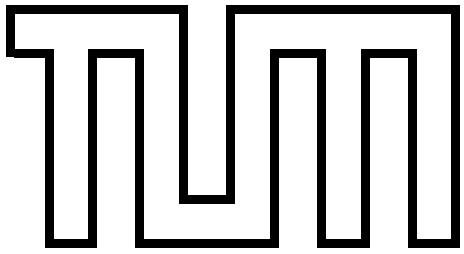

Technische Universität München

Fakultät für Informatik

Forschungs- und Lehreinheit Informatik VI

Beschreibungslogiken und OWL

Seminar Kognitive Robotik (SS12)

Stefan Profanter

Betreuer:	Alexander Perzylo, M.Sc.
Leitung:	Prof. Alois Knoll
Abgabetermin:	21. Juli 2012

Inhaltsverzeichnis

1 Einführung

Das World Wide Web (kurz Web) bietet eine umfassende Möglichkeit, Wissen, Informationen und Neuigkeiten zu verbreiten und zur Verfügung zu stellen. Es wird vor allem durch die hohe Aktualität und Verfügbarkeit ausgezeichnet.
Doch die Repräsentation des Web mit seiner unüberschaubaren Menge an Informationen ist zum Großteil auf den Menschen als Endnutzer ausgerichtet. Dies erschwert Maschinen die Auswertung und das Auffinden genau dieser Informationen.
Zur Lösung dieses Problems gibt es grundsätzlich zwei komplementäre Herangehensweisen: Zum einen wäre es denkbar, Methoden aus der künstlichen Intelligenz (KI) anzuwenden, um die kognitiven Aufgaben eines Menschen nachzubilden und die Informationen in maschinenverarbeitbare Daten zu verwandeln. Doch die bisher erreichten Resultate der KI reichen noch nicht für eine webweite zuverlässige Anwendung.
Eine alternative Herangehensweise besteht im Ansatz des Semantic Web. Es steht für die Idee, die Informationen von Beginn an so zur Verfügung zu stellen, dass diese von Maschinen einfach verarbeitet werden können. Damit hier die Interoperabilität gewährleistet ist, sind offene Standards nötig. Diese Standards wurden bereits vom *World Wide Web Consortium* (W3C) [1] unter XML, RDFS und OWL als Informations- und Spezifikationssprachen zusammengestellt.
Das Semantic Web wurde von Tim Berners-Lee, dem Erfinder des Internets, bereits im Jahre 2001 vorgestellt [3]. Dort beschreibt er das Semantic Web als ein „Web von Daten, die direkt oder indirekt von Maschinen verarbeitet werden können“.
Um die strukturierten Informationen zu verarbeiten und Schlussfolgerungen zu ziehen wird eine formale Logik, die Beschreibungslogik (BL), definiert. Beschreibungslogiken dienen dazu, Zusammenhänge und logische Beziehungen der dargestellten Informationen für Maschinen und Programme zu beschreiben. [4]

1.1 Historie

Ansätze zur Wissensrepresentation, entwickelt im den 1970ern, werden oft in zwei Kategorien unterteilt: die logikbasierten Formalismen ähnlich zur Prädikatenlogik und nicht logikbasierte Formalismen. Letztere werden als Netzwerkstrukturen auf Basis des menschlichen Gedächtnisses sowie in der Form, wie Menschen Aufgaben ausführen dargestellt. Die netzwerkbasierte Representation zielt darauf ab, Individuen und ihre Beziehungen als ein Netz darzustellen. Doch durch das Fehlen semantischer Eindeutigkeit hat sich jedes System anders verhalten, obwohl die Grunddaten identisch waren.
Ein wichtiger Schritt in Richtung Eindeutigkeit war die Verwendung von First-Order Logik (ähnlich Prädikatenlogik) (Hayes, 1979). Unäre Prädikate definierten dabei Individuen, binäre Prädikate Beziehungen zwischen diesen.

