OntoIOp – Nonmonotonicity, Linked Data, etc.

Till Mossakowski[1,2], Michael Grüninger[3],
Christoph Lange[1], Oliver Kutz[1]

[1]SFB/TR 8 "Spatial cognition", University of Bremen, Germany
[2]DFKI GmbH, Bremen, Germany
[3]University of Toronto, Canada

2012-04-18

Nonmonotonic Logics: OWL Integrity Constraints

Last time's outcome: RIF-PRD is no option. So what instead?

OWL Integrity Constraints (proposal by Fabian)

$$\mathcal{K} = \{\mathsf{Product}(p)\}$$
$$\alpha : \mathsf{Product} \sqsubseteq \exists \mathsf{hasProducer}.\mathsf{Producer}$$

Semantics: based on closed-world assumption.
α, regarded as an integrity constraint, is violated in $\mathcal{K}$ because p does not have a producer.

OWL Integrity Constraints

$$\mathcal{K} = \{\text{Product}(p), \{p\} \sqsubseteq \exists \text{hasProducer}.\text{Producer}\}$$
$$\alpha : \text{Product} \sqsubseteq \exists \text{hasProducer}.\text{Producer}$$

α is violated because

- $\mathcal{K}$ implies to the existence of an unnamed invidiual
- the integrity constraint semantics of "$\sqsubseteq \exists$hasProducer.Producer" only requires the existence of a named individual.

Still quite counterintuitive: two identical occurences of "$\sqsubseteq \exists$hasProducer.Producer" with completely different semantics. SPARQL or FOL would be more intuitive here. It would then be intuitively clear that integrity constraints have a semantics different from OWL- they stem from a different language.

Circumscription

In circumscription,

$$a \wedge (b \vee c)$$

has three models: $\{a, b, c\}$, $\{a, b\}$ and $\{a, c\}$. Circumscription will only take the latter two *minimal models*.
We could introduce a minimization operator for DOL ontologies.

Syntax for Module Extraction

Extracting a module from a given ontology:

```
ONTO                   ::= ...
                         | ONTO extract RESTRICTION-SIGNATURE CONSERVATIVE
RESTRICTION-SIGNATURE  ::= ENTITY-ITEMS
ENTITY-ITEMS           ::= ENTITY ,..., ENTITY
CONSERVATIVE           ::= %ccons | %mcons
```

Declaring that one ontology is a module of another one (creating a proof obligation):

```
LINK-DEFN   ::= INTPR-DEFN | MODULE-DEFN | ALIGN-DEFN
MODULE-DEFN ::= module MODULE-NAME : MODULE-TYPE for RESTRICTION-SIGNATURE
MODULE-NAME ::= IRI
MODULE-TYPE ::= ONTO of ONTO
```

Two aspects of Linked Data compliance

1. HTTP IRIs as identifiers for everything in a distributed ontology
2. graph of OntoIOp-conforming ontology languages, logics, and translations published as linked open dataset

IRIs (encourage HTTP URLs) as identifiers

- IRIs: web-scalable identification and disambiguation
- "inject" IRIs into basic ontology languages that don't support them yet (e.g. F-logic) or don't enforce them (e.g. Common Logic)
- encourage usage of HTTP URLs as IRIs:
  - encourage publishing of machine readable information about things at these IRIs
  - most suitable format: RDF; others (DOL Text, DOL XML, etc.) possible as well

("Linked Data" principle)

IRIs and their syntactic abbreviation (I)

Now fixed a syntax for abbreviated IRIs in DOL
(see later slides for OntoIOp registry details):

```
%prefix( :       <http://example.org/repo/my#>
         cal:    <http://example.org/repo/Calendar/>
         intvl:  <http://example.org/repo/time_interval#>
         log:    <http://purl.net/dol/logics/>
         ser:    <http://purl.net/dol/serializations/>
         trans:  <http://purl.net/dol/translations/> )%

distributed-ontology MyDistributedOntology

logic log:DLLiteR syntax ser:OWL/Manchester
ontology cal: = ...

logic log:CommonLogic syntax ser:CommonLogic/CLIF
ontology intvl:time_interval = ...

interpretation i : cal: to intvl:time_interval =
  logic trans:DLLiteRtoSROIQ, logic trans:SROIQtoCommonLogic,
  cal:earlierThan ↦ intvl:precedes
```

IRIs and their syntactic abbreviation (II)

Result after prefix expansion:

```
distributed-ontology <http://example.org/repo/my#MyDistributedOntology>

logic <http://purl.net/dol/logics/DLLiteR>
syntax <http://purl.net/dol/serializations/OWL/Manchester>
ontology <http://example.org/repo/Calendar/> = ...

logic <http://purl.net/dol/logics/CommonLogic>
syntax <http://purl.net/dol/serializations/CommonLogic/CLIF>
ontology <http://example.org/repo/time_interval#time_interval> = ...

interpretation <http://example.org/my-ontology#i> :
  <http://example.org/repo/Calendar/>
  to <http://example.org/repo/time_interval#time_interval> =
    logic <http://purl.net/dol/serializations/DLLiteRtoSROIQ>,
    logic <http://purl.net/dol/serializations/SROIQtoCommonLogic>,
    <http://example.org/repo/Calendar/earlierThan> ↦
      <http://example.org/repo/time_interval#precedes>
```

Authors don't see this, but systems.

Semantics specification starts here.

Injecting IRIs into basic ontologies (I)

Two problems with basic ontology languages in DOL:

- DOL expects IRI identifiers everywhere, but some basic ontology languages . . .
  - support IRI identifiers but don't enforce their usage (e.g. Common Logic)
  - don't support IRI identifiers at all (e.g. TPTP)

  . . . for their entities, axioms, etc.
- DOL meta-language supports syntactic abbreviation of IRIs. In basic ontologies, . . .
  - one might also like to abbreviate IRIs (if the language doesn't support it, as e.g. CLIF)
  - one might want to reuse meta-level prefixes (if the language supports abbreviation, as e.g. OWL Manchester)

  (This is more of an author's convenience problem, but still. . .)

