Konjunktive Anfragen und Regelsprachen

Pascal Hitzler Markus Krötzsch Sebastian Rudolph

Institut AIFB · Universität Karlsruhe

Semantic Web Technologies 1 (WS07/08)
23. Januar 2008
http://semantic-web-grundlagen.de

Semantic Web Technologies 1

1. Einleitung und Ausblick
2. XML und URIs
3. Einführung in RDF
4. RDF Schema
5. Logik – Grundlagen
6. Semantik von RDF(S)
7. OWL – Syntax und Intuition
8. OWL – Semantik und Reasoning
9. SPARQL – Syntax und Intuition
10. Semantik von SPARQL
11. **Konjunktive Anfragen und Regelsprachen** (→ Webseite)
12. OWL 1.1 – Syntax und Semantik
13. Bericht aus der Praxis
14. Semantic Web – Anwendungen

Literatur zu dieser Vorlesung online siehe
→ Semantic Web – Grundlagen, Kapitel 7 und → Vorlesungswebseite

Die Grenzen von OWL

OWL-Konzepte als Anfragesprache ungenügend:

- „Welche Paare von Personen haben ein gemeinsames Elternteil?“
- „Welche Personen wohnen bei einem ihrer Eltern?“
- „Welche Paare (direkter oder indirekter) Nachkommen gibt es?“

Relevante Informationen nicht in OWL-Ontologie darstellbar:

- „$(\forall x)(\forall y)(\forall z)\ (\text{bruder}(y,z) \wedge \text{vater}(x,y) \rightarrow \text{onkel}(x,z))$“
- „$(\forall x)\ (\text{liebt}(x,x) \rightarrow \text{Narzist}(x))$“

OWL ungeeignet zur Programmierung:

- OWL ist entscheidbar: es kann grundsätzlich nicht alles Programmierbare ausdrücken (*Halteproblem*).
- OWL wird nicht „abgearbeitet“, es ist *nicht prozedural*: Bestimmte Erweiterungen (Built-ins) sind nur schwer zu realisieren.

Übersicht über nächsten Vorlesungen

Vorlesung 11:

- Ausdrucksstarke Anfragen für OWL
  ⇝ **Konjunktive Anfragen**
- Erweiterung von OWL zur Wissensrepräsentation
  ⇝ **SWRL, Datalog, und DL-safe Rules**

Vorlesung 12:

- Aktuelle Erweiterung der OWL-Sprache
  ⇝ **OWL 1.1**

Semantic Web Technologies 2:

- Logikprogrammierung als semantische Technologie
  ⇝ **F-Logik**

Ausdrucksstarke Anfragen für OWL?

SPARQL als Anfragesprache für RDF
$\rightsquigarrow$ keine direkte Unterstützung für OWL

Anfrageformalismus für OWL DL: **konjunktive Anfragen**

- keine offizielle Spezifikation, keine normative Syntax
  $\rightsquigarrow$ uns reichen hier Kurzschreibweisen anstelle von vollen URIs
- Ziel: ausdrucksstärkere Anfragen nach *Individuen*
- keine Betrachtung von Formatierung oder Nachbearbeitung der Ergebnisse
- praktische Bedeutung für Anwendungen
- verschiedene Implementationen verfügbar

Syntax und Intuition

Konjunktive Anfragen sind recht einfach!

Beispiel:

$$\texttt{Buch}(x) \wedge \texttt{VerlegtBei}(x, \texttt{Springer}) \wedge \texttt{Autor}(x, y)$$

„Welche Bücher sind bei Springer erschienen und wer hat sie geschrieben?“

- Syntax angelehnt an Prädikatenlogik
- Hauptelemente: Bezeichner von Rollen/Klassen/Individuen, Variablen, Konjunktion $\wedge$

Semantik konjunktiver Anfragen

Konjunktive Anfragen ähneln logischen Formeln
$\rightsquigarrow$ Anfragen ohne Variablen können aus einer Ontologie *folgen*
$\rightsquigarrow$ Variablen als Platzhalter für Bezeichner von Individuen

Funktion μ is **Lösung einer konjunktiven Anfrage** q für eine Ontologie O, falls gilt:

1. Domäne von μ ist die Menge der freien Variablen in q
2. Wertebereich von μ ist die Menge der Individuenbezeichner in O
3. $O \models \mu(q)$, d.h. q mit dieser Variablenbelegung folgt aus O

