Universität Rostock

Traditio et Innovatio

DFG

Representing n-ary Relations in DL

A Generic Reification Strategy

Niels Grewe
University of Rostock, Institute of Philosophy

Outline

The problem

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

binary predicate

Natural language: Every book has an author.
First order logic: $\forall x(Bx \supset \exists y(Ryx \wedge Hy))$
Description logic: $B \sqcap \exists R.H$

ternary predicate

Natural language: Every critic criticises somebody for something.
First order logic: $\forall x(Cx \supset \exists y \exists z(Sxyz \wedge Hy \wedge Tz)$
Description logic: ?

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.

Ternary Relations in the Biomedical Domain

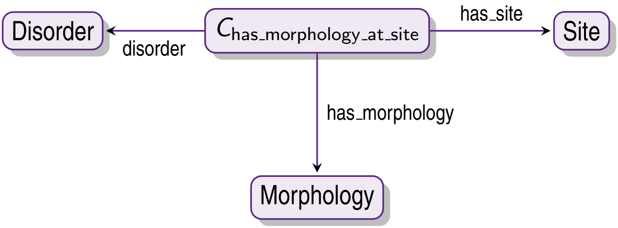
1. Cell C has organelle O with function F.
2. Disposition D has realisation R under conditions C (cf. Schulz/Jansen 2009).
3. Disorder D has morphology M at site S (cf. Spackman et al. 2002).

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



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 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.

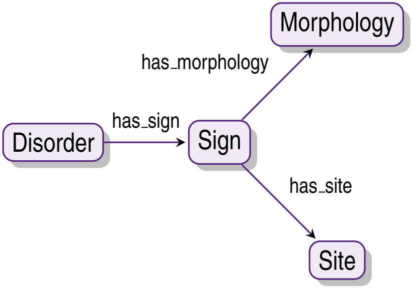
Reification Best Practises

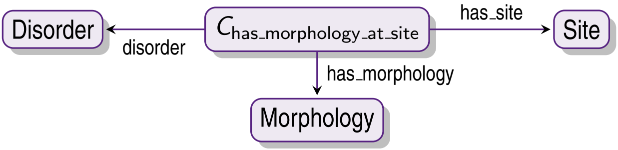
Cf. Severi/Fiadeiro/Ekserdjian 2010

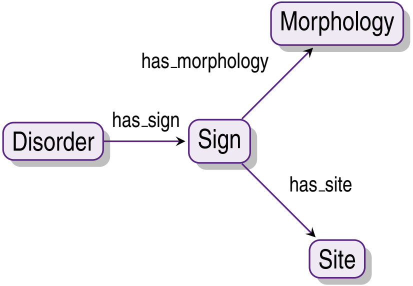
> *[The] problem lies first of all in helping modellers to conceptualize the real world in a way that can lead to a better representation [. . .]. By 'better' we mean a more controlled use of reification and a closer fit between the resulting ontology and the real-world domain [. . .]. (Severi/Fiadeiro/Ekserdjian 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
disorder
$C_{\text{has\_morphology\_at\_site}}$
has_site
Site
has_morphology
Morphology


Morphology
has_morphology
has_sign
Disorder
Sign
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':

$\text{Argument} \equiv \forall \textit{succeeds}.\text{Argument} \sqcap \leq 1\ \textit{succeeds} \sqcap$
$\quad \leq 1\ \textit{succeeds}^{-} \sqcap = 1\ \textit{has\_value}.\top$

$\text{Dis}(\textit{succeeds}, \textit{succeeds}^{-})$

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':

$$\text{Argument} \equiv \forall \textit{succeeds}.\text{Argument} \sqcap \leq 1\ \textit{succeeds} \sqcap$$
$$\leq 1\ \textit{succeeds}^{-} \ \sqcap\ = 1\ \textit{has\_value}.\top$$
$$\text{Dis}(\textit{succeeds}, \textit{succeeds}^{-})$$

Generic Reified Relations in DL

continued...

Arity I

It is tempting to define the class ‘Relation’ with a *has_argument* role and a role composition axiom:

$$\text{Relation} \equiv \geq 1\ \textit{has\_argument}.\text{Argument}$$

$$\text{has\_argument} \circ \text{succeeds}^{-} \sqsubseteq \text{has\_argument}$$

Generic Reified Relations in DL

continued...

Arity I

It is tempting to define the class 'Relation' with a *has_argument* role and a role composition axiom:

$$\mathsf{Relation} \equiv \geq 1\ \mathit{has\_argument}.\mathsf{Argument}$$

$$\mathsf{has\_argument} \circ \mathsf{succeeds}^{-} \sqsubseteq \mathsf{has\_argument}$$

$\mathcal{SROIQ}$ (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.

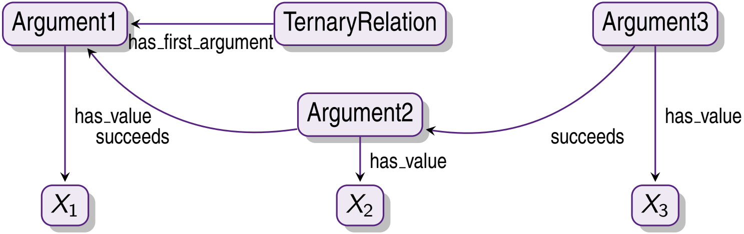
$$\text{Argument1} \equiv \text{Argument} \sqcap \neg\exists \mathit{succeeds}.\text{Argument} \sqcap {} = 1\ \mathit{has\_first\_argument}^{-}.\text{Relation}$$

$$\text{ArgumentN} \equiv \text{Argument} \sqcap \geq 1\ \mathit{succeeds} \sqcap \geq 1\ \mathit{succeeds}^{-}$$

$$\text{ArgumentLast} \equiv \text{Argument} \sqcap \neg\exists \mathit{succeeds}^{-}.\text{Argument}$$

$$\text{Argument0} \sqcap \text{ArgumentN} \equiv \bot$$

Example: Ternary Relation



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:

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

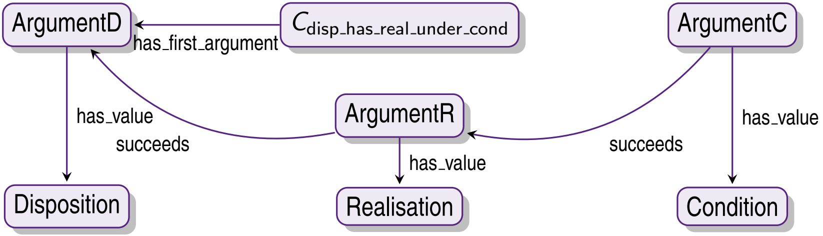
$$\text{ArgumentR} \equiv \text{ArgumentN} \sqcap \exists \text{succeeds.ArgumentD} \sqcap \forall \text{has\_value.Realisation}$$

$$\text{ArgumentC} \equiv \text{ArgumentN} \sqcap \text{ArgumentLast} \sqcap$$

$$\exists \text{succeeds.ArgumentR} \sqcap \forall \text{has\_value.Condition}$$

$$C_{\text{disp\_has\_real\_under\_cond}} \equiv \text{Relation} \sqcap \exists \text{has\_first\_argument.ArgumentC}$$

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

Ontology	Classification Time
BL	13.69s
NA-20	26,59s
R3-10	38,39s
R3-20	74,37s
R4-10	73,67s

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