Linked Data Reasoning

Jens Lehmann and Lorenz Bühmann

Universität Leipzig

Institut für Informatik

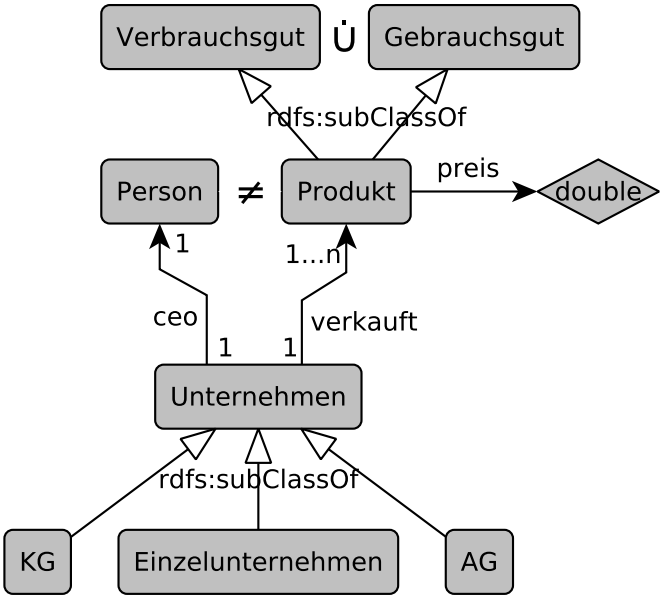
Zusammenfassung. In diesem Kapitel beschreiben wir die Grundlagen des Reasonings in RDF/OWL-Wissensbasen. Wir gehen darauf ein welche unterschiedlichen Arten des Reasonings es gibt, geben einen Überblick über verwendete Verfahren und beschreiben deren Einsatz im Linked Data Web.

1 Einleitung

Die Publikation von Informationen als Linked Data bietet Möglichkeiten maschinenlesbare Daten auszutauschen und miteinander zu integrieren. Ein Vorteil des Einsatzes von semantischen Technologien, insbesondere den W3C-Standards RDF, RDFS und OWL, ist in diesem Fall die klar definierte *Semantik*. Diese erlaubt es Anwendungen die Daten nicht nur einzulesen, sondern auch zu verstehen.

In diesem Kapitel möchten wie uns eben mit diesem Verstehen der Daten und dem Ziehen von Schlussfolgerungen darauf befassen. Wir verwenden dafür den Begriff *Reasoning*, der weitaus geläufiger als eingedeutschte Formen ist. Insbesondere möchten wie dem Leser hier zuerst einen breiteren Überblick über die verschiedenen Arten von Reasoning geben um ein Grundverständnis aufzubauen. Darauf aufbauend möchten wir konkrete Reasoning-Verfahren skizzieren und anschließend Anwendungsszenarien mit einem Fokus auf den Linked Enterprise Data Kontext beschreiben.

Als laufendes Beispiel werden wir uns soweit möglich auf folgendes Szenario beziehen:



Beispiel 1 (Laufendes Beispiel). Hierbei beschreiben wir einen kleinen Ausschnitt aus einer Unternehmensontologie. In dieser gibt es Unternehmen, die sich in Einzelunternehmen, KG und AG unterteilen. Diese Unternehmen verkaufen Produkte, welche Gebrauchs- oder Verbrauchsgüter sind. Zudem besitzt jedes Produkt einen Preis. Außerdem haben die Unternehmen einen CEO, bei denen es sich um Personen handelt. Zusätzlich ist bekannt, dass Personen und Produkte nicht dasselbe sein können.

Eine Darstellung des Schemas in Form von Tripeln könnte dabei folgendermaßen aussehen:

```
:ceo          rdfs:domain :Unternehmen; rdfs:range :Person.
:verkauft     rdfs:domain :Unternehmen; rdfs:range :Produkt.
:preis        rdfs:domain :Produkt; rdfs:range  xsd:double.
:Produkt      rdfs:subClassOf
          [owl:unionOf(:Gebrauchsgut, :Verbrauchsgut)].
:Person       rdfs:subClassOf
                    [owl:intersectionOf(
                          [owl:complementOf :Gebrauchsgut],
                          [owl:complementOf :Verbrauchsgut])].
:Unternehmen rdfs:subClassOf
          [rdf:type owl:Restriction; owl:onProperty :ceo;
           owl:someValuesFrom :Person],
          [rdf:type owl:Restriction; owl:onProperty :verkauft;
           owl:someValuesFrom :Produkt].
```

```
:Gebrauchsgut rdfs:subClassOf
        [owl:complementOf :Verbrauchsgut].
```

Anhand dieses Beispiels werden wir zuerst in Abschnitt 2 die verschiedenen Arten des Reasoning erläutern. In den Abschnitten 3 und 4 erläutern wir sogenannte regelbasierte und tableaubasierte Verfahren. Danach wird in 5 der Einsatz solcher Verfahren in Unternehmen erläutert. In Abschnitt 6 schließen wir das Kapitel mit einem Ausblick auf zukünftige Perspektiven ab.

