Source URL: https://github.com/protegeproject/cellfie-plugin/wiki/Grocery-Tutorial

cellfie-tutorial.md

This tutorial will give you a quick introduction of Cellfie plugin. We are going to guide you step-by-step through the process of building an ontology from an Excel workbook about grocery items.

In this short tutorial you will:

- learn on how to work with the plugin UI,
- understand some basic MappingMaster syntax and directives to create the transformation rules,
- generate the OWL axioms from the Excel spreadsheet and import them into an OWL ontology.

The estimated completion time: 30-45 minutes.

Setup and Installation

Cellfie is part of Protégé 5.0 distribution so no installation is required. However, you might need to upgrade its version through the Protégé plugin auto-update, or by checking it manually at File > Check for plugins....

There are three files to download:

- Grocery ontology -- alternative URL: https://git.io/vPhdf,
- Excel workbook -- alternative URL: https://git.io/vPhdP
- (optional) Transformation rules -- alternative URL: https://git.io/vPjcs

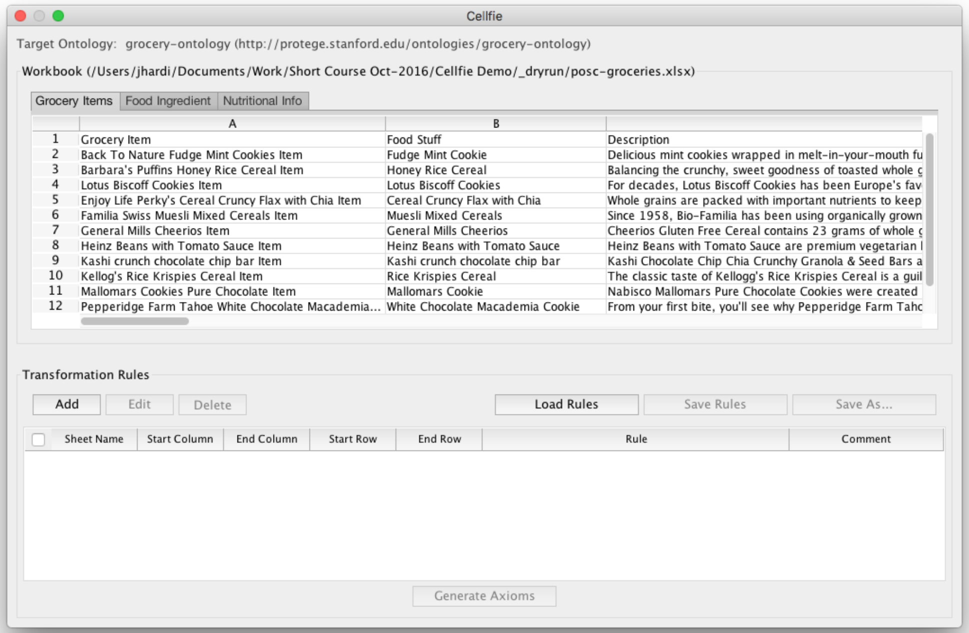
(You might need to use "Save Link As..." in your browser if the download doesn't happen immediately)

The ontology file contains a baseline for the grocery items model, the Excel file contains our dataset and the rule file contains all the transformation rules used in this tutorial. Feel free to review those files before continuing.

STEP 1 - Starting Cellfie

Open the grocery ontology using Protégé and make sure it is currently the active ontology.

Select Tools > Create axioms from Excel workbook... to open the Cellfie window. Load the Excel file in the prompted dialog and Cellfie will appear with the content of the Excel file.



Target Ontology: grocery-ontology (http://protege.stanford.edu/ontologies/grocery-ontology)

Workbook (/Users/jhardi/Documents/Work/Short Course Oct-2016/Cellfie Demo/_dryrun/posc-groceries.xlsx)