[1] `http://www.w3.org`

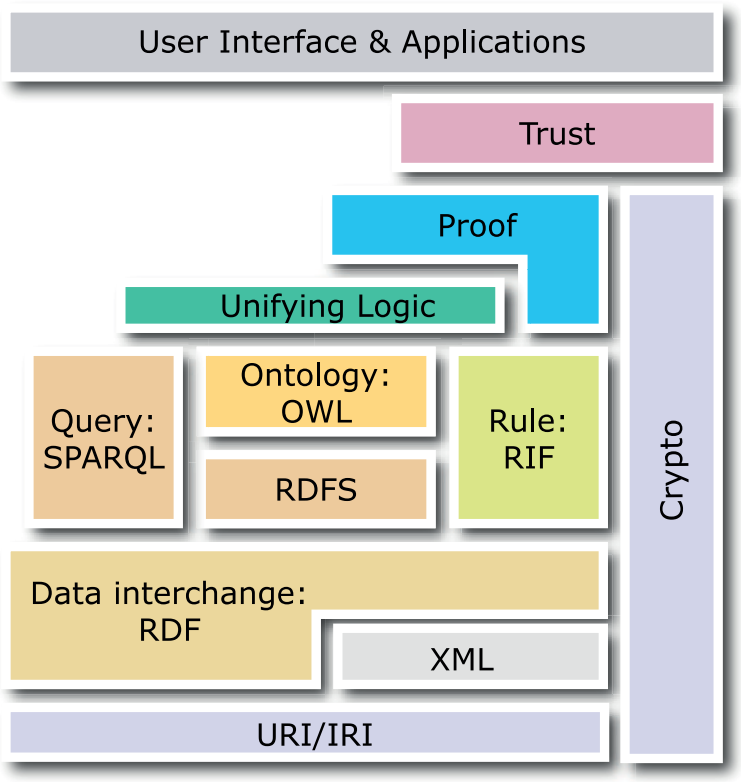

Abbildung 1: *Semantic Web Layer. Quelle: [2]*

Forschungen im Bereich der Beschreibungslogik begannen unter der Bezeichnung „teminologische Systeme“ um auszudrücken, dass die Beschreibungssprache zur Darstellung der Grund-Terminologie der beschriebenen Domäne dient. Später lag der Schwerpunkt auf konzeptgebenden Konstrukten. Daraus entstand die Bezeichnung „Konzeptuelle Sprachen“. Mit der Zeit verschob sich der Schwerpunkt auf die Eigenschaften der darunterliegenden logischen Systeme woraus dann die Bezeichnung „Beschreibungslogik“ entstand. (Siehe auch [1])

1.2 Semantic Web Pyramide

Die Semantic Web Pyramide (Abb. 1) wurde von Tim Berners-Lee eingeführt uns stellt die einzelnen Schichten des Semantic Web dar.
Im folgenden werden die Funktionen der wichtigsten Schichten kurz erläutert:

Unicode / URI

Diese Schicht ist Semantikfrei. Dabei wir Unicode zur binären Repräsentation von Zeichen verwendet. URI dient als das Identifikationsmerkmal einer virtuellen oder physischen Ressource.

XML

XML ist ein Standard zur Darstellung strukturierter Daten. Hier ist es möglich, spezielle Schemata für Daten zu definieren. XML dient als Grundlage für RDF und OWL.

RDF-Modell (RDF)

RDF ist eine Abkürzung für Resource Description Framework und dient als Datenmodell für die höheren Schichten. (Siehe auch: 4.1)

RDF-Schema (RDFS)

RDFS ist eine Vokabular-Definition zur Verwendung unter RDF um Typisierungen (Class, Resource, Property, ...) und Eigenschaften (subClassOf, subPropertyOf, ...) einheitlich darzustellen. (Siehe auch: 4.1)

Ontology: OWL

Eine Ontologie stellt ein Netzwerk von Informationen mit logischen Relationen dar. Im Bereich des Semantic Web wird dabei hauptsächlich die Web Ontology Language (OWL) als Formale Sprache zur Beschreibung von Ontologien verwendet und gewährleistet dabei vor allem die Interoperabilität von RDF Datenbasen. (Siehe auch: 4.2)

Unifying logic

Die Unifying logic dient zur einheitlichen Darstellung der Daten aus den Verschiedenen Darstellungsmodellen wie RDFS, OWL oder SPARQL. Die Unifying logic beinhaltet außerdem Inferenzmaschinen um Schlüsse aus den vorhandenen Daten zu ziehen.

Trust

Auf der Vertrauensebene soll sichergestellt werden, dass bei der Bearbeitung nur Aussagen aus vertrauenswürdigen Quellen berücksichtigt werden (RDF-Spam). Dieses Thema mag

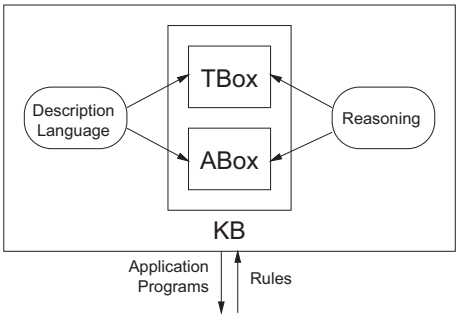

Abbildung 2: *Aufbau eines Wissensrepresentations-Systems. Quelle: [1]*

momentan noch keine große Rolle spielen, sobald aber das Semantic Web weit verbreitet ist, werden sicher einige versuchen, das System zu stören.

Proof

Die Beweisebene soll dazu dienen, die Wahrheit bestimmter Aussagen auf Basis der logischen Regeln zu überprüfen.