2 Arten des Reasoning

Reasoning selbst ist ein relativ allgemeiner Prozess, welcher basierend auf bereits vorhandenen Wissen dazu genutzt werden kann um z.B. neues Wissen abzuleiten, Erklärungen für Fakten zu finden, oder auch Vorhersagen zu treffen. Im Rahmen dieses Kapitels werden wir grundsätzlich zwischen 3 der gängigsten Arten von Reasoning unterscheiden und diese genauer erläutern: Die erste Methode ist das sogenannte *Induktive Reasoning*, bei der auf Basis von einer Menge von Beobachtungen versucht wird, etwas allgemeingültiges herzuleiten, d.h. der Weg führt hier „vom Speziellen auf etwas Allgemeines“. Bei der zweiten Form, dem *Deduktiven Reasoning*, wird dagegen von einer Menge von allgemein geltenden Regeln ausgegangen, und unter Anwendung dieser versucht etwas konkretes, möglicherweise Neues herzuleiten. Man kann sich diese Vorgehensweise also prinzipiell als den Prozess „vom Allgemeinen auf etwas Spezielles“ vorstellen. Neben diesen beiden Verfahren gibt es noch eine dritte Form von Reasoning, das *Abduktive Reasoning*. Hierbei wird wie schon beim Induktiven Reasoning üblicherweise von einer Menge von Beobachtungen ausgegangen, welche aber zusätzlich auch unvollständig sein kann. Für diese Menge wird dann versucht eine ”passende”mögliche Erklärung zu finden. Im Folgenden zeigen wir Reasoning anhand eines einfachen Beispiels.

2.1 Deduktives Reasoning

Das deduktive Reasoning ist die geläufigste der 3 Arten von Reasoning. In vielen Fällen wird Reasoning mit deduktivem Reasoning gleichgesetzt aufgrund der erheblichen Unterschiede zu den beiden anderen Arten. Innerhalb dieses Buchkapitels wollen wir jedoch eine breitere Perspektive bieten. Um die Begriffe näher zu bringen, verwenden wir hier sehr einfache Beispiele und verweisen auf Kapitel 5 für reale Anwendungsmöglichkeiten.

Beispiel 2 (Deduktives Reasoning).
Gegeben: Alles was einen CEO hat, ist ein Unternehmen. `ReasoningAG` hat einen CEO.
Deduktives Reasoning: `ReasoningAG` ist ein Unternehmen.

In diesem Beispiel sind zwei Informationen gegeben aus denen eine dritte Information geschlussfolgert werden kann. Das deduktive Reasoning zeichnet sich dadurch aus, dass die Schlussfolgerung zwangsläufig aus den gegebenen Informationen folgt: Da `ReasoningAG` einen CEO hat, muss es ein Unternehmen sein. Wie wir in den Abschnitten 3 und 4 zeigen, können diese Schlussfolgerungen mit Reasoning-Verfahren berechnet werden. Innerhalb einer Sprache wie OWL DL kommt jedes solche Verfahren, welches berechnet ob eine Aussage aus anderen Aussagen folgt, immer zur gleichen Lösung.[1]

2.2 Induktives Reasoning

Obiges Beispiel kann etwas abgewandelt werden um induktives Reasoning zu erläutern:

Beispiel 3 (Induktives Reasoning).
Gegeben: `ReasoningAG` hat einen CEO. `ReasoningAG` ist ein Unternehmen.
Induktives Reasoning: Alle Unternehmen haben einen CEO.

In diesem Fall sind zwei Fakten gegeben. Aus diesen Fakten kann deduktiv nicht geschlussfolgert werden, dass alle Unternehmen einen CEO haben. Wir wissen lediglich, dass dies für ein konkretes Unternehmen zutrifft: `ReasoningAG`. Beim induktiven Reasoning geht es also darum konkrete Fakten zu verallgemeinern und in einer allgemeineren schematischen Aussage zusammenzufassen. Inwiefern diese Verallgemeinerung tatsächlich der Wahrheit entspricht, kann entweder durch die Prüfung durch einen Menschen (wie in diesem Fall möglich) oder z.B. ein Kreuzvalidierungsverfahren herausgefunden werden. Je mehr konkrete Fakten die Schlussfolgerung unterstützen, desto besser. Wenn also in diesem Fall neben `ReasoningAG` noch zahlreiche andere Unternehmen einen CEO haben, dann kann ein induktiver Reasoner mit höherer Sicherheit lernen, dass alle Unternehmen einen CEO haben. Wichtig ist beim induktiven

[1] Wir setzen hier voraus, dass die Verfahren korrekt und vollständig sind, d.h. dass die Verfahren fehlerfrei jede Schlussfolgerung berechnen können. In der Praxis wird teilweise eine dieser Eigenschaften „geopfert“ um bessere Performance zu erzielen.