$\rightsquigarrow$ keine partielle Funktion – alle Variablen müssen belegt sein
$\rightsquigarrow$ Literale (Datentypen) hier zur Vereinfachung nicht betrachtet

Unbenannte Elemente

Variablen bisher als Platzhalter für (benannte) Individuen,

aber

OWL-Ontologien können auch die **Existenz unbenannter Elemente** implizieren

Beispiel:

$\texttt{Buch}(a)$	(a ist ein Buch)
$\texttt{Buch} \sqsubseteq \exists \texttt{Autor}.\top$	(jedes Buch hat einen Autor)

$\rightsquigarrow$ Anfrage $\texttt{Buch}(x) \wedge \texttt{Autor}(x, y)$ hat keine Lösung!

Unbestimmte Variablen

Wie können Anfragen auch unbenannte Individuen berücksichtigen?

- unbenannte Elemente können kaum als Teil der Lösung ausgegeben werden
- wir wollen nur die *Existenz* geeigneter Elemente fordern

$\rightsquigarrow$ **Unbestimmte Variablen** werden durch Existenzquantoren gebunden

Beispiel:

$\texttt{Buch}(a)$	(a ist ein Buch)
$\texttt{Buch} \sqsubseteq \exists \texttt{Autor}.\top$	(jedes Buch hat einen Autor)

Anfrage $\exists y.(\texttt{Buch}(x) \wedge \texttt{Autor}(x, y))$

$\rightsquigarrow$ Lösung $\{x \mapsto a\}$, aber y nicht Teil der Lösung

Variablen in SPARQL und in konjunktiven Anfragen

Kennzeichen verschiedener Arten von Anfragevariablen:

- Unbenannte Werte: Sind Werte möglich, die keine Bezeichner (URI/Literal) haben?
- Ausgabe: Erscheint die Variable in Lösungen der Anfrage?

Unterschiedliche Formen von Variablen:

	Unbenannte Werte	Ausgabe
Bestimmte Variable	—	Ja
Unbestimmte Variable	Ja	—
SPARQL-Variable	Ja	Ja
Leerer Knoten in SPARQL	Ja	—
Nicht ausgewählte SPARQL-Variable	Ja	—

$\rightsquigarrow$ „SPARQL für OWL“:
leere Knoten der Ontologie: Individuen oder unbenannte Elemente?

Vergleich mit SPARQL

SPARQL	**konjunktive Anfragen**
Muster in Graphen	logische Konjunktionen
ein kanonisches Modell	viele mögliche Modelle
Optionen, Alternativen, Filter	—
Abfrage beliebiger Elemente, z.B. von Property-Bezeichnern	nur Abfrage von Instanzen (strikte Typung)
Variablen für beliebige Elemente	Bestimmte und unbestimmte Variablen

„SPARQL für OWL“ ist möglich:

- Darstellung logischer Konsequenzen als Graph
- Typung wie bei OWL DL, ev. Erweiterung um in OWL übliche Anfragen (z.B. Klassenhierarchie)
- Inkompatibilitäten bei Variablensemantik müssen akzeptiert werden

Erweiterungen konjunktiver Anfragen

Mögliche Erweiterungen konjunktiver Anfragen:

- **Filter, Modifikatoren, Ergebnisformate:** Definition wie in SPARQL möglich, unabhängig von Anfrage (Filter unproblematisch wenn kein `OPTIONAL`)
- **Negation:** Zulassung von $\neg$ vor Anfrageausdrücken
- **Disjunktionen:** Zulassung von $\vee$, entspricht `UNION` in SPARQL, „disjunktive Anfragen“
- **Komplexe Pfadausdrücke:** Reguläre Ausdrücke zur Beschreibung von Mustern aus Rollen, z.B. Anfrage nach allen Vorfahren einer Person (enthält beliebig lange Kette aus Rolle „KindVon“)

Komplexität und Entscheidbarkeit

Schlussfolgern mit OWL DL ist sehr komplex (NEXPTIME-vollständig)
⇝ Wie schwierig sind dann konjunktive Anfragen?

Bisher noch nicht abschließend geklärt!

- Konjunktive Anfragen für $\mathcal{SHIQ}$ (und für OWL Lite): 2EXPTIME-vollständig!
- Konjunktive Anfragen für $\mathcal{SHOQ}$: entscheidbar in 2EXPTIME
- Konjunktive Anfragen für $\mathcal{SHOIQ}$ (und für OWL DL): Entscheidbarkeit nicht bekannt!

