

Modul Semantik Web (SS2011)

Dr. Sören Auer
Dr. Jens Lehmann
Prof. Dr. Gerhard Brewka
Frank Loebe

Institut für Informatik · Universität Leipzig

Regeln im Semantic Web
24. Mai 2011



- 1 Einleitung und Ausblick
- 2 URIs und Einführung in RDF
- 3 RDF Schema
- 4 Logik – Grundlagen
- 5 Semantik von RDF(S)
- 6 OWL – Syntax und Intuition
- 7 OWL – Semantik und Reasoning
- 8 **Regeln im Semantic Web**
- 9 RDF-Datenbanken, Triple- und Knowledge-Stores, Anfragesprachen SPARQL, SPARUL
- 10 Integration von RDF und XHTML - RDFa, GRDDL
- 11 Linked Data Web, Semantische Wikis
- 12 Semantic Web Anwendungen, Rück- und Ausblick

- 1 Einleitung und Motivation
- 2 Regelsprachen und das Semantic Web
- 3 Datalog
- 4 Regeln für OWL: SWRL
- 5 Description Logic Rules
- 6 Rule Interchange Format (RIF)
- 7 Zusammenfassung

- 1 Einleitung und Motivation
- 2 Regelsprachen und das Semantic Web
- 3 Datalog
- 4 Regeln für OWL: SWRL
- 5 Description Logic Rules
- 6 Rule Interchange Format (RIF)
- 7 Zusammenfassung

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?“

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:

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



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.

Gliederung

- 1 Einleitung und Motivation
- 2 Regelsprachen und das Semantic Web**
- 3 Datalog
- 4 Regeln für OWL: SWRL
- 5 Description Logic Rules
- 6 Rule Interchange Format (RIF)
- 7 Zusammenfassung

Was sind Regeln?

- ① Logische Regeln (Implikationen in Prädikatenlogik):
 - „ $F \rightarrow G$ “ (\equiv „ $\neg F \vee G$ “)
 - Logische Erweiterung der Wissensbasis \rightsquigarrow **statisch**
 - Open World
 - **Deklarativ** (beschreibend)

Was sind Regeln?

- ① Logische Regeln (Implikationen in 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)

Was sind Regeln?

- ① Logische Regeln (Implikationen in 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 (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**“

Was sind Regeln?

- ① Logische Regeln (Implikationen in 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 (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**“
- ④ Ableitungsregeln eines Kalküls (z.B. Regeln zur RDF-Semantik)
 - Regeln nicht als Teil der Wissensbasis, „Meta-Regeln“ \rightsquigarrow nicht Thema dieser Vorlesung

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)?
- ...

Welche Regelsprache?

- ① Logische Regeln (Implikationen in Prädikatenlogik):
 - klar definiert, umfassend erforscht, gut verstanden
 - sehr gut kompatibel mit OWL DL und RDF
 - ohne Einschränkungen nicht entscheidbar
- ② Prozedurale Regeln (z.B. Production Rules):
 - viele unabhängige Ansätze, oft nur vage definiert
 - Verwendung oft wie Programmiersprachen, Beziehung zu OWL und RDF unklar
 - effiziente Abarbeitung möglich
- ③ Logikprogrammierung (z.B. Prolog, F-Logik):
 - klar definiert, aber viele unterschiedliche Ansätze
 - teilweise kompatibel mit OWL und RDF
 - Entscheidbarkeit/Komplexität stark vom gewählten Ansatz abhängig

Welche Regelsprache?

- ① Logische Regeln (Implikationen in Prädikatenlogik):
 - klar definiert, umfassend erforscht, gut verstanden
 - sehr gut kompatibel mit OWL DL und RDF
 - ohne Einschränkungen nicht entscheidbar
- ② Prozedurale Regeln (z.B. Production Rules):
 - viele unabhängige Ansätze, oft nur vage definiert
 - Verwendung oft wie Programmiersprachen, Beziehung zu OWL und RDF unklar
 - effiziente Abarbeitung möglich
- ③ Logikprogrammierung (z.B. Prolog, F-Logik):
 - klar definiert, aber viele unterschiedliche Ansätze
 - teilweise kompatibel mit OWL und RDF
 - Entscheidbarkeit/Komplexität stark vom gewählten Ansatz abhängig