Reasoning auch anzumerken, dass es nicht nur dazu dienen kann, absolute Wahrheiten herauszufinden, sondern auch schwächere Zusammenhänge. Zum Beispiel könnte ein induktiver Reasoner im E-Commerce Bereich lernen, dass Personen, die sich für den ersten Teil eine Filmtrilogie interessieren, sich auch für den zweiten und dritten Teil interessieren. In diesem Fall ist es offensichtlich, dass es sich nicht um eine absolute Wahrheit handelt, da es zweifelsohne Personen gibt, die einer Filmtrilogie nach dem ersten Teil den Rücken kehren. Dennoch kann der obige Zusammenhang für den Betreiber eines Online-Shops sehr sinnvoll und wichtig sein um Kunden gezielt Angebote zu unterbreiten.

2.3 Abduktives Reasoning

Unser Beispiel kann wiederum umgestellt werden um abduktives Reasoning zu erklären:

Beispiel 4 (Abduktives Reasoning).
Gegeben: Alle Unternehmen haben einen CEO. `ReasoningAG` hat einen CEO.
Abduktives Reasoning: `ReasoningAG` ist ein Unternehmen.

In diesem Fall ist eine schematische Aussage und ein konkreter Fakt gegeben. Analog zum induktiven Reasoning kann auch hier die präsentierte Aussage, dass `ReasoningAG` ein Unternehmen ist, nicht logisch geschlussfolgert werden. Statt einem Unternehmen könnte `ReasoningAG` beispielsweise ein Produkt sein. Beim abduktiven Reasoning geht es also darum konkrete Fakten zu finden, die als Erklärung für unser bekanntes Wissen verwendet werden können. Diese Fakten sollten möglichst plausibel sein, aber wie bereits beim induktiven Reasoning, wird hier eine separate Prüfung z.B. durch einen Menschen benötigt. Abduktion kann zum Beispiel zum vervollständigen von Wissen eingesetzt werden. In einigen Fällen wurden abduktive Reasoner sogar eingesetzt, damit Roboter komplett selbständig wissenschaftliche Vermutungen aufstellen und diese dann anschließend prüfen können [19] und so letztendlich Erkenntnisse gewinnen.

3 Leichtgewichtiges Regelbasiertes Reasoning

Die Semantik von RDF und RDFS [12], sowie das Reasoning in OWL RL [26] kann durch sogenannte Ableitungsregeln repräsentiert werden. Intuitiv handelt es sich bei einer Ableitungsregel um eine “Wenn - Dann”

Regel, die das Wissen festlegt (Regelkopf, Konklusion) welches aus dem vorhandenen Wissen (Regelkörper, Prämisse) folgt. Der Regelkörper besteht dabei für gewöhnlich aus RDF Tripeln, in denen Variablen an jeder Position (Subjekt, Prädikat, Objekt) vorkommen können. Der Regelkopf umfasst die Konsequenzen, wobei jede davon ebenfalls ein RDF Tripel repräsentiert, mit der Einschränkung dass keine Variablen auftreten dürfen, die nicht auch im Regelkörper verwendet werden. Die Liste von RDF/RDFS Regeln ist in [12] definiert, für OWL RL findet man die Regeln unter [26].

Beispiel 5 (Ableitungsregel).
Als ein Beispiel sei hier die RDFS Ableitungsregel *rdfs9* für die Subklassenrelation (rdfs:subClassOf) aufgeführt:

```
WENN T(?x, rdf:type, ?c), T(?c, rdfs:subClassOf, ?d)
DANN T(?x, rdf:type, ?d)
```

Die Regel kann dabei folgendermaßen verstanden werden: Wenn es eine Instanz `x` der Klasse `c` gibt, und `c` als Subklasse der Klasse `d` definiert ist, dann ist `x` auch Instanz von `d`.

Üblicherweise unterscheidet man zwischen zwei verschiedenen Vorgehensweisen bei der Anwendung von Regeln um zu einer Konklusion zu gelangen:

1. Man beginnt mit der gegebenen Menge an Fakten und wendet rekursiv die Regeln "vorwärts" an, um neue Fakten zu erzeugen. Dieses Vorgehensweise wird auch als *Forward-Chaining* bezeichnet.
2. Man beginnt mit einer Hypothese und versucht durch rekursive Anwendung der Regeln Fakten zu finden, die diese Hypothese belegen. Die Regeln werden dabei "rückwärts" angewendet, weshalb man dieses Prinzip auch als *Backward-Chaining* bezeichnet.

Ein einfaches Beispiel für Reasoning in OWL RL könnte dabei folgendermaßen aussehen:

Beispiel 6 (Forward Chaining in OWL RL).
Nehmen wir einmal folgendes Wissen basierend auf unserem eingangs definierten Schema an: Es gibt ein Unternehmen `ReasoningAG`, das `ReasonerXYZ` verkauft. Anhand der Daten ist allerdings nicht explizit bekannt um was es sich bei `ReasonerXYZ` handelt. Aus dem Schema wissen wir bereits, dass der Range der Relation `verkauft` ein Produkt ist. Mit Hilfe der OWL RL Regel *prp-rng*

```
WENN T(?p, rdfs:range, ?c), T(?x, ?p, ?y)
DANN T(?y, rdf:type, ?c)
```

können wir dann ableiten, dass `ReasonerXYZ` ein Produkt ist.

