Representing n-ary Relations in DL
A Generic Reification Strategy

Niels Grewe
University of Rostock, Institute of Philosophy
Outline

Introduction

The Conventional Strategy

Best Practise Driven Reification

A Generic Reification Strategy

Conclusion
The problem

Description logics are usually restricted to the two-variable fragment of first order logic:

<table>
<thead>
<tr>
<th>Predicate Type</th>
<th>Natural Language</th>
<th>First Order Logic</th>
<th>Description Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Binary Predicate</strong></td>
<td>Every book has an author.</td>
<td>$\forall x (Bx \supset \exists y (Ryx \land Hy))$</td>
<td>$B \sqcap \exists R.H$</td>
</tr>
<tr>
<td><strong>Ternary Predicate</strong></td>
<td>Every critic criticises somebody for something.</td>
<td>$\forall x (Cx \supset \exists y \exists z (Sxyz \land Hy \land Tz))$</td>
<td>?</td>
</tr>
</tbody>
</table>
Ternary Relations in the Biomedical Domain

1. Cell $C$ has organelle $O$ with function $F$.

2. Disposition $D$ has realisation $R$ under conditions $C$ (cf. Schulz/Jansen 2009).

3. Disorder $D$ has morphology $M$ at site $S$ (cf. Spackman et al. 2002).

Irreducible ternary relations? Some tests:

1. Is the function predicated of the cell or the organelle? Clearly of the organelle. Two ‘daisy-chained’ binary relations: $C$\text{has_part }O \text{ and } O\text{has_function }F$.

2. Does the disposition have the conditions or does the realisation have them? Neither dispositions nor realisations simply have conditions. This is a true ternary relation.

3. Does the morphology have a site or does the site have the morphology? Arbitrary decisions. Cannot be analysed into ‘daisy-chained’ binary relations.
Ternary Relations in the Biomedical Domain

1. Cell $C$ has organelle $O$ with function $F$.

2. Disposition $D$ has realisation $R$ under conditions $C$ (cf. Schulz/Jansen 2009).

3. Disorder $D$ has morphology $M$ at site $S$ (cf. Spackman et al. 2002).

Irreducible ternary relations? Some tests:

1. Is the function predicated of the cell or the organelle? Clearly of the organelle. Two ‘daisy-chained’ binary relations: $Chas_part O$ and $Ohas_function F$.

2. Does the disposition have the conditions or does the realisation have them? Neither dispositions nor realisations simply have conditions. This is a true ternary relation.

3. Does the morphology have a site or does the site have the morphology? Arbitrary decisions. Cannot be analysed into ‘daisy-chained’ binary relations.
Reification
Representing relations as classes

Disorder \( \xrightarrow{\text{disorder}} \) C_{\text{has_morphology_at_site}} \( \xrightarrow{\text{has_site}} \) Site

Morphology \( \xrightarrow{\text{has_morphology}} \)
Problems

- Uniqueness constraints enforced by set-theoretical semantics don’t apply. (When are two reified relations the same?)
- Role composition, transitive closure etc. are not possible for reified relations.
- Introduces classes for mere technical reasons, lacking clear ontological status.
- Causes a proliferation of classes and roles, impeding manageability of the ontology.
- Design of the reifications is at the modellers liberty. Quality varies.
- Different reification strategies can hinder ontology alignment.
Reification Best Practices
Cf. Severi/Fiadeiro/Ekserdjian 2010

[The] problem lies first of all in helping modellers to conceptualize the real world in a way that can lead to a better representation [...] . By ‘better’ we mean a more controlled use of reification and a closer fit between the resulting ontology and the real-world domain [...] .  
(Severi/Fiadeiro/Ekserdjian 2010, 417)

- Less is more: Try to use as few reified relations as possible.
- Be smart about it: Reify sub-relations that correspond to proper ontological classes.
- Reuse: Prefer to reify relations that partake in many other relations.
Reification Best Practises
Cf. Severi/Fiadeiro/Ekserdjian 2010

[The] problem lies first of all in helping modellers to conceptualize the real world in a way that can lead to a better representation [. . . ]. By ‘better’ we mean a more controlled use of reification and a closer fit between the resulting ontology and the real-world domain [. . . ]. (Severi/Fiadeiro/Ekserdjian 2010, 417)

- Less is more: Try to use as few reified relations as possible.
- Be smart about it: Reify sub-relations that correspond to proper ontological classes.
- Reuse: Prefer to reify relations that partake in many other relations.
Smarter Reification

- Disorder
- Sign
- Site

Relations:
- Disorder has_sign Sign
- Sign has_morphology Morphology
- Sign has_site Site
Disorder \(\xrightarrow{\text{disorder}}\) Disorder
\(\xrightarrow{C_{\text{has\_morphology\_at\_site}}}\) Morphology
\(\xrightarrow{\text{has\_site}}\) Site

Morphology \(\xrightarrow{\text{has\_morphology}}\) Morphology
\(\xrightarrow{\text{has\_site}}\) Site

Disorder \(\xrightarrow{\text{has\_sign}}\) Sign
\(\xrightarrow{\text{has\_site}}\) Site
Remaining Problems

- Does not ensure inter-modeller consistency.
- Requires ontologically ‘fitting’ classes to be identified.
- Necessary constraints need to be enforced on a case by case basis.

Desideratum: A ‘fallback’ strategy that

- can be used independently from domain ontology considerations
- minimizes inter-modeller variability
- makes modelling mistakes unlikely
Remaining Problems

- Does not ensure inter-modeller consistency.
- Requires ontologically ‘fitting’ classes to be identified.
- Necessary constraints need to be enforced on a case by case basis.

