POSH - The Prolog OWL Shell

Chris Mungall

Abstract. Two of the most common ways of processing and manip-
ulating OWL ontologies are through an ontology editing environment
(e.g. Protege or TopBraid) or via a programmatic interface, such as the
OWL API. A complementary method is to use an OWL-aware command
line shell. Posh, the Prolog OWL Shell is an interactive toplevel read-
eval-print-loop interface that provides powerful capabilities for querying
and transforming ontologies. It includes a bridge to the OWLAPI and to
multiple OWL reasoners, and allows a mixture of closed-world rule-based
querying on top of open world reasoning. It also provides an interface to
POPL, the Prolog Ontology Processing Language.

1 Motivation

Powerful and feature-rich java libraries such as Jena and the OWLAPI[3] pro-
vide a means of developing infrastructure and applications that leverage OWL
technology. However, these libraries are not always appropriate for lightweight
exploratory programming, scripting or hacking purposes.

Posh is a simple wrapper for the Thea library[11]. It extends Thea by allowing
for interactive use, and is also intended to open up the capabilities of Thea for
non-Prolog programmers.

2 Posh: The Prolog OWL Shell

The best way to describe Posh is by example. The following example command
initiates Posh with java enabled and the fruitfly anatomy ontology[6] loaded into
memory:

thea-poshj http://purl.obolibrary.org/obo/fbbt.owl

This puts the user into an enhanced prolog REPL (Read-Eval-Print-Loop)
shell with provides expected features such as readline support and history.

A little knowledge of prolog syntax helps - variables are indicated by a lead-
ing upper case character or underscore, and queries are terminated by a “.”,
Posh makes use of prolog’s infix operators to define a Manchester Syntax[4]-like
Domain Specific Language with quasi-DL style symbols. You can type arbitrary
prolog goals during a posh session, but posh also provides a number of top-level
convenience commands, such as l to list all axioms associated with a named
entity and q for performing a structural query on asserted axioms.

The following query finds all subclass axioms whose second argument is an
existential restriction using the connected to property (in other words, it finds
the connected parts of a fly):
?- q X < connected_to some Y.

The \( \sqsubseteq \) operator is shorthand for SubClassOf (posh favors brevity at the possible expense of obfuscation). We can query more specifically for all asserted connections to the antennal lobe, getting back a single axiom as an answer:

?- q X < connected_to some 'antennal lobe'.
'cortex of antennal lobe'<connected_to some 'antennal lobe'.

This illustrates a useful configurable feature whereby labels in queries are substituted for the corresponding IRI; this is very handy for bio-ontologies where the convention is to use numeric IDs.

A standard SELECT-WHERE type construct is also supported:

?- q X where X < connected_to some 'antennal lobe'.
'cortex of antennal lobe'.

2.1 Reasoner Integration

Posh comes bundled with jars for Pellet[9], HermiT[7] and FaCT++[10], as well as an OWLLink client library[5] and an experimental OWL2-RL reasoner implemented in prolog (Vassiladis, unpublished). The \texttt{init} command can be used to fire up a reasoner, reasoner queries are embedded in curly braces. At this point we introduce another feature, the ability to feed in arbitrary prolog goals to constrain which axioms are passed on to the reasoner (see [2] for context):

?- init pellet with [filter(A,(A\=annotationProperty(_,),A\=annotationAssertion(_,_,_,_))]).
?- q \{X < connected_to some 'antennal lobe'}.

Annotations are not given to the reasoner, and after initialization the reasoner is queried for the contents of the curly braces. Closed-world prolog queries can be freely mixed with open-world reasoner queries. We can use the prolog negation-as-failure operator to find all neuron classes that are not known to be restricted to the head:

?- q X where \{X < neuron\},\+\{X<part_of some head\}.

Users can write their own prolog predicates and mix these in with their queries, or they can take advantage of powerful SWI-Prolog library[12] queries, such as those for spatial queries, NLP or relational database access.
2.2 POPL : Prolog Ontology Processing Language

Posh also makes use of a new extension of Thea called the Prolog Ontology Process Language. This is similar to OPPL2[1], but offers additional features as well as the full expressive power of prolog.

POPL directives can be issued directly from within an interactive Posh session. For example, the following command will rewrite all overlaps `SomeValues-From` expressions to use a nested class expression:

```
overlaps some Y ==> has_part some part_of some Y.
```

The same thing can be done using the powerful visitor pattern in the OWL API, albeit at the cost of more lines of code.

The following example adds equivalence axioms wherever two classes have the same label:

```
add X==Y where labelAnnotation_value(X,N),labelAnnotation_value(Y,N),X\=Y.
```

3 Other Features

Posh is a bit of a Swiss-army knife with a growing list of features, including:

- basic graphviz visualization of ontology graphs;
- ascii subclass hierarchy trees;
- configurable template-based ontology editing and bulk axiom entry;
- integration with unix commands such as `grep`;
- integration with bioinformatics sql databases.

4 Pathologically Obfuscated Semantic Hacking

Posh occupies something of a niche in the ecosystem of OWL tooling. Upstanding software engineers are likely to favour solid java APIs, and ontology power users can make use of an increasing number of plugins for their ontology development environment of choice. In particular, the new SPARQL-DL[8] extension for the OWL API\(^1\) removes many of the limitations of pure DL queries (e.g. use of annotations in queries, closed-world negation), and consequently some of the comparative advantages of hybrid prolog-reasoner querying.

Furthermore, REPLs to the java OWL API are available in a number of scripting languages from Groovy to Lisp, exemplified in toolkits such as El-Vira and LSW\(^2\) (LSW also allows scripting from with Protege 4). Nevertheless, there are many advantages to a pure prolog/Thea based environment, such as the ability to manipulate OWL constructs directly in the host language.

In the spirit of what was once the hacking language of choice for bioinformaticians, Perl\(^3\), an alternative acronym for Posh is the “Pathologically Obfuscated Semantic Hacker”. Posh aims to fill a similar niche, allowing powerful obfuscated operations to be scripted quickly and easily.

---

1 http://www.derivo.de/en/resources/sparql-dl-api/
2 http://svn.mumble.net:8080/svn/lsw/trunk/
3 whose alternative acronym is “Pathologically Eclectic Rubbish Lister”
5 Conclusions

Posh is a powerful and somewhat ad-hoc shell that allows prolog, OWL reasoning and unix operations to be mashed up interactively on the command line. It is available as part of the Thea library\(^4\), but to obtain the latest version please see the main Posh page (http://blipkit.wordpress.com/posh/) which also includes a collection of examples and a companion guide to the examples in this note.

References


\(^4\) http://github.com/vangelisv/thea