

Konjunktive Anfragen und Regelsprachen

Sebastian Rudolph

Institut AIFB · KIT

Semantic Web Technologies I (WS12/13)

Institut AIFB, KIT

<http://semantic-web-grundlagen.de>

Foliensatz von M. Krötzsch. Die nichtkommerzielle Vervielfältigung, Verbreitung und Bearbeitung dieser Folien ist zulässig (→ Lizenzbestimmungen CC-BY-NC).

Grundlagen Semantic Web

- 1 Einleitung und XML
- 2 Einführung in RDF
- 3 RDF Schema
- 4 Logik – Grundlagen
- 5 Semantik von RDF(S)
- 6 SPARQL – Syntax und Intuition
- 7 Semantik von SPARQL
- 8 Linked Data
- 9 Semantic Search
- 10 OWL – Syntax und Intuition I
- 11 OWL – Syntax und Intuition II
- 12 OWL – Semantik und Reasoning
- 13 **Konjunktive Anfragen und Regelsprachen**
- 14 ...

Literatur zu dieser Vorlesung online siehe

→ [Semantic Web – Grundlagen, Kapitel 7](#) und → [Vorlesungswebseite](#)

Vorlesung 11:

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

Ausdrucksstarke Anfragen für OWL?

SPARQL als Anfragesprache für RDF

↪ keine direkte Unterstützung für OWL

Anfrageformalismus für OWL DL: **konjunktive Anfragen**

- keine offizielle Spezifikation, keine normative Syntax
 - ↪ 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 Implementierungen verfügbar

Konjunktive Anfragen sind recht einfach!

Beispiel:

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

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

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

Konjunktive Anfragen ähneln logischen Formeln

↪ Anfragen ohne Variablen können aus einer Ontologie *folgen*

↪ Variablen als Platzhalter für Bezeichner von Individuen

Funktion μ ist **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

↪ keine partielle Funktion – alle Variablen müssen belegt sein

↪ Literale (Datentypen) hier zur Vereinfachung nicht betrachtet

Variablen bisher als Platzhalter für (benannte) Individuen,
aber

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

Beispiel:

$\text{Buch}(a)$ (a ist ein Buch)

$\text{Buch} \sqsubseteq \exists \text{Autor.T}$ (jedes Buch hat einen Autor)

\rightsquigarrow Anfrage $\text{Buch}(x) \wedge \text{Autor}(x, y)$ 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

↪ **Unbestimmte Variablen** werden durch Existenzquantoren gebunden

Beispiel:

$\text{Buch}(a)$ (a ist ein Buch)

$\text{Buch} \sqsubseteq \exists \text{Autor.} \top$ (jedes Buch hat einen Autor)

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

↪ 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	—

⇒ „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

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“)

Schlussfolgern mit OWL DL ist sehr komplex (NEXPTIME-vollständig)

↪ Wie schwierig sind dann konjunktive Anfragen?

Bisher noch nicht abschließend geklärt!

- Konjunktive Anfragen für *SHIQ* (und für OWL Lite):
2EXPTIME-vollständig!
- Konjunktive Anfragen für *SHOQ*: entscheidbar in 2EXPTIME
- Konjunktive Anfragen für *SHOIQ* (und für OWL DL):
Entscheidbarkeit nicht bekannt!
(Entscheidbar falls keine non-simple roles in der Anfrage
enthalten sind.)

↪ konjunktive Anfragen für OWL sind äußerst kompliziert

Implementationen von konjunktiven Anfragen für OWL verfügbar:

- **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 \rightsquigarrow **statisch**
 - Open World
 - **Deklarativ** (beschreibend)
- 2 Prozedurale Regeln (z.B. Production Rules):
 - „*If X then Y else Z*“
 - Ausführbare Maschinen-Anweisungen \rightsquigarrow **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)?
- ...

- Regeln als Implikationsformeln der Prädikatenlogik:

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

↔ Semantisch äquivalent zu Disjunktion:

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

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

↔ disjunktive Regel:

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

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:

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*)

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 *Datenmengen*, Gesamtkomplexität EXPTIME

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)

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 Datenlitterale als Konstanten,
- Zusätzliche *Built-Ins* zur Verarbeitung von Datentypen
- mehrere syntaktische Darstellungen

Beispiele:

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

- 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.

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

Beispiel:

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

↪ 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.

↪ Regel nur noch auf bekannte OWL-Individuen anwendbar

Beispiel

„Wer seinen Bruder *hasst* ist böse.“

Wissensbasis

vater(cain, adam)

hasst(romulus, remus)

vater(abel, adam)

hasst(cain, abel)

\exists vater. \exists vater \neg {remus}(romulus)

Böse(x) \leftarrow vater(x, z), vater(y, z), hasst(x, y)

Böse_s(x) \leftarrow vater(x, z), vater(y, z), hasst(x, y), O(x), O(y), O(z)

O(cain) O(abel) O(remus) ...

für alle Individuen der OWL-Ontologie

Logische Konsequenzen:

Böse(cain), Böse(romulus), Böse_s(cain), ...

Aber: Böse_s(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:

- Hermit: Unterstützung von DL-safe Rules für OWL, unvollständig für non-simple roles in der Anfrage (→ Webseite)
- KAON2: sehr effiziente Umsetzung von DL-safe Rules, Unterstützung für disjunktive Regeln (→ Webseite)
- Pellet: „preliminary implementation“ für DL-safe rules (→ Webseite)

↔ Umsetzung mit klassischen Tableau-Methoden kompliziert

↔ „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:

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

entspricht

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

↪ 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)

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 Syntax (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

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