1.3 Weiterführende Literatur

Das bekannteste Werk zu Beschreibungslogiken wurde von Baader et. al. [1] veröffentlicht und bietet eine ausführliche Einführung in den Bereich der Beschreibungslogiken.
Unter dem Titel „Semantic Web“ wurde das deutschsprachige Buch von Pascal Hitzler et. al. [4] veröffentlicht, welches Grundlagen zum Semantic Web sowie zu RDFS und OWL vermittelt.
Das Online Tutorial „Ontologies and Semantic Web“ von Marek Obitko [2] bietet vor allem anschauliche Beispiele zu den Themen des Semantic Web und Ontologien.

2 Sprachen und Modellierung

Ein System zur Wissensrepresentation (siehe Abb. 2) besteht aus mehreren Komponenten: Die Wissensbasis setzt sich aus zwei Komponenten, der TBox und der ABox, zusammen. Die TBox beinhaltet terminologisches Wissen, die ABox assertionales Wissen.
Um die Daten der Wissensbasis zu beschreiben, verwendet man eine Beschreibungssprache. Alle Sprachen der Beschreibungslogik bestehen aus einem Grundvokabular: Den Konzepten (concepts) und Rollen (roles). Konzepte sind dabei Mengen von Objekte und Rollen binäre Beziehungen zwischen diesen Objekten.

[2] `http://www.obitko.com/tutorials/ontologies-semantic-web/introduction.html`

Außerdem gibt es die Möglichkeit, aus dem vorhandenen Wissen neues Wissen, also neue Konzepte und Rollen, durch Reasoning zu bilden.
Damit dises Wissen auch für andere Systeme zur Verfügung steht, ist zudem eine Schnittstelle nach außen verfügbar, welche zusätzlich die Möglichkeit bietet, neues Wissen in die Wissensbasis einzufügen oder Wissen auszulesen.
Beispiel:
Konzepte: Professor, Student, Vorlesung, Seminar
Rollen: besucht, hält
TBox: FaulerStudent := Student, der keine Vorlesung besucht
ABox: Knoll ist ein Professor, KogRob ist ein Seminar, Stefan ist ein Student, Stefan besucht KogRob

2.1 Konzepte und Rollen

Als Grundlage für die meisten Sprachen der Beschreibungslogik dient eine abstrakte Sprache, bestehend aus zwei Mengen disjunkter Symbole, um *atomare Konzepte* und *atomare Rollen* darzustellen. Ein Ausdruck wird dann mit Hilfe von Konstruktoren aus den Grundsymbolen gebildet. Ein Konstruktor ist zum Beispiel die *Schnittmenge von Konzepten*, dargestellt als $C \sqcap D$. Bei der Schnittmenge sind nur Konzepte zulässig, welche zu beiden der angegebenen Konzepte gehören. Somit bildet ein Konstruktor aus atomaren Konzepten neue nicht atomare Konzepte.
Die wichtigste Eigenschaft der Beschreibungslogik ist das Darstellen von Beziehungen zwischen den Konzepten. Diese dienen dazu, aus einer Kombination von Rollen und Konzepten neue Konzepte zu erstellen.
Als ein Beispiel definieren wir `Weiblich`, `Person` und `Frau` als atomare Konzepte, `hatKind` und `hatWeiblicheVerwandte` als atomare Rollen. Durch Verwendung von Schnittmenge, Vereinigung und Komplement lässt sich das Konzept „nicht weibliche Person“ als $\texttt{Person} \sqcap \neg\texttt{Weiblich}$ und „Personen die männlich oder weiblich sind“ als $\texttt{Weiblich} \sqcup \texttt{Männlich}$ darstellen.
Zudem existieren Beschränkungen der Anzahl an Konzepten. Zum einen gibt es den Existenz- und Allquantor: $\exists\texttt{hatKind}.\texttt{Weiblich}$ bezeichnet alle Personen, die ein weibliches Kind haben, zum anderen Beschränkungen in der Kardinalität: $(\geq 3\texttt{hatKind}) \sqcap (\leq 2\texttt{hatWeiblicheVerwandte})$ bezeichnet „Personen die mindestens drei Kinder haben und maximal zwei weibliche Verwandte“.
Es gibt auch die Möglichkeit, Rollen zu verknüpfen: „hat Tochter“ wird zu $\texttt{hatKind} \sqcup \texttt{hatWeiblicheVerwandte}$.

2.2 TBox und ABox Axiome

TBox und ABox ist die Wissensbasis einer Beschreibungslogik. Die TBox beinhaltet wissen über Konzepte einer Domäne, die ABox hingegen enthält das Wissen über Entitäten oder

Instanzen dieser Konzepte sowie deren Beziehungen untereinander. Somit ist das TBox Wissen ein eher statisches Wissen. Das ABox Wissen hingegen meist dynamisch. Diese Unterscheidung lässt sich bei ausdrucksstärkeren Beschreibungslogiken nur mehr schwer durchführen.