Erweitern wir nun unsere Wissensbasis um den Fakt, dass `ReasonerXYZ` der CEO von `ReasoningAG` ist. Aus unserer Ontologie wissen wir, dass jeder CEO eine Person ist, d.h. nach obiger Regel ist auch `ReasonerXYZ` eine Person. Zusätzlich ist aus dem Schema bekannt, dass ein Produkt keine Person ist. Wenden wir jetzt OWL RL Regel *cax-dw* an, welche definiert ist als

```
WENN T(?c1, owl:disjointWith, ?c2), T(?x, rdf:type, ?c1),
     T(?x, rdf:type, ?c2)
DANN Widerspruch
```

so erhalten wir offensichtlich einen Widerspruch, denn `ReasonerXYZ` ist sowohl ein Produkt als auch eine Person, jedoch ist dies explizit in unserem Schema untersagt.

4 Tableau-Basiertes Reasoning

Während für RDF/RDFS und einige OWL 2 Profile meist regelbasierte Methoden für deduktives Reasoning verwendet werden können (siehe Abschnitt 3), ist dies für OWL 2 nicht möglich, da man bedingt durch die höhere Ausdrucksmächtigkeit nicht in der Lage ist, eine vollständige Menge von Regeln zu definieren. OWL basiert auf sogenannten Beschreibungslogiken [3], so dass es sich anbietet auf deren Beweisverfahren zurückzugreifen. Die in der Praxis mit am häufigsten verwendete Methode ist das Tableauverfahren. Wir verzichten hier bewusst auf theoretische Grundlagen und werden im folgenden nur eine sehr grobe Übersicht darüber geben. Für eine ausführliche Beschreibung verweisen wir daher auf [3].

Die Grundidee beim Tableauverfahren ist, dass man eine Aussage versucht zu beweisen, indem man annimmt, dass die gegenteilige Aussage gilt und die gegebene Wissensbasis dann auf Erfüllbarkeit testet. Man zählt das Tableauverfahren deshalb auch zu den sogenannten Widerlegungskalkülen. Die Erfüllbarkeit einer Aussage wird dabei getestet indem versucht wird ein minimales Modell zu konstruieren. Dazu wird schrittweise unter Anwendung von Tableau-Erweiterungsregeln (ein Auszug siehe Tabelle 1) ein Tableau aufgebaut. Ein Tableau ist ein Baum welcher solch ein Modell repräsentiert. Die Knoten in dem Baum stehen dabei für Individuen und die Kanten stellen Beziehungen zwischen den Individuen dar. Jeder Knoten ist mit den Konzepten markiert zu denen das Individuum gehört und jede Kante ist mit der Relation ausgezeichnet, welche die Beziehung beschreibt. Je nach anzuwendender Regel wird u.a. das Tableau

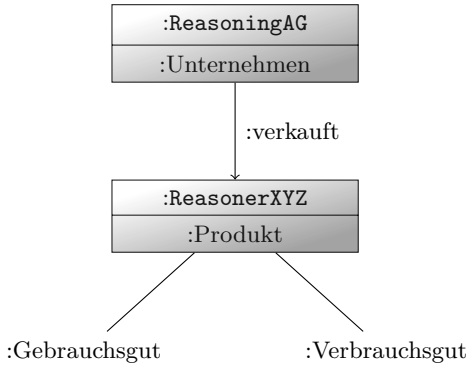
Name	Auswahl	Aktion
$C_{\mathcal{A}}$	`:x rdf:type :C`	Füge neuen Knoten `:x` mit dem Label `:C` hinzu.
$R_{\mathcal{A}}$	`:a :p :b`	Füge Kante mit Label `p` von Knoten `a` zu Knoten `b` hinzu.
$\sqcap$	`:x rdf:type [owl:intersectionOf (:A, :B)]`	Füge `:A` und `:B` zu Knoten `:x` hinzu.
$\sqcup$	`:x rdf:type [owl:unionOf (:A, :B)]`	Dupliziere den Zweig. Füge zum einen Zweig $C(a)$ und zum anderen Zweig $D(a)$ hinzu.
$\exists$	`:x rdf:type [rdf:type owl:Restriction; owl:onProperty :p; owl:someValuesFrom :A]`	Füge $R(a, b)$ und $C(b)$ für neues Individuum b hinzu.
$\forall$	`:x rdf:type [rdf:type owl:Restriction; owl:onProperty :p; owl:allValuesFrom :A]`	Falls es eine Kante `:p` von `:x` zu einem Knoten `:y` gibt, so füge `:A` zu `:y` hinzu.

Tabelle 1. Auszug aus Tableau-Erweiterungsregeln basierend auf OWL2 DL.