⇝ konjunktive Anfragen für OWL sind äußerst kompliziert

Implementationen

Implementationen von konjunktiven Anfragen für OWL verfügbar (23.1.2008)

- **KAON2**: konjunktive Anfragen ohne unbestimmte Variablen, eingeschränkte Negation zulässig (→ Webseite)
- **Pellet**: konjunktive Anfragen mit unbestimmten Variablen und Negationen, nicht vollständig für OWL DL (→ Webseite)
- weitere Systeme mit speziellen Anfragesprachen (**RacerPro**) oder Beschränkung auf einfachere DLs (**QuOnto** für „DL-Lite“)

⇝ Einschränkung des Problems für bessere Implementierbarkeit

Was sind Regeln?

1. Logische Regeln (Fragmente von Prädikatenlogik):
   - „$F \rightarrow G \equiv \neg F \vee G$“
   - Logische Erweiterung der Wissensbasis ⇝ **statisch**
   - Open World
   - **Deklarativ** (beschreibend)
2. Prozedurale Regeln (z.B. Production Rules):
   - „*If X then Y else Z*“
   - Ausführbare Maschinen-Anweisungen ⇝ **dynamisch**
   - **Operational** (Bedeutung = Effekt bei Ausführung)
3. Logikprogrammierung et al. (z.B. PROLOG, F-Logik):
   - „`mann(X) <- person(X) AND NOT frau(X)`“
   - Approximation logischer Semantik mit operationalen Aspekten, Built-ins möglich
   - häufig Closed World
   - **Semi-deklarativ**

Welche Regelsprache?

Regelsprachen sind untereinander kaum kompatibel!
⇝ Wahl der geeigneten Regelsprache sehr wichtig

Mögliche Kriterien:

- Klare Spezifikation von Syntax und Semantik?
- Unterstützung durch Software-Tools?
- Welche Ausdrucksmittel werden benötigt?
- Komplexität der Implementierung? Performanz?
- Kompatibilität mit bestehenden Formaten wie OWL?
- Deklarativ (Beschreiben) oder operational (Programmieren)?
- . . .

Prädikatenlogik als Regelsprache

- Regeln als Implikationsformeln der Prädikatenlogik:

$$\underbrace{H}_{\text{Kopf}} \leftarrow \underbrace{A_1 \wedge A_2 \wedge \ldots \wedge A_n}_{\text{Rumpf}}$$

  $\rightsquigarrow$ Semantisch äquivalent zu Disjunktion:

$$H \vee \neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n$$

- Implikation von rechts nach links üblich ($\leftarrow$)
- Konstanten, Variablen und Funktionssymbole erlaubt
- Quantoren für Variablen werden oft weggelassen:
  freie Variablen als universell quatifiziert verstanden
  (d.h. Regel gilt für alle Belegungen)
- Disjunktion mit mehreren nicht-negierten Atomen
  $\rightsquigarrow$ disjunktive Regel:

$$\underbrace{H_1 \vee H_2 \vee \ldots \vee H_m}_{\text{Kopf}} \leftarrow \underbrace{A_1 \wedge A_2 \wedge \ldots \wedge A_n}_{\text{Rumpf}}$$

Arten von Regeln

Bezeichnungen für „Regeln“ der Prädikatenlogik:

- **Klausel:** Disjunktion von atomaren Aussagen oder negierten atomaren Aussagen
- **Hornklausel:** Klausel mit *höchstens* einem nicht-negiertem Atom
- **Definite Klausel:** Klausel mit *genau einem* nicht negiertem Atom
- **Fakt:** Klausel aus einem einzigen nicht-negiertem Atom

Besipiele:

$\text{Frau}(x) \vee \text{Mann}(x)$	$\leftarrow$	$\text{Person}(x)$	(Klausel)
$\text{Vater}(x)$	$\leftarrow$	$\text{Mann}(x) \wedge \text{hatKind}(x, y)$	(definite Klausel)
$\text{OnkelVon}(x, y)$	$\leftarrow$	$\text{hatBruder}(\text{mutter}(x), y)$	(Funktionsymbol)
	$\leftarrow$	$\text{Mann}(x) \wedge \text{Frau}(x)$	(Hornklausel, „Integritätsbed.“)
$\text{Frau}(\text{gisela})$			(Fakt)

Semantik von Regeln

Semantik von Regeln:
Standardsemantik der Prädikatenlogik!

