ROBOT: A command-line tool for ontology development

James A. Overton,[1]* Heiko Dietze,[2] Shahim Essaid,[3]
David Osumi-Sutherland,[4] and Christopher J. Mungall [2]

[1]Knocean, Toronto, Ontario, Canada
[2]Lawrence Berkeley National Laboratory, Berkeley, California, USA
[3]Oregon Health and Science University Library, Portland, Oregon, USA
[4]European Bioinformatics Institute, TODO LOCATION

ABSTRACT

ROBOT is a command-line tool for working with ontologies, especially Open Biomedical Ontologies. It builds on OWLAPI and is designed to eventually replace Oort and OWLTools. Currently implemented commands include: reporting on differences between ontologies, merging ontologies, extracting ontology modules, filtering by ObjectProperties, and reasoning. Commands can be chained together to form powerful, repeatable workflows. ROBOT is in early development but is available for use under an open source (BSD) license.

1 INTRODUCTION

ROBOT is a new command-line tool for working with ontologies, with special emphasis on Open Biomedical Ontologies (OBO) (Smith *et al.*, 2007). It provides convenient commands for merging ontologies, extracting subsets, filtering for selected axioms, running reasoners, and converting between file formats. Commands can be chained together to form powerful, repeatable workflows.

OWLTools and Oort CITE are the predecessors to ROBOT. OWLTools provides various functionality to support the Gene Ontology (GO) and many other ontology projects. In particular, it has supported the transition of GO and other projects from OBO format toward OWL format (Mungall *et al.*, 2014). Oort, the OBO Ontology Release Tool, is part of OWLTools and specifically designed as a command-line tool to help automate the transformation of ontology files from editing versions to publishable release files. Both are used extensively by BerkeleyBOP and other projects, often scripted by GNU Make files and run as part of Continuous Integration systems (Mungall *et al.*, 2012).

OWLTools and Oort provide more core functionality than ROBOT currently does. However, they have "evolved" over the years without an overarching design. The resulting tools are powerful, but somewhat difficult for new users to learn, and the code base was designed around certain assumptions that held true for OBO format ontologies but do not apply to OWL format ontologies. (TODO: This may be putting the point badly.)

With ROBOT we aim to create a more modular and extensible code base for developers, and provide a friendlier and more consistent interface for users. We plan to incrementally replace OWLTools and Oort where they are used, and promote the use of ROBOT as a standard tool for use by other OBO projects. In this paper we describe the design of ROBOT, it usage, and some of our future plans.

*To whom correspondence should be addressed: james@overton.ca

ROBOT is Open Source software, released under a BSD license, and is available on GitHub.[1] Although ROBOT is in early development, it is available for download as a JAR file and with executable scripts for Unix (including Mac OS X and Linux) and Windows platforms. We appreciate feedback, especially in the form of issues and pull-requests on GitHub.

2 OPERATIONS

ROBOT is a Java project built with Apache Maven[2], and divided into two modules:

- `robot-core` implements the basic operations that ROBOT provides
- `robot-command` implements the command-line interface

`robot-core` is designed to be used as a library from any language that runs on the Java Virtual Machine. Operations are classes that provide static methods for working with ontologies, usually building on functionality from OWLAPI (Horridge and Bechhofer, 2011). The currently implemented operations are:

- `DiffOperation`: find the differences between two ontologies
- `ExtractOperation`: extract a module from an ontology
- `FilterOperation`: remove ObjectProperties from an ontology
- `MergeOperation`: merge axioms from one or more ontologies and their imports into a single ontology
- `ReasonOperation`: use a reasoner to add axioms to an ontology

We plan to provide operations for MIREOT (Courtot *et al.*, 2011), for validating ontologies using SPARQL queries, and for modifying ontologies using SPARQL Update. `robot-core` also provides an `OntologyHelper` class with static utility methods, and an `IOHelper` class that manages state such as prefixes and namespaces.

