SBOL-OWL Supplementary

SBOL-OWL enabled semantic queries vs graph-based queries

Supplementary Table 1: Report inconsistent use of properties between valid SBOL entities.

<table>
<tr><th>OWL-DL & SBOL-OWL</th><th>SPARQL without SBOL-OWL</th></tr>
<tr><td>No need for a query. Reasoners use axioms to report inconsistencies for all properties (example data: Table 2.11).</td><td>SPARQL queries can be used to check for correct use of properties. However, these queries may have to be executed for each property that is being validated. The example query below lists <code>ModuleDefinitions</code> that incorrectly use the <code>model</code> property. If this property does not refer to a <code>Model</code> entity, the <code>ModuleDefinition</code> is reported to be inconsistent (example data: Table 2.11).
<pre>
PREFIX sbol:<http://sbols.org/v2#>
SELECT ?MD WHERE {
 ?MD sbol:model ?Model.
 FILTER NOT EXISTS {?Model a sbol:Model }
}
</pre></td></tr>
</table>

Supplementary Table 2: Return all composite parts that contain child parts (Figure 6).

<table>
<tr><th>OWL-DL & SBOL-OWL</th><th>SPARQL without SBOL-OWL</th></tr>
<tr><td>Example data: Table 2.3
<pre>
DNA and component some Component
</pre></td><td>Example data: Table 2.2
<pre>
PREFIX sbol: <http://sbols.org/v2#>
PREFIX biopax: <http://www.biopax.org/release/
    biopax-level3.owl#>
SELECT DISTINCT ?X WHERE {
 ?X a sbol:ComponentDefinition ;
     sbol:component ?Component ;
     sbol:type biopax:DnaRegion .
}
</pre></td></tr>
</table>

Supplementary Table 3: Return all composite parts that directly contain promoters (Figure 6).

OWL-DL & SBOL-OWL

Example data: Table 2.3.

```
DNA and component some (
    definition some Promoter)
```

SPARQL without SBOL-OWL

Example data: Table 2.2

```
PREFIX sbol: <http://sbols.org/v2#>
PREFIX biopax: <http://www.biopax.org/release/
    biopax-level3.owl#>
PREFIX so: <http://identifiers.org/so/>
SELECT DISTINCT ?X WHERE {
?X a sbol:ComponentDefinition ;
  sbol:component ?Component ;
  sbol:type biopax:DnaRegion .
?Component a sbol:Component ;
  sbol:definition ?Y .
?Y a sbol:ComponentDefinition ;
  sbol:type biopax:DnaRegion ;
  sbol:role so:SO:0000167 .
}
```

Supplementary Table 4: Return all composite parts that directly or indirectly contain promoters (Figure 6).

OWL-DL & SBOL-OWL

The `PromoterContainer` query is defined as a class that refers to itself (example data: Table 2.13).

```
Class: PromoterContainer
ComponentDefinition and component
    some (definition some (
    Promoter or PromoterContainer
    ))
```

SPARQL without SBOL-OWL

This requires programmatically executing SPARQL queries and implementing an algorithm, for example using the Java programming language, to recursively fetch the parts that contain promoters. The SPARQL query below can be used to return a list of parent, child, child role tuples. Each item in the list can then be checked if the role matches the `SO:0000167` promoter term. Then all the parents of the child component can be programmatically retrieved using a recursive algorithm (example data: Table 2.2).

```
PREFIX sbol: <http://sbols.org/v2#>
PREFIX biopax: <http://www.biopax.org/release/
    biopax-level3.owl#>
PREFIX so: <http://identifiers.org/so/>
SELECT DISTINCT ?X ?Y ?childrole WHERE {
?X a sbol:ComponentDefinition ;
  sbol:component ?Component ;
  sbol:type biopax:DnaRegion .
 ?Component a sbol:Component ;
          sbol:definition ?Y.
?Y a sbol:ComponentDefinition ;
  sbol:type biopax:DnaRegion ;
  sbol:role ?childrole .
}
```

Supplementary Table 5: Return all of the interactions used in a design. The list should include all interactions in a module and those that are included in submodules directly or indirectly (Figure 11).

