5.1.2 Clark & Parsia Stardog

Im laufe der Arbeit wurde festgestellt, dass das Reasoning unter Protégé nicht einwandfrei funktioniert, also sparql Anfragen nicht erwartungsgemäss beantwortet. Ausserdem exisitert in Protégé keine, mit einem sinnvollen Zeit und Arbeitsaufwand zu verwendende, http Schnittstelle. Aus diesen Gründen wird zur Weiterverarbeitung, der in Protégé erzeugten Ontologie Stardog verwendet.
Konkret wird unter Stardog eine Datenbank erstellt, in welche die vorbereitete Ontologie im RDF/XML Format eingespielt wird. Stardog bietet dann die Möglichkeit per http Anfragen über REST Anfragen zu stellen. Dabei ist ein komplettes und zuverlässiges Reasoning gewährleistet.

Bei Stardog handelt es sich um ein führendes Entwicklung und Anwendungssystem von Wissenmodellierung mit Suchen, Anfragen und Reasonings für Java Systeme. Stardog bietet unterschiedliche Formate an. Neben der kostenpflichtigen Enterprise Version, gibt es glücklicherweise eine Community Version. Diese bietet eine eingeschränkte Vielfallt von Datenbanken, Benutzern und Verbindungen, ist aber frei verfügbar.[13]

Stardog kann auch als Graphische Datenbank bezeichnet werden. Dabei unterstützt es viele nützliche Verwendungsformen:

- RDF Daten Modelle
- SPARQL[1] als Abfragesprache
- HTTP und SNARL Protokolle als remote Zugriff und Kontrolle
- OWL 2 zur Modellierung der Ontologien
- Regeln für Inferenz und Datenanalyse
- Java, JavaScript, Ruby, ṄET und weitere Programmiersprachen

[14]

Features

Der folgende Abschnitt basiert auf der offiziellen Stardog Dokumentation.[15]

Querying:
Anfragen können auf einer RDF Datenbank mittel der SPARQL Abfragesprache gestellt werden. Mit dem folgenden Befehlt kann eine Afrage abgesetzt werden.

```
\$ stardog query <Datenbankname> ''<SPARQLAnfrage>''
```

Neben sämtlichen SPARQL Funktionen Unterstützt Stardog weitere von XPath und SQRL. Einige dieser Funktionen können für Anfragen oder Regeln verwendet werden.

Updating:
Es gibt verschiedene Arten um in Stardog die Datenbank zu aktualisieren. Die üblichste Art ist das Updaten mittels CLI und SPARQL Queries. Diese Funktion wird in diesem Projekt nicht genutzt, da die Daten in Protégé aufbereitet werden.

Versioning:
Stardog unterstützt Versionierung mittels "graph change management capability", welche es dem Benutzer erlaubt Änderungen zu verfolgen.
Die Versionierung für eine Datenbank standarmässig deaktiviert und muss bei bedarf aktiviert werden.

Exportieren:
Es ist möglich die in der Stardog Datenbank enthaltenen Daten nach RDF zu exportieren. Insbesondere können auch nur einzelne "named graph" exportiert werden.

[1] http://www.w3.org/TR/sparql11-query/

Searching:
Stardog includes an RDF-aware semantic search capability: it will index RDF literals and supports information retrieval-style queries over indexed data

Obfuscating:
When sharing sensitive RDF data with others, you might want to (selectively) obfuscate it so that sensitive bits are not present, but non-sensitive bits remain. For example, this feature can be used to submit Stardog bug reports using sensitive data.

und was ist mit der ganzen Technik hinter Stardog?

Reasoner

Reasoner sind Komponenten, welche eine Folgerung von implizitem Wissen zulassen bzw. bieten. Es handelt es sich um eine Art "Verstehen" durch Maschinen. Man möchte also implizite Fakten finden, welche durch explizite Fakten in einer Ontologie definiert sind.

Als Reasoner kommt Pellet zum Einsatz. Dabei handelt es sich um einen Reasoner, für die Sprache OWL-DL [2], welche wiederum eine Teilsprache von OWL darstellt. Eine komplette Unterstützung des OWL full Profils ist so nicht möglich, da dieses nicht entscheidbar ist, daher beschränkt sich die Umsetzung auf OWL DL. Pellet setzt dabei komplett die Spezifikation der WebOnt Arbeitsgruppe um. OWL-DL ist eine syntaktische Variante der Beschreibungslogik SHOIN (D).

