SBML2SHACL

Edoardo De Matteis
1746561

11 agosto 2020

Indice

1 Introduzione

SBML è un formato basato su XML per definire conoscenza medica e biochimica, più che di linguaggio si parla di lingua franca dal momento che si pone come tale, risolvendo il problema causato dall'eterogeneità di standard tra software in ambito biomedico.

SHACL invece è uno standard del W3C per la verifica di grafi rispetto a dei vincoli definiti, questi grafi sono rappresentati in qualsiasi formato RDF, ad esempio Turtle; Il Resource Description Network (RDF) è necessario per la codifica e la manipolazione di metadati e consente la modellazione di informazioni come risorse web, dal momento che si fa uso di logiche descrittive tali informazioni sono codificate sotto forma di triple `<subject> <predicate> <object>` equivalente in logica del primo ordine ad una formula ove subject e object sono due termini ground e predicate un predicato binario.

Dato un modello SHACL il linguaggio di query che permette di interrogarlo è SPARQL; RDF, SHACL e SPARQL sono standard del World Wide Web Consortium (W3C).

L'obiettivo di questo progetto, sotto la guida del Professor Tronci, è quello di convertire automaticamente codice SBML in SHACL. A tal fine il problema è stato diviso in tre fasi:

1. Selezionare un sottoinsieme di costrutti in SBML e modellarlo in SHACL.
2. Scrivere un parser che traduca una specifica SBML in SHACL.
3. Scrivere un parser che traduca una specifica SHACL in SBML.

2 Modellazione

Nella prima fase si è scelto un sottoinsieme di SBML 3.2 e la totalità dei costrutti aggiuntivi di extended SBML, nella tabella 1 vengono descritti i principali costrutti, sono state volontariamente omesse entità quali `listOf*` il cui significato è intuitivo e avrebbero solo reso la tabella meno leggibile, in ogni caso è possibile consultare ulteriormente il diagramma `diagram.png` e il file `shapes.ttl`, in qest'ultimo è scritta la definizione del modello con i vincoli che dovranno essere rispettati dai file di output del parser; è possibile verificare la consistenza del file di output rispetto al modello tramite lo script `shacl_verifier.py`. Per entrambi i file si è scelto di utilizzare il formato Turtle

Extended SBML introduce la definizione di una gerarchia, è possibile definire ad esempio l'annidamento "cellula-nucleo-dna", aggiunge inoltre la possibilità di definire modelli e poterli importare ed esportare, favorendo il riuso di codice.

Entità	**Descrizione**
SBML 3.2	

SBase	Classe astratta che rappresenta un tipo - ogni classe definisce un tipo - e ogni nodo sarà sottoclasse di SBase, poiché non esistono attributi di tipo SBase a questa classe si riserva un trattamento differente rispetto alle classi che rappresentano tipi composti di attributi quali ID, SId, ecc.. le quali sono omesse dal diagramma ma sono chiaramente presenti nel file Turtle.
Sbml	Ogni file SBML ha un'etichetta con tag `sbml`, grazie a questo nodo è possibile costruire grafi SHACL composti da multipli modelli SBML in input dato che saranno sempre radicati in questo nodo, è importante specificare che questo modello non sarà legale in SBML 3.2 né extended. Il costrutto Sbml in linea teorica dovrebbe poter ammettere anche attributi oltre quelli noti, siccome però in SHACL ho necessità di conoscerne il tipo ho tolto questa possibilità.
Model	Rappresenta il modello, se ne può avere più di uno come detto sopra nonostante SBML non lo permetta.
Unit	Definisce un'unità di misura definita dall'utente, in SBML sono definite delle unità base (i.e. le unità del SI e altre scelte dagli sviluppatori di SBML) e combinandole opportunamente tra loro è possibile definire nuove unità di misura (e.g l'accelerazione $\frac{m}{s^2}$ altro non è che $m^1 s^{-2}$). Nel modello in `shapes.ttl` le unità base (kind) sono trattate come normalissime unità di misura, dal momento che in SBML non è concesso avere come attributo kind un'unità di misura che non sia tale questa rappresentazione non genera problemi.
Compartment	Rappresenta un insieme di entità biologiche.
Species	Rappresentano delle entità biologiche.