<table>
<tr><th>OWL-DL & SBOL-OWL</th><th>SPARQL without SBOL-OWL</th></tr>
<tr><td>

The circuit_0x78_Interaction class is used to identify all interactions of the circuit_0x78_environment_md entity. Any child design that is hierarchically part of this entity is also taken into consideration (example data: Table 2.5).

```
Class: circuit_0x78_Module
ModuleDefinition and
 (isDefinitionOf some (Module
  and (
   (isModuleOf some
       circuit_0x78_Module)
   or
   (isModuleOf value
       circuit_0x78_environment_md
       ))
  ))

Class: circuit_0x78_Interaction:
Interaction and (isInteractionOf
    some circuit_0x78_Module)
```

</td><td>

This requires programmatically executing SPARQL queries multiple times. The SPARQL query in A is initially used to retrieve information about interactions. The query in B is then used to retrieve the submodules in order to recursively fetch the interactions for the next iteration. The execution of SPARQL queries should be stopped when an entity does not include a submodule (example data: Table 2.4).

A)

```
PREFIX sbol: <http://sbols.org/v2#>
SELECT ?Interaction
WHERE {
 <MODULE_DEFINITION_URI> a sbol:ModuleDefinition ;
 sbol:interaction ?Interaction .
}
```

B)

```
PREFIX sbol: <http://sbols.org/v2#>
SELECT ?ModuleDefinition
WHERE {
 <MODULE_DEFINITION_URI> a sbol:ModuleDefinition ;
  sbol:module ?Module .

  ?Module a sbol:Module ;
    sbol:definition ?ModuleDefinition .
}
```

</td></tr>
</table>

Supplementary Table 6: Return direct or indirect types of a part. As an example, ptetR in Figure 6 is used.

<table>
<tr><th>OWL-DL & SBOL-OWL</th><th>SPARQL without SBOL-OWL</th></tr>
<tr><td>

No need for a query. The direct type (`ComponentDefinition`) and indirect types (`TopLevel` and `Identified`) are available via reasoners. (example data: Table 2.3)

</td><td>

The transitive query only returns the direct type (example data: Table 2.2). the query only succeeds to return all types when SBOL-OWL is applied (example data: Table 2.3, RDFS reasoning example: Table 2.1).

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>
PREFIX sbol: <http://sbols.org/v2#>
PREFIX pr: <http://parts.igem.org/>
SELECT ?type WHERE {
 ?dtype rdfs:subClassOf* ?type .
 pr:ptetR a ?dtype.
}
```

</td></tr>
</table>

Supplementary Table 7: Return all TopLevel entities.

OWL-DL & SBOL-OWL	SPARQL without SBOL-OWL

No need for a query. All entities of the TopLevel class are directly available via reasoners (example data: Table 2.5). Compatible with future SBOL specification changes, when new TopLevel entities are introduced to SBOL in the future.	Succeeds when all types inheriting from TopLevel are included in the SPARQL query. The query needs to be upgraded when new TopLevel entities are introduced to SBOL in the future (example data: Table 2.4).
	`PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
	`PREFIX sbol: <http://sbols.org/v2#>`
	`SELECT ?entity  WHERE {`
	`{?entity a sbol:ComponentDefinition .} UNION`
	`{?entity a sbol:Sequence .} UNION`
	`{?entity a sbol:ModuleDefinition .} UNION`
	`{?entity a sbol:Attachment .} UNION`
	`{?entity a sbol:Model .} UNION`
	`{?entity a sbol:CombinatorialDerivation .} UNION`
	`{?entity a sbol:Attachment .} UNION`
	`{?entity a sbol:Implementation .}`
	`}`