\}$

### Background Knowledge

<table>
<thead>
<tr>
<th>Icon $\sqsubseteq$ Thing</th>
<th>$domain(isInSpatialRelation) \sqsubseteq$ Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign $\sqsubseteq$ Thing</td>
<td>$range(isInSpatialRelation) \sqsubseteq$ Sign</td>
</tr>
<tr>
<td>Document $\sqsubseteq$ Sign</td>
<td>…</td>
</tr>
<tr>
<td>HardDisk $\sqsubseteq$ Sign</td>
<td>…</td>
</tr>
<tr>
<td>MagnifyingGlass $\sqsubseteq$ Sign</td>
<td>$isAbove \sqsubseteq isInSpatialRelation$</td>
</tr>
<tr>
<td>Pen $\sqsubseteq$ Sign</td>
<td>$isBehind \sqsubseteq isInSpatialRelation$</td>
</tr>
<tr>
<td>$domain(hasSign) \sqsubseteq$ Icon</td>
<td>$isLeft \sqsubseteq isInSpatialRelation$</td>
</tr>
<tr>
<td>$range(hasSign) \sqsubseteq$ Sign</td>
<td>$isRight \sqsubseteq isInSpatialRelation$</td>
</tr>
</tbody>
</table>
Modeling Computer Icons in $\mathcal{EL}^{++}$

$N_C = \{\text{Icon, Sign, Document, MagnifyingGlass, Pen, HardDisk}\}$
$N_r = \{\text{hasSign, isAbove, isLeft, isRight, isBelow, isSpatialRelation}\}$

**Background Knowledge**

<table>
<thead>
<tr>
<th>Icon $\sqsubseteq$ Thing</th>
<th>domain(isInSpatialRelation) $\sqsubseteq$ Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign $\sqsubseteq$ Thing</td>
<td>range(isInSpatialRelation) $\sqsubseteq$ Sign</td>
</tr>
<tr>
<td>Document $\sqsubseteq$ Sign</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>HardDisk $\sqsubseteq$ Sign</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>MagnifyingGlass $\sqsubseteq$ Sign</td>
<td>isAbove $\sqsubseteq$ isInSpatialRelation</td>
</tr>
<tr>
<td>Pen $\sqsubseteq$ Sign</td>
<td>isBehind $\sqsubseteq$ isInSpatialRelation</td>
</tr>
<tr>
<td>$domain(\text{hasSign})$ $\sqsubseteq$ Icon</td>
<td>isLeft $\sqsubseteq$ isInSpatialRelation</td>
</tr>
<tr>
<td>$range(\text{hasSign})$ $\sqsubseteq$ Sign</td>
<td>isRight $\sqsubseteq$ isInSpatialRelation</td>
</tr>
</tbody>
</table>
Modeling Computer Icons in $\mathcal{EL}^{++}$

$N_C = \{\text{Icon, Sign, Document, MagnifyingGlass, Pen, HardDisk}\}$

$N_r = \{\text{hasSign, isAbove, isLeft, isRight, isBelow, isSpatialRelation}\}$

### Background Knowledge

- $\text{Icon} \subseteq \text{Thing}$
- $\text{Sign} \subseteq \text{Thing}$
- $\text{Document} \subseteq \text{Sign}$
- $\text{HardDisk} \subseteq \text{Sign}$
- $\text{MagnifyingGlass} \subseteq \text{Sign}$
- $\text{Pen} \subseteq \text{Sign}$

### Domain Knowledge

- Domain Knowledge:
  - $\text{Icon} \cap \exists \text{hasSign. HardDisk} \cap \exists \text{hasSign.}$
    - (MagnifyingGlass $\cap \exists \text{isAbove. HardDisk}$)
  - Input 1
  - Input 2

- Input 1
- Input 2
Blending Computer Icons

Input 1
Icon ⊓ ∃hasSign.HardDisk ⊓ ∃hasSign. (MagnifyingGlass ⊓ ∃isAbove.HardDisk)

Input 2
Icon ⊓ ∃hasSign.Document ⊓ ∃hasSign. (Pen ⊓ ∃isAbove.Document)

Blend
Icon ⊓ ∃hasSign.HardDisk ⊓ ∃hasSign. (MagnifyingGlass ⊓ ∃isAbove.HardDisk)
Icon ⊓ ∃hasSign.Document ⊓ ∃hasSign. (Pen ⊓ ∃isAbove.Document)
Generalising Icon Concepts

Input 1

Icon ⊓ ∃hasSign.HardDisk ⊓ ∃hasSign.(MagnifyingGlass ⊓ ∃isAbove.HardDisk)