Desideratum: A ‘fallback’ strategy that

- can be used independently from domain ontology considerations
- minimizes inter-modeller variability
- makes modelling mistakes unlikely
Generic Reified Relations in DL

Strategy: Explicitly encode crucial features of relations in DL classes describing the arguments of a relation

- Order
- Arity
- Value ranges

Order and value ranges

Using the primitive roles has_value and succeeds to construct a class ‘Argument’:

Argument \equiv \forall \text{succeeds}. \text{Argument} \sqcap \leq 1 \text{succeeds} \sqcap \\ \leq 1 \text{succeeds}^- \sqcap = 1 \text{has_value}. \top \\
\text{Dis(succeeds, succeeds}^-)
**Generic Reified Relations in DL**

**Strategy:** Explicitly encode crucial features of relations in DL classes describing the arguments of a relation

- Order
- Arity
- Value ranges

**Order and value ranges**

Using the primitive roles `has_value` and `succeeds` to construct a class ‘Argument’:

\[
\text{Argument} \equiv \forall \text{succeeds}. \text{Argument} \sqcap \leq 1 \text{succeeds} \sqcap \\
\leq 1 \text{succeeds}^- \sqcap = 1 \text{has_value}. \top \\
\text{Dis}(\text{succeeds}, \text{succeeds}^-)
\]
Generic Reified Relations in DL
continued...

Arity I

It is tempting to define the class ‘Relation’ with a $\text{has_argument}$ role and a role composition axiom:

\[
\text{Relation} \equiv \geq 1 \text{ has_argument.} \text{Argument}
\]

\[
\text{has_argument} \circ \text{succeeds}^{-} \sqsubseteq \text{has_argument}
\]
Generic Reified Relations in DL
continued...

Arity I

It is tempting to define the class ‘Relation’ with a \textit{has\textunderscore argument} role and a role composition axiom:

\[
\text{Relation} \equiv \geq 1 \text{ has\textunderscore argument} . \text{Argument}
\]

\[
\text{has\textunderscore argument} \circ \text{succeeds\textasciitilde} \subseteq \text{has\textunderscore argument}
\]

\textit{SROIQ} (and hence OWL 2) only allows simple roles in number restrictions!
Generic Reified Relations in DL
Arity II

Workaround: Explicitly model beginning and end of the argument chain.

\[ \text{Argument}1 \equiv \text{Argument} \sqcap \neg \exists \text{succeeds.}\text{Argument} \sqcap \]
\[ = 1 \text{ has}_\text{first}_\text{argument}^\text{−}.\text{Relation} \]

\[ \text{Argument}N \equiv \text{Argument} \sqcap \geq 1 \text{succeeds} \sqcap \geq 1 \text{succeeds}^\text{−} \]

\[ \text{Argument}Last \equiv \text{Argument} \sqcap \neg \exists \text{succeeds}^\text{−}.\text{Argument} \]

\[ \text{Argument}0 \sqcap \text{Argument}N \equiv \bot \]
Example: Ternary Relation

- Argument1
- TernaryRelation
- Argument2
- Argument3

- \( X_1 \)
- \( X_2 \)
- \( X_3 \)

- has_first_argument
- succeeds
- has_value
- succeeds
- has_value
How to use this framework

- The proposed generic structure of relations can be regarded as a template.
- For concrete relations that shall be reified, the structure is subclassed by simple value restrictions.

Analysing the disposition example:

\[ \text{ArgumentD} \equiv \text{Argument1} \sqcap \forall \text{has\_value}.\text{Disposition} \]

\[ \text{ArgumentR} \equiv \text{ArgumentN} \sqcap \exists \text{succeeds}.\text{ArgumentD} \sqcap \forall \text{has\_value}.\text{Realisation} \]

\[ \text{ArgumentC} \equiv \text{ArgumentN} \sqcap \text{ArgumentLast} \sqcap \exists \text{succeeds}.\text{ArgumentR} \sqcap \forall \text{has\_value}.\text{Condition} \]

\[ C_{\text{disp\_has\_real\_under\_cond}} \equiv \text{Relation} \sqcap \exists \text{has\_first\_argument}.\text{ArgumentC} \]
How to use this framework
Performance

Performance measurements using the HermiT reasoner:

- **BL** Baseline profile, ‘toy’-ontology based on OGMS
- **NA-20** BL + 20 reified ternary relations using the ‘naïve’ pattern
- **R3-10** BL + 10 reified ternary relations using the new scheme
- **R3-20** BL + 10 reified ternary relations using the new scheme
- **R4-10** BL + 10 reified quaternary relations using the new scheme

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Classification Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL</td>
<td>13.69s</td>
</tr>
<tr>
<td>NA-20</td>
<td>26.59s</td>
</tr>
<tr>
<td>R3-10</td>
<td>38.39s</td>
</tr>
<tr>
<td>R3-20</td>
<td>74.37s</td>
</tr>
<tr>
<td>R4-10</td>
<td>73.67s</td>
</tr>
</tbody>
</table>
Conclusion

Benefits:
- Systematic framework for reified relations in the general case
- Does not ‘pollute’ the domain specific namespace
- Leaves little space for arbitrary modeller decisions
- Eases comparability and consistency

Costs:
- Even greater proliferation of classes
- Notable performance penalty

Open Questions:
- Where exactly can reified relations fit into a top-level ontology?
- Is it possible to hide the complexity resulting from the reification scheme?

Get the code:
https://code.google.com/p/nrel-ontology-template/
Thank you!

This work is being supported by a grant from the DFG
Project ‘GoodOD’
Good Ontology Design