TBox

In der TBox werden hauptsächlich neue Konzepte durch Kombination von bereits existierenden Konzepten gebildet. Zum Beispiel kann das Konzept „Frau“ definiert werden als: `Frau` $\equiv$ `Person` $\sqcap$ `Weiblich`.
Diese Form der Konzeptbildung bietet große Freiheit. Trotzdem muss man folgende Einschränkungen beachten:

- es ist nur eine Definition pro Konzept zugelassen
- Konzepte müssen azyklisch sein. D.h. sie dürfen weder auf sich selbst noch auf Konzepte, die indirekt oder direkt das Konzept verwenden, verweisen.

Dadurch ist es möglich, alle Konzepte der TBox in die entsprechenden atomaren Konzepte zu erweitern.
Eine wichtige Aufgabe der TBox ist die Bildung einer Terminologie, also die Klassifikation von Konzepten. Klassifikation wird erreicht durch auswerten der Subsumtion (siehe 3.1).

ABox

Die ABox beinhaltet Aussagen über Individuen bzw. Instanzen. `Weiblich`$\sqcap$`Person(ANNA)` bedeutet zum Beispiel, dass ANNA eine weibliche Person ist. Aus vorheriger Definition einer Frau kann hergeleitet werden, dass ANNA eine Frau ist. Auch Aussagen durch Verwendung von Rollen sind zulässig: `hatKind(ANNA,PETER)` sagt aus, dass PETER ein Kind von ANNA ist.
Die Hauptaufgabe im Reasoning im Kontext der ABox ist die *Überprüfung von Instanzen*, also die Überprüfung, ob ein gegebenes Individuum eine Instanz eines angegebenen Konzeptes ist. Weitere, auf Überprüfung von Instanzen aufbauenden Reasoning Aufgaben sind: *Konsistenz der Wissensbasis*: Überprüfung ob jedes Konzept mindestens ein Individuum besitzt, *Realisierung*: Finden des spezifischsten Konzepts eines Individuums, und das Suchen von Individuen zu einem gegebenen Konzept (siehe 3.1).

2.3 $\mathcal{AL}$ Beschreibungssprache

Bei der Sprache $\mathcal{AL}$ handelt es sich um eine der ersten und wichtigsten Beschreibunssprachen, die die Grundlage für viele der nachfolgenden Sprachen bildet.
Elementare Beschreibungen sind `atomare Konzepte` und `atomare Rollen`. Komplexe

Beschreibungen können mit Hilfe von `Konzept Konstruktoren` gebildet werden. Für die abstrakte Notation verwenden wir A und B für atomare Konzepte, R für atomare Rollen und C und D für Konzept Beschreibungen.
Beschreibungssprachen unterscheiden sich vor allem an den Konstruktoren. In der Sprache $\mathcal{AL}$ existieren folgende Konstruktoren:

$C, D \rightarrow A\|$	(atomares Konzept)
$\top\|$	(universelles Konzept)
$\bot\|$	(Grundkonzept)
$\neg A\|$	(atomare Negation)
$C \sqcap D\|$	(Schnittmenge)
$\forall R.C\|$	(Allquantor)
$\exists R.\top$	(eingeschränkter Existenzquantor)

In $\mathcal{AL}$ ist die Negation nur auf atomare Konzepte zulässig. Um diese Konzepte formell mit First-Order-Logik zu beschreiben, definieren wir Interpretationen $\mathcal{I}$ als eine nicht leere Menge $\Delta^{\mathcal{I}}$ (die Domäne der Interpretation) und die Interpretationsfunktion, welche jedem atomaren Konzept A eine Menge $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ zuweist und eine Rolle R als eine binäre Relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} | \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} | \exists b.(a,b) \in R^{\mathcal{I}}\}
\end{aligned}
$$

C und D sind equivalent $(C \equiv D)$, wenn $C^{\mathcal{I}} = D^{\mathcal{I}}$ für alle Interpretationen $\mathcal{I}$

Als direkte Erweiterung für die Sprache $\mathcal{AL}$ existieren: $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}]$ (geschrieben als z.B. $\mathcal{ALUE}$,$\mathcal{ALC}$). Das bedeutet, dass die Sprache $\mathcal{AL}$ durch hinzufügen weiterer Konstruktoren erweitert wird:

- $\mathcal{U}$ (Vereinigung)
  $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $\mathcal{E}$ (volle Existenzquantor)
  $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$

- $\mathcal{N}$ (Beschränkungen in der Kardinalität)
  $(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \ |\{b|(a,b) \in R^{\mathcal{I}}\}| \geq n\}$
  $(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \ |\{b|(a,b) \in R^{\mathcal{I}}\}| \leq n\}$
- $\mathcal{C}$ (Erweiterte Negation), auch für nicht atomare Konzepte
  $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$

3 Reasoning

Reasoning wird dazu verwendet, um aus der vorhandenen Wissensbasis neue Schlüsse zu ziehen oder Konzepte aus anderen Systemen zu verifizieren. Der Haupt-Inferenztyp ist die Subsumtion (Unterordnung), typischerweise geschrieben als $C \sqsubseteq D$. Beim Überprüfen einer Subsumtion wird bestimmt, ob das Konzept D allgemeiner als das Konzept C ist. Ein weiterer Inferenztyp ist die Erfüllbarkeit: sie gibt an, ob ein Konzept erfüllbar ist, also keine leere Menge darstellt.