	A	B	
1	Grocery Item	Food Stuff	Description
2	Back To Nature Fudge Mint Cookies Item	Fudge Mint Cookie	Delicious mint cookies wrapped in melt-in-your-mouth fu
3	Barbara's Puffins Honey Rice Cereal Item	Honey Rice Cereal	Balancing the crunchy, sweet goodness of toasted whole g
4	Lotus Biscoff Cookies Item	Lotus Biscoff Cookies	For decades, Lotus Biscoff Cookies has been Europe's fav
5	Enjoy Life Perky's Cereal Cruncy Flax with Chia Item	Cereal Cruncy Flax with Chia	Whole grains are packed with important nutrients to keep
6	Familia Swiss Muesli Mixed Cereals Item	Muesli Mixed Cereals	Since 1958, Bio-Familia has been using organically grown
7	General Mills Cheerios Item	General Mills Cheerios	Cheerios Gluten Free Cereal contains 23 grams of whole g
8	Heinz Beans with Tomato Sauce Item	Heinz Beans with Tomato Sauce	Heinz Beans with Tomato Sauce are premium vegetarian l
9	Kashi crunch chocolate chip bar Item	Kashi crunch chocolate chip bar	Kashi Chocolate Chip Chia Crunchy Granola & Seed Bars a
10	Kellog's Rice Krispies Cereal Item	Rice Krispies Cereal	The classic taste of Kellogg's Rice Krispies Cereal is a guil
11	Mallomars Cookies Pure Chocolate Item	Mallomars Cookie	Nabisco Mallomars Pure Chocolate Cookies were created
12	Pepperidge Farm Tahoe White Chocolate Macademia...	White Chocolate Macademia Cookie	From your first bite, you'll see why Pepperidge Farm Tahc

Working with Cellfie is about creating the *Transformation Rules* that define the mapping between the cell values and the OWL axiom structure. The rule is written in Manchester Syntax used commonly in Protégé. The following steps will show you how to create these rules for different types of OWL axiom.

STEP 2 - Creating Class Declaration Axiom

In this step, we are going to create a simple class declaration axiom rule from the "Grocery Items" sheet that will make each cell in column A an OWL class.

To create a new transformation rule, select the Add button and fill out all the parameters as below:

- Sheet name: `Grocery Item`
- Start column: `A`
- End column: `A`
- Start row: `2`
- End row: `+`

The parameters above tells a cell range selection from A2 to A14 in the "Grocery Item" sheet. Notice the wildcard character '+' at the "End row" parameter. This wildcard will tell Cellfie to use the last row index in the spreadsheet. It works similarly for the "End column" parameter, i.e., to use the last column index in the spreadsheet.

> Note: Starting from version 2.1.0, you can use the interactive mode to fill out the input parameters by creating a cell range selection in the Workbook view.

Next, type the expression below inside the "Rule" field and select OK to save it.

```
Class: @A*
```

> Note: The `@A*` is called the cell reference notation. The notation is composed of 3 elements: `@` + `column-name` + `row-number`. Use the wildcard asterisk `*` to mean "Select All". In the expression above, the asterisk at the `row-number` position will select all rows within a range between the "Start row" and the "End row" parameters, thus, `@A*` will refer to cells A2, A3, ..., A13, A14.

Select "Generate Axioms" and a preview window will appear with a list of generated axioms. Close the preview window by selecting the "Cancel" button.

STEP 3 - Creating Sub-Class Axiom

In this step, we are going to create subclass axioms that will make all cells in column A subclasses of `GroceryItem`.

Select our previous transformation rule and click the "Edit" button. Modify the rule expression as below and select OK to save it.

```
Class: @A*
   SubClassOf: GroceryItem
```

> Note: When an entity name is explicitly mentioned in the transformation rule, Cellfie requires the entity to be present in the ontology. In the example above, the class `GroceryItem` is already part of the Grocery ontology.

Select "Generate Axioms" and see the output preview.

> Exercise: Repeat these same steps to create another subclass axiom that uses column B and the cells are subclasses of `FoodStuff`.

STEP 4 - Creating Some Value Restriction Axiom

In this step, we are going to create some value restriction axiom that will make all cells in column A subclasses of `containsFoodStuff` some *FoodStuff*, where the *FoodStuff* are the values in column B.

Add an extra rule in the SubClassOf axioms as below:

```
Class: @A*
   SubClassOf: GroceryItem, containsFoodStuff some @B*
```

Select "Generate Axioms" and see the output preview.

> Exercise: Go to the "Recipe" sheet and create two SubClassOf axiom rules. Hint: the rule contains `FoodStuff` and `hasIngredient`.

STEP 5 - Creating Annotation Assertion Axiom

In this step, we are going to add two annotations in each of our class declaration axiom, i.e., `rdfs:comment` and `foaf:depiction`.

Add two extra lines for the annotation assertions such that our previous subclass rule becomes as follow:

```
Class: @A*
   SubClassOf: GroceryItem, containsFoodStuff some @B*
   Annotations: rdfs:comment @C*,
                foaf:depiction @D*(IRI)
```

> Note: By default, Cellfie considers the annotation assertion value as a string. To change the type into an IRI, append the `IRI` directive after the cell reference notation (see the `foaf:depiction` example above)

We could also add a detail about the language used in the annotation by appending the `xml:lang` directive like so:

```
Class: @A*
   SubClassOf: GroceryItem, containsFoodStuff some @B*
   Annotations: rdfs:comment @C*(xml:lang="en"),
                foaf:depiction @D*(IRI)
```

STEP 6 - Creating Class Assertion Axiom

In this step, we are going to create some class assertions for the `NutritionalInformation` class from the "Nutritional Info" sheet.