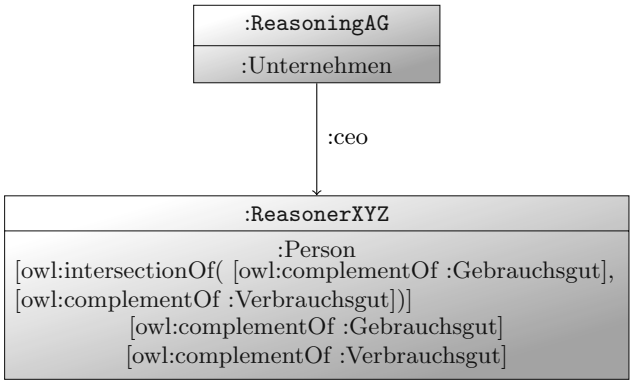
um neue Knoten erweitert, Markierungen bestehender Knoten werden mit weiteren Konzepten versehen, oder es entstehen Verzweigungen im Baum. Ein Pfad vom Wurzelknoten zu einem Blattknoten wird nun als abgeschlossen bezeichnet, wenn entlang des Pfades Knoten auftreten, die widersprüchliche Aussagen repräsentieren, d.h. Knoten deren Markierung sowohl ein Konzept als auch dessen Negation enthalten. Ein Tableau ist abgeschlossen, wenn alle Pfade abgeschlossen sind. In diesem Fall wurde die Unerfüllbarkeit der Wissensbasis gezeigt, da kein widerspruchsfreies Modell konstruiert werden konnte.

Betrachten wir erneut die erweiterte Wissensbasis aus Beispiel 6 und testen der Erfüllbarkeit, d.h. wir suchen nach einem Modell oder finden einen Widerspruch. Zunächst einmal haben wir nach Regel $C_{\mathcal{A}}$ zwei Knoten im Tableau: Knoten :`ReasoningAG` welcher als Label :`Unternehmen` enthält sowie einen Knoten :`ReasonerXYZ`, vorerst ohne Label. Analysieren wir zuerst die :`verkauft` Relation aus unserer Ontologie. Wir wissen dass `ReasoningAG ReasonerXYZ` verkauft, das bedeutet nach Regel $R_{\mathcal{A}}$ fügen wir eine Kante von :`ReasoningAG` nach :`ReasonerXYZ` hinzu, welche als Bezeichnung :`verkauft` erhält. Als nächstes können wir nun Regel $\forall$ anwenden, wodurch Knoten :`ReasonerXYZ` das Label :`Produkt` erhält. Aus unserer Ontologie wissen wir, dass ein Produkt entweder Gebrauchsgut oder Verbrauchsgut ist, so dass wir hier Regel $\sqcup$ nutzen können. Dadurch

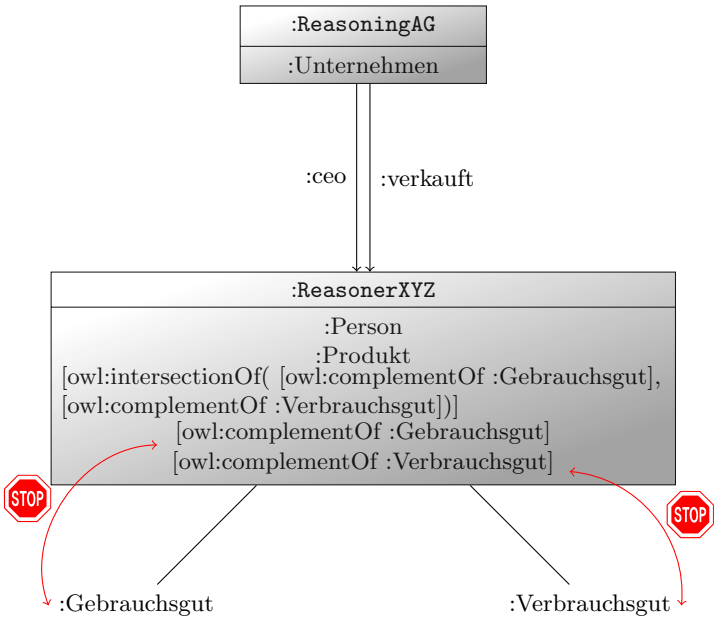
erhalten wir eine Verzweigung im Tableau, denn es besteht die Möglichkeit dass :`ReasonerXYZ` ein Gebrauchsgut oder aber ein Verbrauchsgut ist. Bis hierhin haben wir soweit alles hinsichtlich Relation :`verkauft` getestet und das im folgenden einmal graphisch veranschaulichte Tableau wäre nicht abgeschlossen, d.h. die Wissensbasis wäre erfüllbar.