3.1 Inferenzprobleme (TBox / ABox)

Im Bereich der Wissensrepresentation bedeutet Inferenz, das Wissen der Wissensbasis durch Schlussfolgerungen (*reasoning*) zu extrahieren.

TBox

Die Inferenzprobleme der TBox können auf zwei Haupt-Inferenzprobleme reduziert werden. Somit müssen nur diese zwei untersucht werden. Dabei sind C und D jeweils Konzepte der TBox.

Reduzierung auf Subsumierung (Subsumption):

Erfüllbarkeit (Satisfiability): C ist unerfüllbar $\Leftrightarrow$ C wird subsumiert von $\bot$

Äquivalenz (Equivalence): C und D sind äquivalent $\Leftrightarrow$ C wird subsumiert von D und D wird subsumiert von C

Disjunktheit (Disjoint): C und D sind disjunkt $\Leftrightarrow$ $C \sqcap D$ wird subsumiert von $\bot$

Reduzierung auf Unerfüllbarkeit (Unsatisfiability)

Subsumierung (Subsumption): C wird subsumiert von $D \Leftrightarrow C \sqcap \neg D$ ist unerfüllbar

Äquivalenz (Equivalence): C und D sind äquivalent $\Leftrightarrow (C \sqcap \neg D)$ und $(\neg C \sqcap D)$ sind unerfüllbar

Disjunktheit (disjoint): C und D sind disjunkt $\Leftrightarrow C \sqcap D$ ist unerfüllbar

ABox

Die Inferenzprobleme der ABox, Konsistenz und Instanzüberprüfung, lassen sich auf eine einzige Überprüfung der Konsistenz reduzieren:

Konsistenz (Consistency) ABox A is konsistent bzgl. einer TBox $T \Leftrightarrow$ Für jedes Konzept aus T existiert mindestens ein Individuum in A.

Instanzüberprüfung (Instance check) $a^{\mathcal{I}} \in C^{\mathcal{I}}$ gilt für alle Modelle $A = C(a) \Leftrightarrow A \cup \neg C(a)$ ist inkonsistent.

3.2 open-world - closed-world

Oft werden Datenbanken mit BL Wissensbasen verglichen. Das Schema einer Datenbank entspricht der TBox, die Einträge in der Datenbank der ABox. Die Semantic einer ABox unterscheided sich aber von Datenbank Instanzen: Während Datenbank Instanzen genau eine Interpretation beschreiben, also das definierte Schema mit den Relationen, repräsentiert eine ABox viele unterschiedliche Interpretationen, nämlich alle verschiedenen Modelle. Daraus folgt, dass fehlende Daten in der Datenbank als negative Information interpretiert wird, bei der ABox hingegen bedeutet fehlende Information ein Mangel an Wissen.
Ein Beispiel:
Gegeben ist folgende Beziehung $hatKind(PETER, HARRY)$ in der Datenbank und in der ABox. Aus der Datenbank schlißt man daraus, dass Peter nur ein Kind hat. In der ABox hingegen bedeutet dies, dass Harry ein Kind von Peter ist (dies bedeutet nur, dass Peter mindestens ein Kind hat). Die ABox hat zusätzlich mehrere Modelle: eines in dem Peter ein Kind hat, und eines in dem Harry Geschwister hat. Die einzige Möglichkeit, in der ABox zu definieren, dass Peter nur ein Kind hat ist: $(\leq 1 hatKind)(PETER)$.
Dies bedeutet, dass Informationen in einer Datenbank immer als vollständig angesehen werden, wohingegen Informationen in der ABox unvollständig sind. Man spricht daher auch von *closed-world* (Datenbank) und *open-world* (ABox).