Add a new rule and fill out the input parameters as below:

- Sheet name: `Nutritional Info`
- Start column: `A`
- End column: `A`
- Start row: `2`
- End row: `+`

```
Individual: @A*(mm:hashEncode)
   Types: NutritionalInformation
```

> Note: Every OWL entity requires a unique identifier. Your data model in the spreadsheet might not give you the cells for creating the identifiers. Cellfie provides two options to generate a unique identifier by drawing it out either from the cell value or the cell address. The `mm:hashEncode` will generate the hash string of the cell value and the `mm:uuidEncode` will generate a UUID based on the full cell address.

STEP 7 - Creating Property Assertion Axiom

Next, we now are going to add some property assertions to our class assertion.

Add four extra lines for our property assertions such that our previous class assertion rule becomes as below:

```
Individual: @A*(mm:hashEncode)
   Types: NutritionalInformation
   Facts: productName @A*,
          hasTotalFat @B*,
          hasSaturatedFat @C*,
          hasSugar @I*
```

Select "Generate Axioms" and see the output preview.

STEP 8 - Assigning Datatypes

Notice the string type in all property assertion values. Remember, Cellfie considers assertion values as a string by default. You can change the type by explicitly specifying the datatype in the rule.

> Note: Cellfie supports 13 standard datatypes which are:
>
> - `xsd:string` (by default)
> - `xsd:boolean`
> - `xsd:double`
> - `xsd:float`
> - `xsd:long`
> - `xsd:integer`
> - `xsd:short`
> - `xsd:byte`
> - `xsd:decimal`
> - `xsd:dateTime`
> - `xsd:date`
> - `xsd:time`
> - `rdf:PlainLiteral` (automatically assumed if you use `xml:lang` directive)

Modify the previous property assertions by assigning the proper datatype for our nutritional facts as below:

```
Individual: @A*(mm:hashEncode)
   Types: NutritionalInformation
   Facts: productName @A*,
          hasTotalFat @B*(xsd:decimal),
          hasSaturatedFat @C*(xsd:decimal),
          hasSugar @I*(xsd:integer)
```

Select "Generate Axioms" and see the output preview.

STEP 9 - Using Functions

In this step, we are going to use some of the built-in functions that are useful for doing a bit of string manipulation of the cell values.

We are going to create a custom numeric presentation by using the `mm:decimalFormat` function. Let's have a nice decimal format for the total fat and the saturated fat using that function as follow:

```
Individual: @A*(mm:hashEncode)
   Types: NutritionalInformation
   Facts: productName @A*,
          hasTotalFat @B*(xsd:decimal mm:decimalFormat("##0.00")),
          hasSaturatedFat @C*(xsd:decimal mm:decimalFormat("##0.00")),
          hasSugar @I*(xsd:integer)
```

Next, we can do also a text extraction using a regular expression (regex) through the `mm:capturing` function. Let's add the amount of sodium by capturing its value without the measurement unit from column F as follow:

```
Individual: @A*(mm:hashEncode)
   Types: NutritionalInformation
   Facts: productName @A*,
          hasTotalFat @B*(xsd:decimal mm:decimalFormat("##0.00")),
          hasSaturatedFat @C*(xsd:decimal mm:decimalFormat("##0.00")),
          hasSugar @I*(xsd:integer),
          hasSodium @F*(xsd:integer mm:capturing("([0-9]+)"))
```

Select "Generate Axioms" and see the output preview.

STEP 10 - Import Axioms to Ontology

The options to import axioms are available in the "Generate Axioms" window.

> Note: Starting from version 2.1.0, you have the option to pick which transformation rules for generating the axioms.

There are two options to import axioms to an ontology:

- Add to current ontology - Cellfie will insert all the generated axioms to the currently active ontology
- Add to a new ontology - Cellfie will create a new ontology and insert all the generated axioms. The new ontology will have an import axiom that will point to the original ontology.

Select the Add to current ontology button and see the new axioms in the Grocery ontology.

Conclusion

MappingMaster provides the functionality to import bulk spreadsheet data to OWL ontology constructs. For more information please refer to the MappingMaster wiki to get the complete syntax and functions guide.

Help and Support

If you have questions about MappingMaster or Cellfie, please address them to us via Protégé User Support mailing list.