Input 2

Icon ⊓ ∃hasSign.Document ⊓ ∃hasSign.(Pen ⊓ ∃isAbove.Document)
Generalising Icon Concepts

Generic Space

Input 1

Input 2
Generalising Icon Concepts

Generic Space

Input 1

Input 2

Confalonieri, et al
Upward Refinement for EL++ in ASP
25/07/2015 9 / 23
Generalisation and Refinement operators

- The generalisation in the amalgamation algorithm is based on a search in the poset \( \langle \mathcal{L}(\mathcal{T}), \subseteq \mathcal{T} \rangle \)
- The generalisation of an \( \mathcal{EL}^{++} \) concept can be done through an \textit{upward refinement operator} \( \gamma \)

**Refinement operator properties**

- Local finiteness
- Properness
- Completeness
Generalisation of $\mathcal{EL}^{++}$ concepts

- The upward refinement operator generalises an $\mathcal{EL}^{++}$ concept by:
  - generalising a concept
  - generalising the concept filling the range of a role
  - generalising a role
  - ‘removing’ a role/concept

Properties

- Trade-off between completeness and finiteness
  - The operator is finite but not proper and complete
  - It is possible that the generic space is not least general
  - Not a big issue for conceptual blending, the important thing is to find the commonalities between the concepts
Generalisation in ASP

The search for generalisations is modeled as an ASP search problem where the ‘goal’ is to find a generic space for two input $\mathcal{EL}^{++}$ concepts:

1. $\mathcal{EL}^{++}$ concept in background and domain knowledge are translated to ASP facts (base part)
2. Generalisation operators are implemented as a step-wise process to generalise $\mathcal{EL}^{++}$ concepts in the domain knowledge until they are not equal (cumulative part)
3. Each ASP stable model returns a generalisation path from the input specifications to a generic space
## Background knowledge in ASP

<table>
<thead>
<tr>
<th>Thing relation</th>
<th>ASP expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document $\sqsubseteq$ Sign</td>
<td>...</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Role relation</th>
<th>ASP expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain(hasSign) $\sqsubseteq$ Icon</td>
<td>role(role_hasSign). domain(role_hasSign,concept_Icon).</td>
</tr>
<tr>
<td>range(hasSign) $\sqsubseteq$ Sign</td>
<td>range(role_hasSign,concept_Sign).</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>isAbove $\sqsubseteq$ isInSpatialRelation</td>
<td>subRole(role_isAbove,role_isInSpatialRelation).</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
</tbody>
</table>
Domain knowledge in ASP - EditDocument

| Icon ⊓ ∃ hasSign. Document ⊓ ∃ hasSign. (Pen ⊓ ∃ isAbove. Document) |
| spec(spec_EditDocument). |
| hasConjunct(spec_EditDocument, root, concept_Icon, 0). |
| hasConjunct(spec_EditDocument, root, roleEx1, 0). |
| roleExHasRoleAndConcept(spec_EditDocument, roleEx1, role_hasSign, concept_Document, 0). |
| hasConjunct(spec_EditDocument, root, roleEx2, 0). |
| roleExHasRoleAndConcept(spec_EditDocument, roleEx2, role_hasSign, conjunction1, 0). |
| hasConjunct(spec_EditDocument, conjunction1, concept_Pen, 0). |
| hasConjunct(spec_EditDocument, conjunction1, roleEx3, 0). |
| roleExHasRoleAndConcept(spec_EditDocument, roleEx3, role_isAbove, concept_Document, 0). |
Generalisation in ASP

- A generalisation operator is defined via:
  - *precondition rule:* states when it is possible to execute a generalisation operation (*poss/3*)
  - *inertia rule:* states when an element of a specification stays as it is after the execution (*noninertial/3*)
  - *effect rule:* models how a generalisation operator changes an input specification (*noninertial/3* - only for renaming operators)

- The execution of a generalisation operator is modeled via:
  - \( \text{exec}(\gamma, s, t) \): a generalisation operator \( \gamma \) is applied to \( s \) at a step \( t \)
Generalisation of Atomic concepts

\[ \text{exec}\left(\text{genConcept}(Ex, C_{\text{sub}}, C_{\text{super}}), s, t\right) \] denotes the generalisation of a concept \( C_{\text{sub}} \) to a concept \( C_{\text{super}} \) in \( Ex \) of an icon specification \( s \) at step \( t \) using \( \gamma(A) \). The precondition rule for generalising \( A \) is:

\[
\begin{align*}
\text{poss}(\text{genConcept}(Ex, C_{\text{sub}}, C_{\text{super}}), s_1, t) & \leftarrow \\
& \text{hasConjunct}(s_1, Ex, C_{\text{sub}}, t), \text{subConcept}(C_{\text{sub}}, C_{\text{super}}), \\
& \text{not roleExHasRoleAndConcept}(s_1, C_{\text{sub}}, _, _, t), \\
& \text{not hasConjunct}(s_1, C_{\text{sub}}, _), \\
& \text{conjunctNotEq}(s_1, s_2, C_{\text{sub}}, t), \\
& \text{not exec}(\text{genConcept}(Ex, C_{\text{sub}}, _), s_2, t), \text{spec}(s_2).
\end{align*}
\]

\[
\text{poss}(
\text{genConcept}(\text{conjunction1}, \text{concept_\_Pen}, \text{concept_\_Sign}), \text{EditDoc}, 0)
\]
Generalisation path

- A generalisation path $P = \{\text{exec}(\gamma_1, s, t_1), \cdots, \text{exec}(\gamma_n, s, t_n)\}$ of $s$ is a sequence of generalisation operator steps applied in $s$.

- Generalisation paths are generated with the following choice rule, that allows one or zero generalisation operations per specification at $t$.

$$0\{\text{exec}(a, s, t) : \text{poss}(a, s, t)\}1 \leftarrow \text{not genericReached}(t), \text{spec}(s)$$
Generalisation path

\[ P_{EditDoc} = \{ \]
\[
\text{exec}(\text{genConceptInRole}(\text{roleEx3, role\_isAbove, concept\_Document, concept\_Sign}), \text{EditDoc}, 0),
\]
\[
\text{exec}(\text{genConcept(endowment1, concept\_Pen, concept\_Sign}), \text{EditDoc}, 1),
\]
\[
\text{exec}(\text{genConceptInRole}(\text{roleEx1, role\_hasSign, concept\_Document, concept\_Sign}), \text{EditDoc}, 2)
\}\]

**EditDoc**

0. Icon \( \sqcap \exists \text{hasSign. Document} \sqcap \exists \text{hasSign.}(\text{Pen} \sqcap \exists \text{isAbove. Sign}) \)
1. Icon \( \sqcap \exists \text{hasSign. Document} \sqcap \exists \text{hasSign.}(\text{Sign} \sqcap \exists \text{isAbove. Sign}) \)
2. Icon \( \sqcap \exists \text{hasSign. Sign} \sqcap \exists \text{hasSign.}(\text{Sign} \sqcap \exists \text{isAbove. Sign}) \)
Blends in $\mathcal{EL}^{++}$

Generic Space

Generalisation

Input 1

Input 2

Blend

Confalonieri, et al  
Upward Refinement for $\mathcal{EL}^{++}$ in ASP  
25/07/2015 19 / 23
Blends in $\mathcal{EL}^{++}$

- Blends are computed as MGS of pairs of generalised concepts
- In $\mathcal{EL}^{++}$, the MGS is defined by $\sqcap$

**Generalisations**

\[
C'_1 = \text{Icon} \sqcap \exists \text{hasSign}. \text{Sign} \sqcap \exists \text{hasSign}. (\text{MagnifyingGlass} \sqcap \exists \text{isAbove}. \text{Sign}) \\
C'_2 = \text{Icon} \sqcap \exists \text{hasSign}. \text{Document} \sqcap \exists \text{hasSign}. (\text{Sign} \sqcap \exists \text{isAbove}. \text{Document})
\]

**MGS**

\[
C'_1 \sqcap C'_2 = \text{Icon} \sqcap \exists \text{hasSign}. \text{Sign} \sqcap \exists \text{hasSign}. (\text{MagnifyingGlass} \sqcap \exists \text{isAbove}. \text{Sign}) \\
\sqcap \\
\text{Icon} \sqcap \exists \text{hasSign}. \text{Document} \sqcap \exists \text{hasSign}. (\text{Sign} \sqcap \exists \text{isAbove}. \text{Document})
\]

- **Needed:** Normalisation rules for rewriting the blend $C'_1 \sqcap C'_2$ into a ‘normal’ form (no trivial)

**Blend**

\[
B = \text{Icon} \sqcap \exists \text{hasSign}. \text{Document} \sqcap \exists \text{hasSign}. (\text{MagnifyingGlass} \sqcap \exists \text{isAbove}. \text{Document})
\]
Conclusion and Future Works

Achieved Results

- Upward refinement operator for $\mathcal{EL}^{++}$
- ASP implementation

Future Works

- To extend the icon ontology to include *icon meanings*
- To identify normalisation rules (for blend rewriting and operator properness)
- To extend the approach to a richer DL (without $\sqcap$)
- To incorporate blend evaluation


Thanks for the attention!