3.3 Tableau-Algorithmen

Tableau-Algorithmen dienen dazu, in einer Wissensbasis logische Konsequenzen zu finden und die Erfüllbarkeit der Wissensbasis zu bestimmen. Als Grundlage für diese Algorithmen

dient die *Negationsnormalform*:

$$
\begin{aligned}
NNF(C) &= C && \text{(falls C atomar ist)}\\
NNF(\neg C) &= \neg C && \text{(falls C atomar ist)}\\
NNF(\neg\neg C) &= NNF(C)\\
NNF(C \sqcup D) &= NNF(C) \sqcup NNF(D)\\
NNF(C \sqcap D) &= NNF(C) \sqcap NNF(D)\\
NNF(\neg(C \sqcup D)) &= NNF(\neg C) \sqcap NNF(\neg D)\\
NNF(\neg(C \sqcap D)) &= NNF(\neg C) \sqcup NNF(\neg D)\\
NNF(\forall R.C) &= \forall R.NNF(C)\\
NNF(\exists R.C) &= \exists R.NNF(C)\\
NNF(\neg\forall R.C) &= \forall R.NNF(\neg C)\\
NNF(\neg\exists R.C) &= \exists R.NNF(\neg C)
\end{aligned}
$$

Die Wissensbasis muss zuerst in die Negationsnormalform transformiert werden. Elemente der Form $C \equiv D$ werden durch die Elemente $C \sqsubseteq D$ und $D \sqsubseteq C$ ersetzt und $C \sqsubseteq D$ durch $\neg C \sqcup D$. Dadurch werden alle Symbole $\equiv$ und $\sqsubseteq$ eliminiert. Jetzt wird jedes C durch $NNF(C)$ und jedes $C(a)$ durch $NNF(C)(a)$ ersetzt. Diese Umformung gewährleistet, dass das Negationszeichen nur noch vor atomaren Klassenbezeichnern steht. Dadurch kann der Tableau-Algorithmus übersichtlicher dargestellt werden.
Die grundsätzliche Idee hinter dem Tableau-Algorithmus ist, aus einer Wissensbasis so lange logische Konsequenzen zu bilden, bis ein Widerspruch entsteht. Als Beispiel (siehe auch [4, 179]) sei folgende Wissensbasis gegeben:
$W = \{C(a), \neg C \sqcup D, \neg D(a)\}$

1. $C(a)$ und $\neg D(a)$ sind triviale Konsequenzen und ergeben $T_1 = \{C(a)\ ,\ \neg D(a)\}$
2. Aus $\neg C \sqcup D$ folgt $(\neg C \sqcup D)(a)$. Dies ergibt $T_2 = \{C(a)\ ,\ \neg D(a), (\neg C \sqcup D)(a)\}$
3. $(\neg C \sqcup D)(a)$ bedeutet, dass mindestens eine der Aussagen $\neg C(a)$ oder $D(a)$ wahr sein muss. Da nicht bekannt ist, welche dies ist, müssen beide getrennt untersucht werden: $T_{3,1} = \{C(a)\ ,\ \neg D(a), (\neg C \sqcup D)(a), \neg C(a)\}$ und $T_{3,2} = \{C(a)\ ,\ \neg D(a), (\neg C \sqcup D)(a), D(a)\}$
   $T_{3,1}$ enthält sowohl $C(a)$ und $\neg C(a)$ und somit einen Widerspruch. $T_{3,2}$ enthält sowohl $D(a)$ und $\neg D(a)$ und somit auch einen Widerspruch. Jetzt wurde für alle Alternativen ein Widerspruch abgeleitet und die Suche kann beendet werden.

Daraus folgt die Konsequenz der Wissensbasis $T_2 = \{C(a)\ ,\ \neg D(a), (\neg C \sqcup D)(a)\}$. Bekannte Tableau-Algorithmen wie FaCT, FaCT++, RACER, DLP und Pelle implementieren analytische Tableau-Algorithmen [3].

[3] `https://en.wikipedia.org/wiki/Analytic_tableau_method`

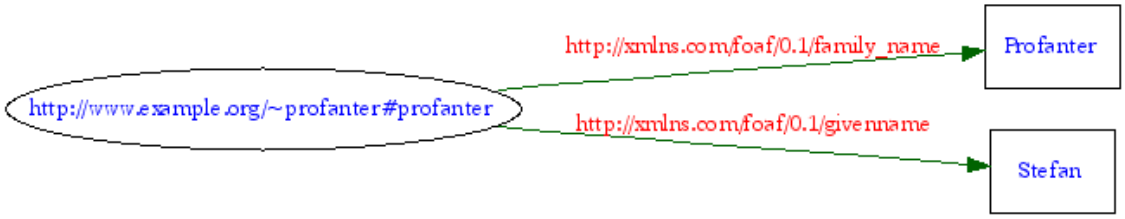

Abbildung 3: *Beispiel Graph für RDF. Generiert mit:* `http://www.w3.org/RDF/Validator/`

4 Ontologien, RDF und OWL

Eine Ontologie ist eine sprachlich gefasste und formal geordnete Darstellung einer Menge von Begrifflichkeiten und der Beziehung zwischen diesen Begrifflichkeiten. Ontologien werden hauptsächlich dazu verwendet, Wissen unter verschiedenen Systemen auszutauschen [4].
Zur Darstellung einfacher Ontologien wird meist das auf XML aufbauende RDF und RDFS verwendet. Eine Erweiterung von RDF stellt die Web Ontology Language (OWL) dar.
Für die Abfrage von Daten aus RDFS und OWL steht die Abfragesprache *Simple Protocol and RDF Query Language* (SPARQL) zur Verfügung.