Allerdings müssen wir für eine vollständige Ausführung des Tableauverfahrens auch noch die Relation :`ceo` untersuchen. Der Übersichtlichkeit halber betrachten wir dies zunächst separat. Wir haben nach Regel $C_{\mathcal{A}}$ wieder die beiden Knoten :`ReasoningAG` mit dem Label :`Unternehmen`, sowie :`ReasonerXYZ` ohne Label. In unserer Wissensbasis ist :`ReasonerXYZ` als CEO von :`ReasoningAG` angegeben, so dass aus Regel $R_{\mathcal{A}}$ eine Kante bezeichnet mit :`ceo` folgt. Außerdem wissen wir, dass eine Person weder Gebrauchs- noch Verbrauchsgut ist, so dass wir nach Regel $\sqcap$ die jeweilige Negation (`owl:complementOf`) als Label zum Knoten :`ReasonerXYZ` hinzufügen. Das resultierende Tableau, im folgenden wieder als Graph dargestellt, wäre ebenso nicht abgeschlossen, wie das für die :`verkauft` Relation, das bedeutet auch in diesem Fall wäre unsere Wissensbasis erfüllbar.



Wir haben beide relevanten Relationen bisher getrennt betrachtet, aber selbstverständlich ist dies im Tableauverfahren nicht der Fall. Wenn wir also beide Relationen gemeinsam analysieren und das entsprechende Tableau aufbauen, so entsteht folgender Graph:



Wir sehen also, dass wir zwei Kanten vom Knoten `:ReasoningAG` zu `:ReasonerXYZ` haben, und wir haben eine Menge von Labels für Knoten `:ReasonerXYZ` sowie eine Verzweigung. Viel entscheidender ist aber, dass wir jetzt einen Widerspruch in allen Pfaden haben - hervorgehoben durch die rote Kanten - denn es kann nicht sein dass ein Individuum (hier `:ReasonerXYZ`) sowohl zu einer Klasse als auch zu deren Negation gehört. Das bedeutet alle Pfade im Tableau sind abgeschlossen, und es ist somit nicht möglich ein Modell für unsere Wissensbasis zu erzeugen. Insbesondere bedeutet dies dass die Wissensbasis unerfüllbar ist, es gibt also Aussagen, die widersprüchlich sind.

5 Reasoning im Linked Data Web

5.1 Besondere Herausforderungen des Einsatz von Reasoning im Linked Data Web

Der Einsatz von den oben eingeführten Reasoning-Technologien ist keinesfalls auf das Linked Data Web beschränkt. Es muss nicht zwangsläufig

im Webkontext eingesetzt werden und kann neben den Semantic Web Standards für andere logische Sprachen eingesetzt werden. Gegenüber der eher traditionellen lokalen Nutzung ergeben sich im Linked Data Web jedoch spezielle Herausforderungen, die wir hier erwähnen möchten.

Der Mehrwert einer guten maschinenlesbaren Syntax zur Wissensrepräsentation wird oft unterschätzt, obwohl er eine fundamentale Basis für den Austausch von Wissen darstellt. Das World Wide Web Consortium (W3C) hat eine Reihe von Empfehlungen verabschiedet, zum Beispiel RDF [21], OWL [14], und RIF [4], aber auch spezialisierte Vokabulare wie SKOS Simple Knowledge Organization System [25], SSN Semantic Sensor Networks [8] und Provenance [11]. Obwohl es allgemein üblich ist die Skalierbarkeit von solchen Sprachstandards zu untersuchen, wird dies durch den Einsatz im Web nochmal auf ein anderes Level gehoben. Paralleles und verteiltes Reasoning spielt hierbei eine große Rolle [18,27].

Außerdem ist es derzeit häufig der Fall, dass Reasoning-Anwendungen auf saubere, anwendungsspezifische, manuell kuratierte Wissensbasen zugreifen. Im Linked Data Web ist solch ein Szenario oft unrealistisch. Obwohl es über Crowdsourcing-Ansätze teilweise möglich ist die Datenqualität zu erhöhen [1], müssen Reasoning-Verfahren im Linked Data Web in der Lage sein massive, verteilte Datenvolumina zu verarbeiten, die von mehreren Autoren für verschiedene Anwendungszwecke erschaffen wurden und in vielen Fällen Fehler enthalten oder nicht vollständig sind [15,17]. Hierbei handelt es sich um typische Big Data Anforderungen, insbesondere die 3 V's: *volume* (Datenmenge), *velocity* (Änderungsgeschwindigkeit der Daten) und *variety* (Diversität der Daten). Es gibt bereits zahlreiche Ansätze für diese Herausforderungen [28,24,2,10], die jedoch derzeit eher noch im Forschungsbereich als in der praktischen Anwendung zu sehen sind.

Eine weitere Herausforderung für den Einsatz von Reasoning im Linked Data Web ist die Verbreitung stabiler und intuitiver Software und Schnittstellen. Solche Werkzeuge könnten den Endanwender von der Hürde befreien Experten für Wissensrepräsentation sein zu müssen um Anwendungen zu entwickeln. Es sind zwar bereits zahlreiche Werkzeuge verfügbar, wie z.B. [22,7,9,16,29], die jedoch auf praktische Anwendungen zugeschnitten werden müssen. Zudem werden derzeit hauptsächlich Triple Stores eingesetzt um größere Datenmengen im RDF-Format zu speichern und abfragbar zu machen. Da diese (ursprünglich) lediglich dazu dienen existierende Informationen zu ermitteln ohne Schlussfolgerungen zu ziehen, ist es notwendig parallel einen Reasoner zu betreiben um dessen Funktionalität zu erhalten. Seit kurzem ändert sich dies durch die Aus-

stattung von Triple Stores mit Reasoning-Funktionen und deren Standardisierung[2]. Außerdem wird daran gearbeitet Reasoning-Funktionalitäten als Module auf Triple Stores aufzusetzen [6,5,13,23].