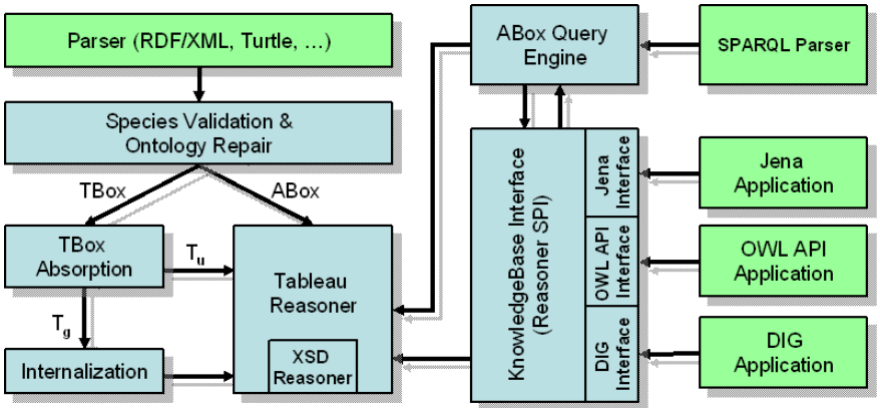

Abbildung 5.1: Hauptkomponenten des Pellet-Reasoners.[3]

Beschreibungslogik

Beschreibungslogiken sind Formalismen um Wissen darzustellen, dabei sind sie eine Teilmenge der Prädikatenlogik. Sie stellen den Kern von Wissensrepräsentationssystemen dar, in dem sie eine Struktur für eine Wissensbasis und den damit verbundenen Methoden zur Folgerung bieten (vgl. [17]).

Die Struktur, welche Beschreibungslogiken als Wissensbasis bereitstellen, besteht aus einem Schema (Tbox, Regeln), sowie aus den Daten (Abox, Fakten).

Die Semantik von Beschreibungslogiken wird durch Interpretationen definiert:

$$I = (\Delta^I, \cdot^I)$$

Dabei ist:

- Δ^I

  Die (Wissens-) Domäne (eine nicht leere Menge).

[2] `http://www.w3.org/TR/owl-ref/#OWLDL`
[3] [16, S. 6]

- $\cdot^I$

  Eine Funktion zur Interpretation von:

  - Konzepten (Klassen, A)

    Wobei A^I eine Teilmenge von Δ^I ist.

  - Rollen (Eigenschaften, R)

    Wobei R^I eine binäre Relation auf Δ^I ist.

  - Individuen (i)

    Wobei i^I Element von Δ^I ist.

Die Funktion zur (Wissens-) Interpretation $\cdot^I$ definiert, wie atomare Konzpte, Eigenschaften und Individuen zu interpretieren sind. Eine Interpretation, welche allen Axiomen einer Ontologie in Beschreibungslogik genügt ist ein Modell der Ontologie.

Eine Ontologie in Beschreibungslogik ist eine Menge von Termen und deren Relationen. Die Interpretation dieser Ontologie stellt als ein Modell dar, welches die Ontologie abbildet.

Die Pellet zu Grunde liegende Beschreibungslogik SHOIN (D) ist ein Kürzel und steht für:

- *S*

  ALC (Attributive Concept Language with Complements) mit einer transitiven Rolle. Bei ALC handelt es sich um die kleinste Beschreibungslogik, welche aussagenlogisch geschlossen ist (d.h. sie bietet, entweder implizit oder explizit, Konjunktion, Union und Negation von Klassen (-beschreibungen)). Eine Rolle entspricht einer Eigenschaft bzw. einem Prädikat in der Prädikatenlogik erster Stufe.

- *H*

  Rollenhierarchie (Sub-Eigenschaften), z.B. rdfs:subPropertyOf.

- *O*

  Nominal, z.B. Wochenden = Samstag, Sonntag.

- *I*

  Inverse Rolle (inverse Eigenschaft bzw. Prädikt).

- *N*

  Nummerische Einschränkung, z.B. >= hatKind 1.

- *(D)*

  Nutzung von Wertebereichen, Werten oder Datentypen.

(vgl. [17])

Folgerung

Die W3C Spezifikation betreffend der Umsetzung eines Reasoners definiert zwei Arten von OWL Dokumentprüfungen: Prüfung auf korrekte Syntax sowie Prüfung der Konsistenz. Bei der Konsistenzprüfung geht es darum festzustellen, ob eine gegebene Eingabe anhand einer definierten Spezifikation syntaktisch korrekt ist. Jedoch macht eine reine Umsetzung der genannten Prüfungen wenig Sinn. Man möchte ja eine Ontologie möglichst effektiv nutzen können, zum Beispiel um indirektes Wissen abzuleiten. Da OWL DL eine syntaktische Variante der Beschreibungslogik SHOIN (D) darstellt, liegt es nahe, mittels einem Reasoner folgende Methoden zur Inferenz zur Verfügung zu stellen:

- *Konsistenzprüfung*

  Stellt sicher, dass sich in der Ontologie keine gegenstätzlichen Fakten befinden. Konkret bedeutet dies anhand der OWL Semantik [4], dass die Konsistenz einer ABox in Bezug auf eine TBox geprüft wird.

- *Konzept-Erfüllbarkeit*

  Stellt sicher, dass Objekte einer Klasse instanziert werden können. Ist dies nicht der Fall, so ist die Ontologie inkonsistent.