4.1 RDF und RDFS

RDF steht für *Resource Description Framework* und beschreibt ein Framework zur Darstellung von Informationen in Form von Graphen. Dazu werden Tripel der Form *Subjekt-Prädikat-Objekt* verwendet. Die einzelnen Elemente des Tripel können dabei frei definiert werden. Subjekte werden in RDF mit Hilfe von URI (Unified Resource Identifier) identifiziert. Da die Tripel frei definierbar sind, wurde das RDF Schema (RDFS) erstellt, welche die Syntax zu Beschreibung von Taxonomien standardisiert.

Als Beispiel definieren wir zwei Tripel, welche den Vor- und Nachnamen einer Person, identifiziert durch die URI: `http://www.example.org/~profanter#profanter`, angibt (siehe auch Abb. 3 für den entsprechenden Graph). Die RDF-Darstellung serialisiert in XML lautet:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <rdf:Description rdf:about="http://www.example.org/~profanter#profanter"
     >
         <foaf:family_name>Profanter</foaf:family_name>
```

[4] `https://de.wikipedia.org/wiki/Ontologie_(Informatik)`

```
        <foaf:givenname>Stefan</foaf:givenname>
  </rdf:Description>
</rdf:RDF>
```

Anstatt XML existieren auch noch andere Serialisierungssprachen wie Turtle oder N3.

RDFS

RDFS erweitert den RDF Wortschatz um Taxonomien über Klassen und Eigenschaften zu beschreiben. Zusätzlich erlaubt RDFS auch die Angabe eines Wertebereiches. Ein Beispiel für RDFS lautet:

```
<?xml version="1.0"?>
<rdf:RDF xml:lang="de"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.example.com/vehicles.rdf#">
  >

  <rdf:Description ID="MotorFahrzeug">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
  </rdf:Description>

  <rdf:Description ID="PKW">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#MotorFahrzeug"/>
  </rdf:Description>

  <rdf:Description ID="LKW">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#MotorFahrzeug"/>
  </rdf:Description>
</rdf:RDF>
```

In RDFS werden Resourcen in Klassen (hier: Class, Resource, MotorFahrzeug) eingeteilt. Diese Klassen besitzen jeweils Eigenschaften (hier: subClassOf). Eigenschaften beschreiben somit eine Beziehung zwischen dem Subjekt und Objekt. Instanzen einer Klasse werden mit `rdf:type` dargestellt. Ein vollständige Liste zu den möglichen Klassen und Eigenschaften findet sich unter `http://www.w3.org/TR/rdf-schema/`.

4.2 OWL

Die Web Ontoloty Language (OWL) erweitert die Syntax von RDF bzw. RDFS um die Konzepte der Beschreibungslogiken. OWL gibt es in drei verschiedenen Versionen:

OWL Lite wird für Taxonomien und einfachen Ontologien verwendet, wobei bestimmte Sprachkonstrukte aus OWL DL nicht vorhanden sind.

OWL DL beinhaltet die komplette Unterstützung für Beschreibungslogiken (Description Logic, DL).

OWL Full führt das Konzept zur freien Formulierung der Syntax aus RDF fort und bietet somit maximale Freiheit zur Darstellung von Ontologien.

OWL ist sehr umfangreich. Zur Einführung wird hier ein kurzes Beispiel gegeben: Basierend auf dem OWL Pizza Beispiel [5] wird folgende Beschreibungslogik (Englisch) definiert (aus [5]):
„Pizza hat PizzaBase als basis; Pizza ist disjunkt mit PizzaBase; NonVegetarianPizza ist eine Pizza welche keine VegetarianPizza ist; isIngredientOf ist eine transitive Eigenschaft; isIngredientOf ist das inverse von hasIngredient“
$Pizza \sqsubseteq \exists hasBase.PizzaBase$
$Pizza \sqcap PizzaBase \equiv \bot$
$NonVegetarianPizza \equiv Pizza \sqcap \neg VegetarianPizza$
$Tr(isIngredientOf)$
$isIngredientOf \equiv hasIngredient^{-}$

Ausgedrückt in OWL serialisiert als XML

```
<rdf:RDF xmlns="http://example.com/pizzas.owl#"
   xmlns:log="http://www.w3.org/2000/10/swap/log#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

   <rdf:Description rdf:about="http://example.com/pizzas.owl#
      NonVegetarianPizza">
       <owl:equivalentClass rdf:parseType="Resource">
           <owl:intersectionOf rdf:parseType="Resource">
               <rdf:first rdf:parseType="Resource">
                   <owl:complementOf rdf:resource="http://example.com/
                      pizzas.owl#VegetarianPizza"/>
               </rdf:first>
```

[5] http://www.co-ode.org/resources/papers/ekaw2004.pdf