5.2 Inferenz

Durch Einsatz von Reasoningverfahren kann implizites Wissen aus explizitem Wissen gefolgert werden. Ein positiver Effekt, der sich dadurch ergibt, ist dass nur Basiswissen gespeichert werden muss und sich weitere Konsequenzen aus der Ontologie ergeben. Dadurch wird die Datenhaltung vereinfacht, insbesondere muss bei der Datenaktualisierung nur das Basiswissen geändert werden statt einer Änderung aller sich daraus ergebenden Fakten. Außerdem ergibt sich gerade bei etwas komplexeren Strukturen das Problem, dass sich aus einer Menge expliziter Fakten eine Unmenge an impliziten Schlussfolgerungen ergeben, so dass es in vielen Fällen gar nicht möglich wäre diese vollständig zu speichern.

Einer Schlüsselrolle beim Reasoning kommt der Wahl der Zielsprache zu. Einfach gesagt, gilt: Je ausdrucksstärker und mächtiger die verwendete Sprache, desto höher die Komplexität der Inferenzverfahren. Sehr mächtige Sprachen, wie zum Beispiel die Prädikatenlogik sind sogar unentscheidbar, dass heißt es gibt kein Verfahren welches in endlicher Zeit bestimmen kann, ob eine Schlussfolgerung gilt. Aus diesem Grund hängt es stark vom Einsatzzweck ab welche Sprachen und Verfahren sich eignen, zum Beispiel muss bei sehr großen Wissensbasen häufig auf einfache regelbasierte Verfahren zurückgegriffen werden.

5.3 Qualitätssicherung

Einer der Haupteinsatzzwecke von Reasoningverfahren ist ist die Qualitätssicherung von strukturierten Wissensbasen. Wie oben erklärt, können zum Beispiel Tableauverfahren eingesetzt werden um Widersprüche zu finden. In der Praxis ist dies jedoch nur der erste Schritt zur Qualitätssicherung. Bei Wissensbasen realistischer Größe reicht es natürlich nicht aus zu wissen, dass eine Wissensbasis einen Widerspruch enthält. Um diese Widersprüche tatsächlich aufzulösen muss die genaue Ursache für die Widersprüche gefunden werden ("pinpointing"). Häufig werden dazu minimale Teile der Wissensbasis gesucht, die einen Widerspruch enthalten (sogenannte "justifications"). Diese sind oftmals klein genug um sie manuell zu analysieren und Widersprüche aufzulösen.

[2] `http://www.w3.org/TR/sparql11-entailment/`

Qualitätssicherung ist auch über mehrere Ontologien hinweg möglich, das heißt es können zum Beispiel mehrere Wissensbasen innerhalb eines Unternehmens miteinander kombiniert werden und somit deren Konsistenz im Zusammenspiel geprüft werden. Unter anderem eignet sich dies zur Überprüfung sogenannter `owl:sameAs` Links.

6 Zukünftige Perspektiven für Reasoning im Web of Data

Für den Bereich des Reasonings im Web of Data macht es Sinn zwischen theoretischen und praktischen Entwicklungen zu unterscheiden. Auf der theoretischen Seite zeichnet sich zunehmend ein Zusammenführen von regelbasierten und beschreibungslogik-basierten Ansätze ab. Zum Beispiel ist OWL 2 deutlich ausdrucksstärker als OWL 1, so dass man viele Regeln inzwischen mit dieser Sprache darstellen kann [20]. Auf praktischer Ebene gibt es industrielle Bemühungen um Weblogiken einzubinden, zum Beispiel deren Unterstützung in Datenbanken wie Oracle. Insgesamt gilt es hier zu beachten, dass relationale Datenbanken den Markt dominieren. Aus diesem Grund wird sehr stark daran geforscht Ontologien auf solche Datenbanken aufzusetzen. Dieser Bereich wird als *ontology-based data access* (OBDA) [7] bezeichnet. Einige dieser Systeme, z.B. OnTop, bieten sehr umfangreichen Reasoning-Support an. Dadurch wird es ermöglicht unverändert relationale Datenbanken zu betreiben und gleichzeitig die Vorteile von Ontologien und Reasoning zu nutzen, z.B. die Integration von mehreren Wissensbasen und das automatische Finden von zusätzlichem Wissen und Widersprüchen.

Danksagung

Wir bedanken uns bei Prof. Pascal Hitzler für die Diskussion zur den besonderen Herausforderungen und den zukünftigen Perspektiven von Reasoning im Linked Data Web. Die Autoren dieses Artikels werden gefördert von den dem EU FP7 Projekt GeoKnow (GA no. 318159) und dem DFG-Forschungsprojekt GOLD.