3 COMMANDS AND CHAINS

The `robot-command` module implements ROBOT's command-line interface using the Apache Commons CLI library.[3]. Each command implements the Command interface, which has `main` and `execute` methods, as well as other common methods for getting the command name and usage information. Commands do

[1] `https://github.com/ontodev/robot`

[2] `https://maven.apache.org`

[3] `https://commons.apache.org/proper/commons-cli/`

not necessarily correspond to operations, but often do. The currently implemented commands are:

- `AnnotateCommand`: add annotations to an ontology
- `ConvertCommand`: save an ontology to another format: RDFXML, RDFOWL, Turtle, Manchester Syntax, OWL Functional Syntax, or OBO.
- `DiffCommand`: show the differences between two ontologies
- `ExportPrefixesCommand`: show the current prefixes (see below)
- `ExtractCommand`
- `FilterCommand`
- `MergeCommand`
- `ReasonCommand`

We plan to implement commands for MIREOT and SPARQL operations.

Commands can be called individually or via the `CommandManager` class. When used with a manager, commands can be *chained* into powerful workflows. When chains, commands communicate using a `CommandState` object that contains the current `OWLOntology` instance. Internally, command accepts a state, updates it, and returns it for the next command to use. Externally, each command is named by a verb and followed by zero or more options and their arguments. Figure 1 shows an example of a chain of commands for releasing an ontology.

```
robot \
  merge --input edit.owl \
  reason --reasoner ELK \
  annotate \
    --annotation-file annotations.ttl \
    --output results/example.owl \
  convert --output results/example.obo
```

Fig. 1. An example of a chain of commands

4 NAMES AND FORMATS

One source of problems with OWLTools has been the use of prefixed identifiers in OBO and OWL. With ROBOT we have made use of the JSON-LD standard[4] and the JSONLD-Java[5] library to provide a concise and effective method for specifying and resolving prefixes. A JSON-LD *context* is a JavaScript object with keys and values that specify how names and prefixes should be expanded, optionally with additional type information. The JSON-LD specification provides an algorithm for resolving names and prefixes to full IRIs.

ROBOT supports the use of JSON-LD for specifying prefixes to be used when loading and saving ontologies and sets of ontology terms. We also support loading ontology and annotation data from JSON-LD files, and from YAML files[6] when they have certain structure corresponding to JSON-LD.

[4] `http://www.w3.org/TR/json-ld/`

[5] `https://github.com/jsonld-java/jsonld-java`

[6] `http://yaml.org`

5 DOCUMENTATION AND TESTING

With ROBOT we aim to provide a user-friendly tool for ontology users and ontology developers. The project repository includes a README with installation instructions, a tutorial on command-line usage with example data files, and full JavaDoc documentation for the code itself. The command-line tool provides interactive help on commands. We plan to include ROBOT in the online OBO Tutorial CITE, replacing custom Java code with standard ROBOT commands.

In addition to a suite of unit tests, the tutorial itself serves as an integration test suite. A test harness extracts the example commands from the tutorial document, runs them, and compares the resulting files to the known-good example data files. Our unit tests and integration tests are run on every commit using Continuous Integration systems.

6 DEMONSTRATION

At ICBO we plan to demonstrate the installation and use of ROBOT for a range of ontology operations, building on the tutorial document. ROBOT will also be used in the OBO Tutorial session at the conference.

7 CONCLUSION

ROBOT is a user-friendly tool for working with ontologies at the command line, and for scripting ontology workflows. It is designed to replace Oort and many of the functions of OWLTools, and to address a wider audience of ontology developers. While still in early development, ROBOT can be used today, and we appreciate feedback via our GitHub repository.

ACKNOWLEDGEMENTS

TODO

REFERENCES

Courtot, M., Gibson, F., Lister, A. L., Malone, J., Schober, D., Brinkman, R. R., and Ruttenberg, A. (2011). Mireot: The minimum information to reference an external ontology term. *Applied Ontology*, **6**(1), 23–33.

Horridge, M. and Bechhofer, S. (2011). The owl api: A java api for owl ontologies. *Semantic Web*, **2**(1), 11–21.

Mungall, C. J., Dietze, H., Carbon, S., Ireland, A., Bauer, S., and Lewis, S. (2012). Continuous integration of open biological ontology libraries. *Bio-Ontologies*.

Mungall, C. J., Dietze, H., and Osumi-Sutherland, D. (2014). Use of owl within the gene ontology. *bioRxiv*.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., The OBI Consortium, Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., Scheuermann, R. H., Shah, N., Whetzel, P. L., and Lewis, S. (2007). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat Biotechnol*, **25**(11), 1251–1255.