CPS Ontology System Documentation

NIST, SJU, NMSU
Written By Matt Bundas (bundasma@nmsu.edu)

Table of Contents

Summary

The purpose of the CPS Ontology System is to provide users with applications and supplemental software to make working with CPS Ontologies more, easy, efficient, and effective. It allows users to easily edit an ontology without having to manually edit a .owl file, run queries to learn about the ontology, and finally visualize the structure and state of the CPS in a meaningful and easy to understand way. At its core, there are three major components, the CPS Ontology Editor, ASP Reasoner, and Tableau visualization. The Ontology Editor is a python-written user interface which allows a user to add/remove/edit an existing ontology's individuals and relations between these individuals, and then output the ontology to a .owl file. The ASP Reasoner is a java-written program which uses Answer Set and Logic Programming to analyze a given ontology through queries, which answer specific questions about the ontology. The Tableau visualization provides both a visually pleasing and fundamentally useful visualization of an ontology after a query, showing both the structure and status of the ontology. The Ontology Editor and ASP reasoner are connected by the .owl file outputted by the Editor, and ASP reasoner and Tableau visualization are connected by a csv file outputted by the ASP Reasoner, which can be used as the data source in Tableau. Documentation with more detail on each component is also provided.

Requirements

The requirements of the system of a whole are that of the combination of all of its components. Details for each component's requirements are in their separate documentations, but in summary you will need a working java, Eclipse, python, and Tableau installations and the following python packages:

Tkinter - https://docs.python.org/3/library/tk.html
Owlready2 - https://pythonhosted.org/Owlready2/
Networkx - https://pypi.org/project/networkx/
Pygraphviz - http://pygraphviz.github.io/documentation/pygraphviz-1.5/index.html
Pydotplus - https://pypi.org/project/pydotplus/
Parse - https://pypi.org/project/parse/
Argparse - https://docs.python.org/3/library/argparse.html

All files related to the system are hosted on our github page:
https://github.com/thanhnh-infinity/Research_CPS

At the moment, all files for the editor are in the /interface/testing/ directory, files for the ASP reasoner are in /src/asklab/, files for the CSVExporter are in /CSVExport/, and files for the Tableau visualization are in /Tableau/. I recommend cloning the GitHub repo, and working within that directory so all directories and files are in the same place relative to how they are developed.

Usage

The CPS Ontology system is used to easily create/modify ontologies, run queries to analyze the CPS, and understand the ontology with an easy to interpret visualization in Tableau. To make use of the system's function at a high level, all components must be used, the editor, ASP solver, and Tableau visualization. See their documentations for a more in depth overview of how they work, but below is information on how to use each component in the context of the system as a whole.

The usage of the system involves:

CPS Ontology Editor - opening the editor, loading an ontology, editing the ontology, outputting the ontology
ASP Solver - launching the ASP solver with the new ontology, running a query of interest, outputting the query as a CSV,
Tableau Visualization - opening the Tableau workbook containing the desired visualization, and then finally connecting the new CSV file to be the data source of Tableau.

CPS Ontology Editor

See the CPS Ontology Editor documentation for more information on the Editor, but here are the essentials for using the editor on a system-wide level.

Launching Editor

To launch the editor, run the python script OntologyGUI.py. This file is found in the Github repo in the /interface/testing/ directory.

Inputting/Outputting Ontology

Inputting and outputting ontologies is performed using the widgets in the top left of the interface, by entering the desired paths in the entries and clicking the corresponding buttons. The paths are relative to where OntologyGUI.py is located. You must load a base ontology before loading the accompanying application ontology. At the moment, you cannot unload ontologies from the editor, you must close it and restart your kernel if necessary. Once an ontology is loaded, it will be displayed in tree-form in the editor.

Editing Ontology

The ontology editor supports the creation, deletion, and editing of individuals like aspects, concerns and properties, and the relations between these individuals. These editing operations are done by clicking on nodes in the ontology tree, or clicking buttons near the tree like the relations and dependencies buttons. Every time an operation is performed, the modification will be shown in the ontology tree.

New Individuals

New individuals are created by clicking on the tree-node you want the new individual to be associated with. For example if you want to add a new subconcern, click on the desired parent node, type in the desired subconcern name, and click add subconcern.

Edit Name

Individual names are edited by clicking on a tree-node, typing in a new name, and clicking the Edit Name button.

Deleting Individuals

Individuals are deleted by clicking on a tree-node, then clicking the Delete button. If the node is a leaf node, it will be deleted immediately, otherwise you will be presented with options for what you want to do with its children, such as deleting all of the selected node’s children, or just the selected node itself.