- *Klassifikation*

  Berechnet die Relationen zwischen Subklassen, stellt also die Klassenhierarchie auf. Dies ermöglicht das Abfragen der Ontologie.

- *Gewinnung von Erkenntnissen*

  Findet die spezifischste Klasse eines Objektes, leitet also den direkten Typ jedes Objektes ab.

(vgl. Sirin et al. [16, S. 1 und 2])

Bei der eigentliche Komponente, welche in Pellet zur Folgerung eingesetzt wird, handelt es sich um den Tableau Reasoner, welcher den gleichnamigen Algorithmus verwendet. Dieser reduziert ein Problem der Folgerung auf ein Problem der Konzept-Erfüllbarkeit, wobei er eine Interpretation sucht, welche die gefragten Konzepte erfüllt. Solch eine Interpretation wird inkrementell als eine Art Tafel (eben, Tableau) aufgebaut.

Nachfolgend ein Beispiel, gegeben sei: $Prolog \subseteq LogischeProgrammiersprache, LogischeProgrammiersprache \subseteq Programmiersprache$ Man stellt nun folgende Anfrage: $if Prolog \subseteq Programmiersprache$

Der Prozess der Folgerung findet nun wie folgt statt:

- Testen, ob ein Individuum existiert, welches eine logische Programmiersprache, aber keine Programmiersprache ist. Man möchte also die Erfüllbarkeit des Konzeptes $C_0 = (Prolog \sqcap \neg Programmiersprache)$ testen.
- $C_0(x) \Rightarrow Prolog(x), (\neg Programmiersprache)(x)$
- $Prolog(x) \Rightarrow LogischeProgrammiersprache(x)$
- $LogischeProgrammiersprache(x) \Rightarrow Programmiersprache(x) \Rightarrow$ Konflikt!

Wie ersichtlich ist, ist das Konzept C_0 nicht erfüllbar, die Anfrage $Prolog \subseteq Programmiersprache$ trifft also innerhalb der gegebenen Ontologie zu. Der Beweis wird mittels Kontradiktion erbracht.

Der generelle Prozess der Folgerung läuft wie folgt ab:

- Umwandlung eines Konzeptes bzw. der Konzepte in die negierte Normalform (NNF), die Negation $\neg$ befindet sich vor Konzeptnamen.
- Ablegen des umgewandelten Konzeptes als C_0.

  Der Algorithmus hat also die Datenbasis (Abox) $A_0 = C_0(x_0)$
- Regeln zur Transformation auf die Datenbasis (Abox) so weit als möglich anwenden.
- Wurde eine gültige Datenbasis (Abox) gefunden, so ist C_0 erfüllbar.
- Wurde keine gültige Datenbasis unter Berücksichtigung aller (Such-) Pfade gefunden, so ist C_0 nicht erfüllbar.

(vgl. [18] und [19])

[4] http://www.w3.org/TR/owl-semantics/

Literaturverzeichnis

[1] Dr. Jürgen Eckerle. Aufgabenstellung bachelorthesis, 2014.

[2] Informatik Spektrum. Was bedeutet eigentlich ontologie?, September 2014.

[3] Informatik Spektrum. Eine kurze geschiche der ontologie, September 2014.

[4] Seaborne Harris. Sparql 1.1 query language, October 2014. URL http://www.w3.org/TR/sparql11-query/.

[5] W3C SPARQL Working group. Sparql 1.1 overview, October 2014. URL http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

[6] Beckett. Rdf/xml syntax specification (revised), October 2014. URL http://www.w3.org/TR/REC-rdf-syntax/.

[7] Raimond Schreiber. Rdf 1.1 primer, October 2014. URL http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/.

[8] Hitzler, Krötzsch, Parsia, Patel-Schneider, and Rudolph. Owl 2 web ontology language, October 2014. URL www.w3.org/TR/2012/REC-owl2-primer-20121211/.

[9] I. Horrocks, Peter F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. Swrl: A semantic web rule language combining owl and ruleml, November 2014. URL http://www.w3.org/Submission/SWRL/.

[10] J. Cleve U. Lämmel. *Künstliche Intelligenz*. Carl Hanser Verlag München, 4rd edition, 2012.

[11] Stanford University. Protege desktop features, November 2014. URL http://protegewiki.stanford.edu/wiki/Protege4Features.

[12] Stanford University. Protege 4.x views, November 2014. URL http://protegewiki.stanford.edu/wiki/Protege4Views#Individual_views.

[13] Clark & Parsia. stardog, November 2014. URL http://www.stardog.com/.

[14] Clark & Parsia. Stardog docs, November 2014. URL http://docs.stardog.com.

[15] Clark & Parsia. Using stardog, November 2014. URL http://docs.stardog.com/using/.

[16] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner, 2005.

[17] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003. ISBN 0-521-78176-0.

[18] Ian Horrocks and Ulrike Sattler. Reasoning with expressive description logics: Logical foundations for the semantic web, 2002.

[19] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure of shoiq, 2005.