Literatur

1. Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann. Crowdsourcing linked data quality assessment. In *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*, 2013.

2. F. Baader and B. Hollunder. Embedding Defaults into Terminological Representation Systems. *J. Automated Reasoning*, 14:149–180, 1995.
3. Franz Baader. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, 2003.
4. Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres, and Dave Reynolds, editors. *RIF Core Dialect (Second Edition)*. W3C Recommendation 5 February 2013, 2013. Available from http://www.w3.org/TR/rif-core/.
5. Lorenz Bühmann and Jens Lehmann. Universal OWL axiom enrichment for large knowledge bases. In *Proceedings of EKAW 2012*, pages 57–71. Springer, 2012.
6. Lorenz Buhmann and Jens Lehmann. Pattern based knowledge base enrichment. In *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*, 2013.
7. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The MASTRO system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011.
8. Michael Compton, Payam M. Barnaghi, Luis Bermudez, Raul Garcia-Castro, Óscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory A. Henson, Arthur Herzog, Vincent A. Huang, Krzysztof Janowicz, W. David Kelsey, Danh Le Phuoc, Laurent Lefort, Myriam Leggieri, Holger Neuhaus, Andriy Nikolov, Kevin R. Page, Alexandre Passant, Amit P. Sheth, and Kerry Taylor. The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics*, 17:25–32, 2012.
9. Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn dos Santos. The Alignment API 4.0. *Semantic Web*, 2(1):3–10, 2011.
10. F.M. Donini, D. Nardi, and R. Rosati. Description Logics of Minimal Knowledge and Negation as Failure. *ACM Transactions on Computational Logic*, 3(2):177–225, 2002.
11. Paul Groth and Luc Moreau, editors. *PROV-Overview, An Overview of the PROV Family of Documents*. W3C Working Group Note 30 April 2013, 2010. Available from http://www.w3.org/TR/prov-overview.
12. Patrick Hayes. RDF Semantics. W3C Recommendation, 2004.
13. Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL class descriptions on very large knowledge bases. *International Journal on Semantic Web and Information Systems*, 5(2):25–48, 2009.
14. Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer (Second Edition)*. W3C Recommendation 11 December 2012, 2012. Available from http://www.w3.org/TR/owl2-primer/.
15. Pascal Hitzler and Frank van Harmelen. A reasonable semantic web. *Semantic Web*, 1(1–2):39–44, 2010.
16. Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11–21, 2011.
17. Krzysztof Janowicz and Pascal Hitzler. The Digital Earth as knowledge engine. *Semantic Web*, 3(3):213–221, 2012.
18. Yevgeny Kazakov, Markus Krötzsch, and František Simančík. Concurrent classification of $\mathcal{EL}$ ontologies. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist, editors, *Proceedings of the 10th International Semantic Web Conference (ISWC'11)*, volume 7032 of *LNCS*. Springer, 2011.

19. Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, et al. The automation of science. *Science*, 324(5923):85–89, 2009.
20. Markus Krötzsch. *Description Logic Rules*, volume 008 of *Studies on the Semantic Web*. IOS Press/AKA, 2010.
21. O. Lassila and R. R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation 10 February 2004, 2004. Available from http://www.w3.org/TR/REC-rdf-syntax/.
22. Jens Lehmann. DL-Learner: learning concepts in description logics. *Journal of Machine Learning Research (JMLR)*, 10:2639–2642, 2009.
23. Jens Lehmann and Lorenz Bühmann. ORE - a tool for repairing and enriching knowledge bases. In *Proceedings of the 9th International Semantic Web Conference (ISWC2010)*, Lecture Notes in Computer Science, pages 177–193. Springer, 2010.
24. Frederick Maier, Yue Ma, and Pascal Hitzler. Paraconsistent OWL and related logics. *Semantic Web*, 4(4):395–427, 2013.
25. A. Miles and S. Bechhofer, editors. *SKOS Simple Knowledge Organization System Reference*. W3C Recommendation 18 August 2009, 2009. Available from http://www.w3.org/TR/skos-reference.
26. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. Owl 2 web ontology language: Profiles. World Wide Web Consortium, Working Draft WD-owl2-profiles-20081202, December 2008.
27. Raghava Mutharaju, Pascal Hitzler, and Prabhaker Mateti. DistEL: A distributed EL+ ontology classifier. In Thorsten Liebig and Achille Fokoue, editors, *SSWS 2013, Scalable Semantic Web Knowledge Base Systems 2013. Proceedings of the 9th International Workshop on Scalable Semantic Web Knowledge Base Systems, co-located with the International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 21, 2013*, volume 1046 of *CEUR Workshop Proceedings*, pages 17–32, 2013.
28. Umberto Straccia. Reasoning within fuzzy description logics. *J. Artif. Intell. Res. (JAIR)*, 14:137–166, 2001.
29. Tania Tudorache, Csongor Nyulas, Natalya Fridman Noy, and Mark A. Musen. Webprotégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semantic Web*, 4(1):89–99, 2013.