Injecting IRIs into basic ontologies (II)

Idea: With the specification of the conformance of a language (abstract syntax plus concrete serializations), also specify a translation of **globally-scoped identifiers** of this language to IRIs.

1. Mandatory: one namespace for the whole basic ontology
2. Optional: multiple namespaces, reusing prefixes declared on DOL meta-level

Possible cases:

- IRI support level: Basic language ...
  - doesn't support IRIs (e.g. TPTP, CASL)
  - supports optional IRIs (e.g. Common Logic)
  - requires IRIs (e.g. OWL)
- IRI abbreviation mechanism: Basic language ...
  - supports DOL-compatible prefixes
  - doesn't support prefixes, or in a way different from DOL

Injecting IRIs into basic ontologies (III)

Case 1: Basic language doesn't support IRIs (e.g. CASL)

```
%prefix( : <http://example.org/repo/> %[ others omitted ]% )%
distributed-ontology MyDistributedOntology
logic log:SubPCFOLEq
ontology Total_Order
%prefix( : <http://example.org/repo/Total_Order#> )% =
  sort OrdElem
  pred __ <= __ : OrdElem * OrdElem
  forall x, y, z : OrdElem
  . x <= x %% ...
ontology Ordered_List = Total_Order then
  %prefix( : <http://example.org/repo/Ordered_List#> )%
  sort ListElem %% ...
```

- DOL interprets *OrdElem* as *http://example.org/repo/Total_Order#OrdElem*.
- *x* remains *x* (DOL doesn't see it).
- Need local prefix annotations (as close as possible to basic ontologies) to disambiguate between multiple basic ontologies.

Injecting IRIs into basic ontologies (IV)

Case 2: Basic language doesn't enforce IRIs (e.g. Common Logic)

```
%prefix( : <http://example.org/repo/> %[ others omitted ]% )%
distributed-ontology MyDistributedOntology
logic log:CommonLogic
ontology total_order
%prefix( : <http://example.org/repo/total_order#> )% =
  (forall (x y)
    (if (and (http://example.org/repo/products#Compound x)
             (http://example.org/repo/products#Compound y))
        (if (and (isPartOf x y) (isPartOf y x)) (= x y))))
```

- DOL interprets *isPartOf* as *http://example.org/repo/total_order#isPartOf*.
- *x* remains *x* (DOL doesn't see it).
- *http://example.org/repo/products#Compound* remains, as it is an IRI already.

Injecting IRIs into basic ontologies (V)

Case 3: Basic language requires IRIs (e.g. OWL) $\Rightarrow$ all fine
So let's look at IRI abbreviation mechanisms.
Case 1: Basic language supports DOL-compatible prefixes (e.g. OWL Manchester):

- Basic ontologies in external files should be self-contained (i.e. declare all prefixes used)
- But for basic ontologies *inside a DOL file* it makes sense to share prefixes.

Injecting DOL prefixes into basic ontologies (I)

Without reuse of DOL prefixes:

```
%prefix( : <http://example.org/repo/>
         foaf: <http://xmlns.com/foaf/0.1/> %[ others omitted ]% )%

distributed-ontology MultipleOWLOntologies

logic log:SROIQ syntax ser:OWL2/Manchester

ontology Workplace =
  Prefix : <http://example.org/repo/Workplace#>
  Class Employee: ...

ontology SocialNetworks = foaf: then
  Prefix : <http://example.org/repo/SocialNetworks#>
  Prefix foaf: <http://xmlns.com/foaf/0.1/>
  Class SocialNetwork: EquivalentTo: hasParticipant only foaf:Person

interpretation i : foaf: to Workplace =
  foaf:Person ↦ <http://example.org/repo/Workplace#>
```

Injecting DOL prefixes into basic ontologies (II)

Reusing DOL prefixes instead:

```
%prefix( : <http://example.org/repo/>
         foaf: <http://xmlns.com/foaf/0.1/> %[ others omitted ]% )%
distributed-ontology MultipleOWLOntologies

logic log:SROIQ syntax ser:OWL2/Manchester

ontology Workplace =
  Prefix : <http://example.org/repo/Workplace#>          # overrides outer
  Class Employee: ...

ontology SocialNetworks = foaf: then
  # foaf:  is reused from outer DOL level
  Prefix : <http://example.org/repo/SocialNetworks#>   # overrides outer
  Class SocialNetwork: EquivalentTo: hasParticipant only foaf:Person

interpretation i : foaf: to Workplace =
  foaf:Person ↦ <http://example.org/repo/Workplace#>
```

"Just" need to teach basic ontology parser about additional prefixes (usually works by prepending them to the basic ontology)

Injecting DOL prefixes into basic ontologies (III)

Case 2: Basic language does not support DOL-style prefixes

- use DOL's empty prefix for unprefixed globally-scoped identifiers (We had this above)
- optionally reserve a character *c* as prefix separator:
  - *c* must be allowed in identifiers in the basic language (otherwise the basic language parser would fail)
  - DOL interprets *prefix c localname* in the basic ontology as an IRI

```
%prefix( :   <http://example.org/repo/>
         TO: <http://example.org/repo/Total_Order#>
         %[ others omitted ]% )%
logic log:SubPCFOLEq
ontology Total_Order
  sort TO?OrdElem
  ...
```

TO?OrdElem interpreted as abbreviated IRI

Question: Specify a fixed *c* once per basic ontology language, or introduce a DOL construct for declaring *c* as needed?