Parameter	In SBML si possono definire parametri sia locali che globali, il modello presentato in queso progetto non presenta località quindi i parametri sono a priori globali, quindi Model avrà come attributo `ListOfParameters` con moltiplicità $[0, n]$ piuttosto che $[0, 1]$.
Extended SBML	
ExternalModelDefinition	Necessario per importare modelli esterni e introdurli nel proprio.
ModelDefinition	Data la presenza di una gerarchia questo costrutto dà la definizione di un modello.
Submodel	L'istanza di una ModelDefinition è rappresentata da un Submodel e si ha effettivamente un modello dentro ad un altro modello. Durante la fase di test per extended SBML non ho usato esempi scaricati da Biomodels perché la gerarchia non veniva rappresentata tramite Submodel ma facendo uso di un attributo `outer` in un compartment, con il Professor Tronci si è ritenuto fosse meglio attenersi allo standard W3C.
Port	Una port permette allo sviluppatore di definire come ci si deve interfacciare con un modello, di norma è preferibile seguire le indicazioni dello sviluppatore.
Deletion	Non è detto che i modelli importati abbiano solo ed esclusivamente componenti desiderabili e con una deletion è possibile ignorare ridondanze.
Replacement	Tramite questo costrutto è possibile sostituire una componente con un'altra, ogni riferimento alla prima ora punta alla seconda. A causa di Replacement sono presenti più parser: il primo `parser.py` esplora il file XML come una lista e associare un Replacement ad un componente risulta estremamente macchinoso se non impossibile, in `extended_parser.py` questo problema non si presenta perché il file XML viene esplorato come un albero.

SBaseRef	Port, Deletion, Replacement e Submodel utilizzano dei riferimenti, SBaseRef agisce similmente a SBase e offre degli attributi comuni a tutte queste classi.

Tabella 1: Modellazione SHACL

3 Parsing da SBML a SHACL

In questa fase sono stati scritti due parser, di questi `extended_parser.py` è la versione finale e corretta. In `parser.py` si riescono a tradurre senza problemi SBML 3.2 ma siccome il file XML viene letto come una lista di etichette risulta macchinoso (ammesso e non concesso sia possibile) fare riferimento all'etichetta padre di una sostituzione o una cancellazione. Il problema non si presenta nel secondo parser dato che la struttura XML viene esplorata come un albero, questa seconda versione è inoltre più breve (444 righe anziché 932), più veloce come si può vedere in tabella 3 ed è più facilmente estendibile. Questo parser infatti prende qualsiasi tag e vi associa i rispettivi attributi, che devono essere noti, al proprio valore; ho comunque imposto dei controlli per evitare questo comportamento dato che avrebbe solo rallentato la fase di verifica.

Il calcolo delle performance è basato sull'esecuzione di `test.sh` e si fa uso di file XML: alcuni sono scaricati dal sito Biomodels e supportano solo SBML 3.2 mentre altri (nella sottocartella `custom`) sono file d'esempio forniti direttamente dal W3C, quindi questi ultimi potranno essere usati solo per il file `extended_parser.py`.

File	**System (s)**	**User (s)**	**Total**	**CPU**
parser.py	52.90	2445.38	42:54.54	97 %
extended_parser.py	42.48	833.54	15:02.19	97 %

Tabella 3: Performance

Si assume a priori che i file SBML in input siano corretti, la verifica è eseguibile online. Per il corretto funzionamento del codice è necessario il download del package *rdflib*.

4 Parsing da SHACL a SBML

In questa terza e ultima fase si esegue la controprova della precedente ovvero si traduce il file di istanze SHACL ottenuto in un file SBML e ci si attende che venga modellata la stessa conoscenza. Nella cartella `query` sono presenti tre file:

- `query.py`: Permette di interrogare il modello con query SPARQL.