- Semantik weithin bekannt und gut verstanden,
- mit anderen prädikatenlogischen Ansätzen kompatibel (z.B. Beschreibungslogik!),
- Konjunktionen in Regelköpfen, Disjunktionen in Regelrümpfen: nicht nötig (*Übung*)

Datalog

Hier: nur Horn-Regeln ohne Funktionssymbole $\rightsquigarrow$ **Datalog-Regeln**

Datalog

- logische Regelsprache, ursprünglich Grundlage *deduktiver Datenbanken*
- Wissensbasen („Programme“) aus Horn-Klauseln ohne Funktionssymbole
- entscheidbar
- effizient für große *Daten*mengen, Gesamtkomplexität wie OWL Lite (ExpTime)

Datalog in der Praxis

Datalog in der Praxis:

- verschiedene Implementierungen verfügbar
- Anpassungen für das Semantic Web: XSD-Typen, URIs (z.B. → IRIS)

Erweiterungen von Datalog:

- *disjunktives Datalog* erlaubt Disjunktionen in Köpfen
- nichtmonotone Negation (keine prädikatenlogische Semantik)
- Einbindung von Informationen aus OWL-Ontologien (→ dl-programs, → dlvhex)
  ⇝ lose Kopplung von OWL und Datalog (nicht über gemeinsame prädikatenlogische Semantik)

Komplexität von SWRL

1. Logisches Schließen in OWL DL ist NExpTime-vollständig.
2. Logisches Schließen in Datalog ist ExpTime-vollständig.

⇝ Wie schwer ist logisches Schließen in OWL+SWRL?

Logisches Schließen in OWL+SWRL ist unentscheidbar.

SWRL

Wie kann man Datalog und OWL DL kombinieren?

SWRL – „Semantic Web Rule Language“

- Vorschlag einer OWL-Regelerweiterung (W3C-Einreichung)
- basierend auf Datalog
- Prädikate nur einstellig (Klassen) oder zweistellig (Rollen)
- Prädikatsnamen aus OWL-Ontologie oder als neue (nicht-OWL) Symbole
- OWL-Terme: OWL-Individuen oder Datenliterale als Konstanten,
- Zusätzliche *Built-Ins* zur Verarbeitung von Datentypen
- mehrere syntaktische Darstellungen

Beispiele:
$\texttt{hatMutter}(?x, ?y) \wedge \texttt{hatBruder}(?y, ?z) \Rightarrow \texttt{hatOnkel}(?x, ?z)$
$\texttt{Person}(?x) \wedge (\leq 1\,\texttt{hatKind})(?x) \Rightarrow (\leq 1\,\texttt{hatSohn})(?x)$

DL-safe Rules

Idee

SWRL soweit einschränken, dass Entscheidbarkeit garantiert ist.

- Beliebige Datalog-Regeln erlaubt, wobei OWL-Klassen und -Rollennamen eingebaut werden dürfen
- Regeln müssen **DL-safe** sein: Jede Variable muss auch in einem Ausdruck im Rumpf auftreten, der keine OWL-Klasse oder -Rolle verwendet!
- Semantik übernommen von OWL+SWRL (Prädikatenlogik).

⇝ DL-safety schränkt die Anwendbarkeit von Regeln auf benannte Individuen ein

Erzwingen von DL-safety

Beispiel:

$\text{onkel}(x,y) \leftarrow \text{bruder}(x,z), \text{vater}(z,y)$

$\rightsquigarrow$ nicht DL-safe, wenn „bruder“ und „vater“ OWL-Rollen sind.

Erzwingen von DL-safeness durch Einschränken der Regeln auf **bekannte** Individuen:

$\text{onkel}(x,y) \leftarrow \text{bruder}(x,z), \text{vater}(z,y), O(x), O(y), O(z)$

wobei der Fakt $O(a)$ für alle OWL-Individuen a angelegt wird.

