GrafOnto user guide

Mateusz Bysiek

September 11, 2012

Contents

1 Introduction

1.1 License

GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

1.2 Qt framework

This application is written with help of Qt framework. It was compiled for Qt 4.8.1.

1.3 Two applications

GrafOnto is split into two separate applications: *GrafOntoGui* (from now on called *Gui*) and *GrafOntoConsole* (from now on called *Console*). The first one runs in window, and the second in terminal/command-line.

However, the difference between them is only in the presentation layer i.e. the engine that performs operations on the ontology is the same.

1.4 Language

Operation of the application is based on text commands given in language similar to SQL. From now on, let's call it *Lang*. It is very simple, below is the basic example:

```
add category (thing, situation) # define two categories, called thing and situation
add thing (writing,book,dictionary,title,index) # create five new things
add situation (is , has) # create two new situations: 'is' and 'has'
dictionary is : book # create a link: (dictionary,is)->(book)
book is: writing
book has: title
dictionary has: index
set situation relation # indicate that category called 'situation' is a relation
# this will cause statements like (dictionary,is)->(book) be seen
# more like (dictionary)--is-->(book) by the application
set is transitive # indicate that situation 'is' applied to a chain of elements
# like in our example (dictonary,is)->(book); (book,is)-->(writing)
# or more simply: dictonary--is-->book; book--is-->writing
# will be treated like: dictonary--is-->book--is-->writing by the application
find all that is: writing # should find book and dictionary
```

This is the introduction, therefore I will not provide a list of all possible commands right now. Full specification of Lang is provided later on. Inconsistency in whitespace characters is deliberate, it is to show that they do not matter and the script will work just as well no matter how many spaces you will insert between the terms.

2 Known issues

Yet unsolved issues.

- Space after colon is mandatory.
- transitivity does not "transfer" other connections of the given element
- "find all that ..." does not always work properly with transitivity
- -auto command-line option performs well only when there is only one further argument - please use quotes!

3 Feature checklist

This project still remains in development phase, this list provides overview of features that are really present.

3.1 Implemented features

Features listed below are fully implemented:

- add category/element (single and multiple)
- find all

3.2 Unimplemented features

Features listed below are not implemented at all or in a degree that is too low to do anything useful.

- deleting
- Saving/loading ontologies from files

4 Script language specification

4.1 Ontology modification

4.1.1 add

Used to enlarge set of categories and elements.

`add category <category_name>`

Unless category with a given name exists, it is defined. Category cannot have name “category”.

`add <category_name> <element_name>`

If a category exists, and element of a given name does not, new element is created.

4.1.2 set

Used to modify category/element properties. List of properties of categories is different than list of properties of elements.

```
set <category_name> relation
set <category_name> not relation
```

Sets a given category to be (or not to be) a relation. If the category already is (or already is not) a relation, this command does nothing.

```
set <element_name> transitive
set <element_name> not transitive
```

Sets a given element to be (or not to be) transitive. If the element already is (or already is not) transitive, this command does nothing.

4.2 Ontology persistence

4.2.1 save

`save <filename>`

Saves the current ontology to a XML file. Format of the file is described later on.

4.2.2 load

`load <filename>`

Loads the ontology from a XML file, completely erasing and the current ontology.

4.3 Ontology querying

4.3.1 find

`find all`

Displays all categories, elements, cells and statements currently present in the ontology.

`find all that <partial_statement>`

Searches for all matching statements in the ontology. Matching statement is defined as a statement that has right hand side identical to the given one, and left hand side being a superset of a given LHS.

5 Command-line options and arguments

GrafOnto supports the following arguments:

`-demo` - Supported in both Gui and Console, causes application to load an example ontology at start-up, before execution of any other operations.

`-auto` - Supported only in Console, causes application to interpret all further arguments as commands in Lang. Since commands are separated by spaces, and spaces are a must in all Lang commands, all further commands are first joined into one long text, and then are split where semicolons (`;`).

`-debug` - Supported only by debug builds. Causes debug output to be displayed. Information about what build are you running can be displayed in Gui via *About* menu option, and in Console it is shown at start of the program.

6 XML ontology container specification

This is a specification of XML files that are read/written via `load`/`save` commands.

```
<ontology>
        <categories>
                <!-- each category has unique id -->
                <category id="#">name</category>
                <category id="#">name2</category>
                ...
        </categories>
        <elements>
                <!-- each element has unique id, and is given a category id;
                     it also has transitivity (1) or it does not have it (0) -->
                <element id="#" cat_id="#" transitive="0">name</element>
                <element id="#" cat_id="#" transitive="0">name2</element>
                <element id="#" cat_id="#" transitive="1">name3</element>
                ...
        </elements>
        <cells>
                <!-- each cell has unique id, and contains a list
                     of elements (in abbreviated form) -->
                <cell id="#">
                        <elem id="#" />
                        <elem id="#" />
                        <elem id="#" />
                        ...
                </cell>
                <cell id="#">
                        <elem id="#" />
                        <elem id="#" />
                        ...
                </cell>
                ...
        </cells>
        <statements>
                <!-- each statement has unique id, and is given an id of cell
                     on the left side, and id of a cell on the right side -->
                <statement id="#" left="#" right="#" />
        </statements>
</ontology>
```