- `ttl2sbml.py`: Il parser da SHACL a SBML.
- `ttl2xml`: Converte un file di istanze Turtle in XML, una query e un file XML possono esprimere la stessa conoscenza traducendo una tripla `<subject> <predicate> <object>` in etichette `<subject> <predicate> <object/> </predicate> </subject>`.

Nella scrittura del parser è risultato necessario poter riottenere la struttura annidata dell'XML e che si era persa nella scrittura delle istanze SHACL, per farlo ho definito un albero con dei nodi e nella fase di lettura del file si costruiva gradualmente l'albero. Una volta ottenuto per scrivere il testo XML basta esplorare ricorsivamente l'albero ricordandosi di chiudere le etichette aperte una volta terminate le chiamate ai figli.

L'algoritmo genera la stessa conoscenza dei file di partenza, per SBML 3.2 non ci sono problemi invece per i file nella cartella `custom` tutti tranne `MANUAL_1.xml` non superano la verifica SBML perché trattandosi di file d'esempio i riferimenti non sono reali; chiaramente anche l'output di `ttl2sbml.py` dà risultato negativo, per `MANUAL_1.txt` invece si ha lo stesso esito ovvero numerosi warning. Non ho usato esempi presi da Biomodels perché i file trovati con extended SBML implementavano la gerarchia con un attributo `outer` piuttosto che con i `submodel` come da documentazione ufficiale.

5 Commenti e critiche

Durante lo sviluppo del progetto sotto direttive del Professor Enrico Tronci si è tenuto conto dei seguenti commenti e critiche.

Tabella 4: Cronologia

Data	**Commento**
22/07/20	Videochiamata con specifica del problema da parte del Professor Tronci.
27/07/20	Prima stesura di una modellazione dei costrutti in SBML ma avendo io dimenticato di modellare SBML gerarchico mi è stato fatto notare e ho risolto, il materiale consultato è stato reso disponibile dal Professore stesso.
28/07/20	Corretto il punto precedente non avevo implementato Deletions e Replacements, il professore ha inoltre consigliato una strategia di testing.
31/07/20	Dopo aver corretto le mie mancanze e eseguito dei test con risultati positivi è seguita una videochiamata con il Professor Tronci in cui mi è stato indicato di modificare il parser in maniera tale da poter ricevere in input più modelli SBML creando quindi un grafo più complesso, e di capire se il risultato di una query SPARQL rappresenti lo stesso tipo di conoscenza di un file XML/SBML, così da poter sfruttare questa corrispondenza per eseguire dei controtest. Questa corrispondenza esiste ed è usata spesso.
06/08/2020	Videochiamata con il Professor Tronci in cui si è deciso di comprendere se fosse possibile scrivere un parser da SHACL a SBML e, in caso di risposta affermativa, farlo.

6 Fonti

- The systems biology markup language.
- Sparql query language for rdf, 2008.
- Sparql query results xml format (second edition), 2013.
- Rdf 1.1 concepts and abstract syntax, 2014.
- Shapes constraint language (shacl), 2017.
- Michael Hucka, Frank T. Bergmann, Claudine Chaouiya, Andreas Dräger, Stefan Hoops, Sarah M. Keating, Matthias König, Nicolas Le Novère,Chris J. Myers, Brett G. Olivier, Sven Sahle, James C. Schaff, RahumanSheriff, Lucian P. Smith, Dagmar Waltemath, Darren J. Wilkinson, andFengkai Zhang. The systems biology markup language (sbml): Languagespecification for level 3 version 2 core, 2019.
- Lucian P. Smith, Stefan Hoops, Martin Ginkel, Ion Moraru, Michael Hucka,Andrew Finney, Chris J. Myers, and Wolfram Liebermeister. Hierarchicalmodel composition, 2013.