$\rightsquigarrow$ Regel nur noch auf bekannte OWL-Individuen anwendbar

Beispiel

„Wer seinen Bruder *hasst* ist böse.“

Wissensbasis

$\text{vater}(\text{cain}, \text{adam})$ $\quad$ $\text{hasst}(\text{romulus}, \text{remus})$
$\text{vater}(\text{abel}, \text{adam})$ $\quad$ $\text{hasst}(\text{cain}, \text{abel})$
$\exists \text{vater}.\exists \text{vater}^{-}\{\text{remus}\}(\text{romulus})$

$\text{Böse}(x) \leftarrow \text{vater}(x,z), \text{vater}(y,z), \text{hasst}(x,y)$

$\text{Böse}_s(x) \leftarrow \text{vater}(x,z), \text{vater}(y,z), \text{hasst}(x,y), O(x), O(y), O(z)$

$O(\text{cain})$ $\quad$ $O(\text{abel})$ $\quad$ $O(\text{remus})$ $\ldots$
für alle Individuen der OWL-Ontologie

Logische Konsequenzen:

$\text{Böse}(\text{cain}), \text{Böse}(\text{romulus}), \text{Böse}_s(\text{cain}), \ldots$

Aber: $\text{Böse}_s(\text{romulus})$ **keine** logische Konsequenz

DL-safe Rules in der Praxis

- OWL DL mit DL-safe Rules ist entscheidbar
- Implementierung in existierenden OWL-DL-Systemen möglich
- Keine größere *theoretische* Komplexität der Berechnung

Implementierungen (23.1.2008):

- KAON2: sehr effiziente Umsetzung von DL-safe Rules, Unterstützung für disjunktive Regeln ($\rightarrow$ Webseite)
- Pellet: „preliminary implementation“ für DL-safe rules ($\rightarrow$ Webseite)

$\rightsquigarrow$ Umsetzung mit klassischen Tableau-Methoden kompliziert
$\rightsquigarrow$ „Vorberechnung“ von OWL-Ergebnissen für die Verwendung in eigenständigen Datalog-Systemen ist nicht ausreichend (unvollständig aber ev. effizienter, optional in Pellet verfügbar)

Was haben konjunktive Anfragen mit Regeln zu tun?

Jede konjunktive Anfrage kann als Regel ausgedrückt werden:

$$\exists y.(\texttt{Buch}(x) \wedge \texttt{Autor}(x,y))$$

entspricht

$$\text{Ergebnis}(x) \leftarrow \text{Buch}(x) \wedge \text{Autor}(x,y)$$

$\rightsquigarrow$ Kopf enthält Bindungen für bestimmte Variablen

Zusätzliche Schwierigkeit von Regeln:

- Ergebnisse können in anderen Regeln/Ontologieaxiomen weiterverwendet werden (Rekursion!)
- Variablen nicht immer auf benannte Individuen beschränkt (außer für Ausgabe und in DL-safe Rules)

Zusammenfassung

Konjunktive Anfragen für OWL DL

- kein offizieller Standard, aber große Verbreitung
- Anfrage basierend auf logischer Beschreibung
- Diverse Erweiterungen (SPARQL-Features, $\neg$, $\vee$, Pfadausdrücke)
- Keine normierte Sysntax (manche Implementationen verwenden SPARQL-Syntax)
- Semantik durch Erweiterung der beschreibungslogischen Interpretation von OWL

Prädikatenlogische Regelerweiterungen für OWL DL

- Datalog als gut bekannter Formalismus
- Kombination mit OWL möglich: SWRL
- Einschränkung auf DL-safe Rules erzielt Entscheidbarkeit
- Keine normierte Syntax, kein offizieller Standard
- Semantik durch Erweiterung der beschreibungslogischen Interpretation von OWL

AIFB

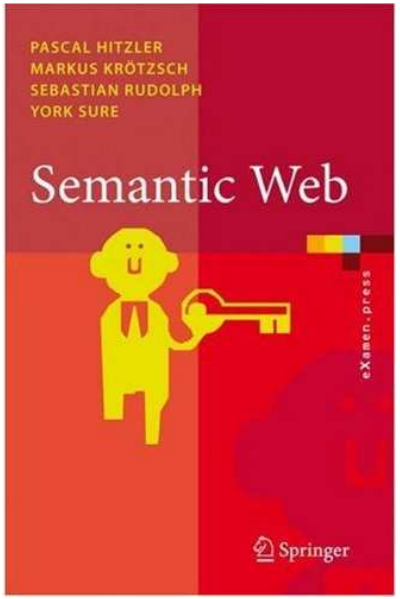
Literatur

Pascal Hitzler
Markus Krötzsch
Sebastian Rudolph
York Sure

Semantic Web
Grundlagen

Springer 2008, 277 S., Softcover
ISBN: 978-3-540-33993-9

Aktuelle Literaturhinweise online:
Kapitel 7 (Anfragen) & Vorlesung 11



AIFB