Relations

Relations are added/deleting by clicking the Relations button, then left clicking on a tree-node to select the parent, left clicking on another node to select the child, then clicking the button corresponding to the operation you'd like to perform. The editor will automatically detect the relation type depending on the two nodes selected.

Dependencies
Dependencies are added by clicking on the Dependencies button which will open up a new window to enter the dependency. The dependency is framed in an IF (properties) THEN (concern) format. Left click on tree nodes to add them with a positive polarity, right click to add them with negative polarity to the rule. Click the IF/THEN buttons to choose which side of the rule you are editing. You can also manually type in the rule.

Remove Relationless
While editing the ontology, it's possible to end up with nodes that have no relation, and thus play no role in the ontology, and interfere with the reasoning process. To remove all of these at once, click the Remove Relationless button.

ASP Solver

Launching Solver

Currently the solver is designed to work in Eclipse. To launch the solver, open the CPS Ontology directory tree as a java project in Eclipse, and run QueryPicker.java. More details on how to do this is in the ASP Solver documentation. The program will make use of the ontologies specified in the code.

Running a Query

To run a query, launch the ASP Solver, and click the button corresponding to the desired query. This will reason with the loaded ontology, and print the results according to the selected query.

Outputting Query as CSV

After running a query, output it to CSV by clicking the "Output CSV" button. A message will be shown showing whether the output was successful or not. If successful, it will save the csv file named out.csv in the /Integration/CSV directory. Customization of this filepath will hopefully be added in the future.

Tableau Visualization

The Tableau Visualization is stored in a Tableau workbook, currently Large_use_case_sunburst.twb. This workbook allows for the creation of the visualization, given it has a data source which it is able to work with. The data source needs to be a csv file outputted by the ASP Solver, or one of the same format.

Launching Tableau Visualization

To open the Tableau Visualization, open Tableau, then open the Large_use_case_sunburst.twb workbook using File-Open.

Connecting/Refreshing Data Source

To connect a new Data source, navigate to the Data Source tab, in the top left near Connections, click add, click text file, then find the desired file. If there already existed a data source prior to adding, you may need to remove the original source, so Tableau uses the new one. The source must be a csv or excel file. To refresh an already existing source, such as after you outputted a new csv after running a query (but the same data source filepath), just click the refresh button above the Connections section in the Data Source tab, it is a button that is two arrows in the shape of a circle.

Viewing Visualization

Once a data source is connected or refreshed, you can view the different visualizations by clicking the tabs near the bottom and navigate to the desired visualization. Each sheet/dashboard will display a different visualization. Clicking on the legend will highlight/display different aspect trees.

Use of Components As System

Using the components of the CPS Ontology System as a whole involves using the three individual components, which are bridged using input/output files from other components. The general flow is that a user can open the Ontology Editor, edit the ontology, output it, reason with it in the ASP Solver with a query, output the query as a CSV for the Tableau visualization to use as a data source. These steps are all outlined above, and can be done assuming all requirements are met in the Requirements section.

Design

The CPS Ontology System functions by having applications which provide a distinct function to the user, and are connected to each other through files which other applications input/output. The Ontology Editor loads an existing .owl file and outputs an edited .owl file, this .owl file can be loaded by the ASP Solver, reasoned with via query, which can be outputted to a csv to be used as a data source in the Tableau visualization.

CPS Ontology Editor

The CPS Ontology Editor is a python-written GUI which allows the user to load an ontology, edit it, and output it as an .owl file. It makes use of several packages to make things easier. It uses tkinter to handle the GUI side of things, like creating frames and buttons to call functions. Owlready2 is used to interact with the ontology, allowing the program to load an ontology from an .owl file, and after that treat the ontology as a python object instead of a text file. Owlready2 has a lot of functions which make editing and ultimately outputting the ontology easy. To graph the oncology, networkx is used to create the visualization itself, and pygraphviz is used to create the tree structure. At a high level, OntologyGUI.py is the driver script which sets up the GUI and references other classes. Ontologies as the GUI uses them are stored in their own python classes, owlBase and owlApplication, these have references to the owlready ontology objects and also essentially encodes the ontology based off the owlready objects using owlNodes, which represent an individual in the ontology. owlNodes have data members like type, name, parents, children etc. OntologyGUI works with owlNodes and owlBase/Application, rather than directly with owlready objects. The graph is stored in the owlGraph class, where it has references to the owlBase/Applications which make it up, and lists of all of their relevant owlNodes. It looks at these nodes to add nodes, edges and labels correctly based on the information in their owlNode object. It represents the first step in the end-to-end system workflow, and connects to the ASP Reasoner by outputting an owl file for it to use.

ASP Solver

The ASP Solver is a java-written program and makes use of Answer Set Programming to inquire about and reason with a CPS Ontology.