```
                <rdf:rest rdf:parseType="Resource">
                    <rdf:first rdf:resource="http://example.com/pizzas.owl#
                        Pizza"/>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf
                        -syntax-ns#nil"/>
                </rdf:rest>
            </owl:intersectionOf>
        </owl:equivalentClass>
    </rdf:Description>

    <rdf:Description rdf:about="http://example.com/pizzas.owl#Pizza">
        <rdfs:subClassOf rdf:parseType="Resource">
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
                Restriction"/>
            <owl:onProperty rdf:resource="http://example.com/pizzas.owl#
                hasBase"/>
            <owl:someValuesFrom rdf:resource="http://example.com/pizzas.owl
                #PizzaBase"/>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="http://example.com/pizzas.owl#
            PizzaBase"/>
    </rdf:Description>

    <owl:ObjectProperty rdf:about="http://example.com/pizzas.owl#
        isIngredientOf">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
            TransitiveProperty"/>
        <owl:inverseOf rdf:resource="http://example.com/pizzas.owl#
            hasIngredient"/>
    </owl:ObjectProperty>
</rdf:RDF>
```

4.3 SPARQL

SPARQL [6] (”sparkle”, Akronym für SPARQL Protocol and RDF Query Language) ist eine SQL ähnliche Sprache zur Abfrage und Manipulation von Daten aus RDF und OWL Datenbanken.
Folgende Abfrage liefert den Namen und die E-Mail Adresse aller Personen in der Datenbank:

[6] https://en.wikipedia.org/wiki/SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}
```

SPARQL untersützt folgende Abfragen:

SELECT Abfrage der Daten, Ergebnisse in Tabellenformat

CONSTRUCT Abfrage der Daten, Ergebnisse in RDF-Format

ASK Liefert Wahr oder Falsch für die Abfrage

DESCRIBE Liefert den RDF Graph für die Abfrage. Der Detailgrad hängt dabei vom verwendeten Backend ab.

SPARUL (SPARQL Update) ist eine Erweiterung zu SPARQL welche zusätzlich das Hinzufügen, Ändern und Löschen von Daten unterstützt.

5 Anwendung

Beschreibungslogik wird bereits in sehr vielen Bereichen eingesetzt. Hier ein paar Beispiele, um ein besseres Bild davon zu bekommen, wöfür Beschreibungslogiken verwendet werden können:

- Konfiguration
  BL wird bereits erfolgreich in Konfigurationsaufgaben eingesetzt. Ein einfaches Beispiel dafür ist das zusammenstellen eines Heim-PCs. Wenn die Komplexität von Anzahl, Typen und Rollen steigt, kann eine Konfigurationsaufgabe sehr schnell sehr komplex werden. Mit Hilfe der BL kann hier überprüft werden, ob eine Konfiguration gültig ist oder es können Konfigurationen nach bestimmten Vorgaben erstellt werden.
- Medizin
  In der Medizin-Praxis ist vor allem die Unterstützung für Diagnose-Entscheidungen der Ärzte eine Aufgabe der BL. Hier existieren bereits sehr ausgeprägte Wissensbasen die oft mit Hilfe öffentlicher Fördergelder aufgebaut wurden.
- Verarbeitung natürlicher Sprache
  Hier dient die BL vor allem als semantisches Wissen welches dazu verwendet wird, um die Bedeutung eines Satzes zu bestimmen. Aber nicht nur zum Verstehen von Sätzen sondern auch zum generieren von natürlicher Sprache wird BL eingesetzt.

- Robotik
  Immer mehr Anwendung findet die BL im Bereich der Robotik. Hier dient sie vor allem zur Darstellung und dem Austausch von Wissen eines Roboters. Siehe dazu auch knowrob [7] und das RoboEarth Projekt [8]

[7] http://www.ros.org/wiki/knowrob
[8] http://www.roboearth.org

Literatur

[1] Franz Baader, Diego Calvanese, Peter Patel-Schneider, Deborah L. McGuinness, and Daniele Nardi, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, Jan. 2003.

[2] Tim Berners-Lee. Semantic web layer cake (png-grafik, 600 × 630 pixel). `http://www.w3.org/2007/03/layerCake.png`, 30.03.2007.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: Scientific american: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. `http://www.scientificamerican.com/article.cfm?id=the-semantic-web`, 17.05.2001.

[4] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure, editors. *Semantic Web*. Springer, Berlin and Heidelberg and New York and NY, 2007.

[5] Marek Obitko. Introduction - introduction to ontologies and semantic web - tutorial. `http://www.obitko.com/tutorials/ontologies-semantic-web/introduction.html`, 2007.