⇒ Schwerpunkt dieser Vorlesung: prädikatenlogische Regeln
(die aber auch die Grundlage der Logikprogrammierung sind)



Gliederung

- 1 Einleitung und Motivation
- 2 Regelsprachen und das Semantic Web
- 3 Datalog**
- 4 Regeln für OWL: SWRL
- 5 Description Logic Rules
- 6 Rule Interchange Format (RIF)
- 7 Zusammenfassung

Prädikatenlogik als Regelsprache

- Regeln als Implikationsformeln der Prädikatenlogik:

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

Prädikatenlogik als Regelsprache

- Regeln als Implikationsformeln der Prädikatenlogik:

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

↔ Semantisch äquivalent zu Disjunktion:

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

- 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:

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

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

Beispiele:

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

Einschränkung auf Horn-Regeln *ohne* Funktionssymbole \rightsquigarrow

Datalog-Regeln

Datalog

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

Semantik von Regeln:

Semantik von Regeln:

Standardsemantik der Prädikatenlogik!

- Semantik weithin bekannt und gut verstanden
- mit anderen prädikatenlogischen Ansätzen kompatibel (z.B. Beschreibungslogik)

Semantik von Datalog (Auffrischung)

Semantik definiert über über **logische Modelle**:

- Interpretation \mathcal{I} mit Domäne $\Delta_{\mathcal{I}}$
- Auswertung von Variablen: Variablenzuweisung \mathcal{Z} (Abbildung von Variablen auf $\Delta_{\mathcal{I}}$)
- Interpretation von Formeln und Termen unter \mathcal{I} (und \mathcal{Z}):
 - Interpretation einer Konstante: $a^{\mathcal{I},\mathcal{Z}} = a^{\mathcal{I}} \in \Delta_{\mathcal{I}}$
 - Interpretation einer Variable: $x^{\mathcal{I},\mathcal{Z}} = \mathcal{Z}(x) \in \Delta_{\mathcal{I}}$
 - Interpretation eines n-stelligen Prädikats: $p^{\mathcal{I}} \in \Delta_{\mathcal{I}}^n$
 - $\mathcal{I}, \mathcal{Z} \models p(t_1, \dots, t_n)$ genau dann wenn $(t_1^{\mathcal{I},\mathcal{Z}}, \dots, t_n^{\mathcal{I},\mathcal{Z}}) \in p^{\mathcal{I}}$,
 - $\mathcal{I} \models B \rightarrow H$ genau dann wenn für jede Variablenzuweisung \mathcal{Z} gilt: entweder $\mathcal{I}, \mathcal{Z} \models H$ oder $\mathcal{I}, \mathcal{Z} \not\models B$.
- \mathcal{I} ist ein Modell für eine Regelmengung, wenn gilt: $\mathcal{I} \models B \rightarrow H$ für alle Regeln $B \rightarrow H$ dieser Menge

Logische Folgerung wie in „Obstlogik“ (vgl. Vorlesung 5)

Datalog in der Praxis:

- verschiedene Implementierungen verfügbar
- Anpassungen für das Semantic Web: Datentypen aus XML Schema, URIs (z.B. → IRIS)

Erweiterungen von Datalog:

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

Gliederung

- 1 Einleitung und Motivation
- 2 Regelsprachen und das Semantic Web
- 3 Datalog
- 4 Regeln für OWL: SWRL**
- 5 Description Logic Rules
- 6 Rule Interchange Format (RIF)
- 7 Zusammenfassung

Wie kann man Datalog und OWL DL kombinieren?

Wie kann man Datalog und OWL DL kombinieren?

SWRL – „Semantic Web Rule Language“

- Vorschlag einer OWL-Regelerweiterung (W3C-Einreichung)
- Idee: Datalog-Regeln mit Bezug zu OWL-Ontologie
- Symbole in Regeln können OWL-Bezeichner sein, oder neue Datalog-Bezeichner
- Zusätzliche *Built-Ins* zur Verarbeitung von Datentypen
- mehrere syntaktische Darstellungen

OWL DL (Beschreibungslogik) und Datalog verwenden die gleichen Interpretationen:

- OWL-Individuen sind Datalog-Konstanten
- OWL-Klassen sind einstellige Datalog-Prädikate
- OWL-Rollen sind zweistellige Datalog-Prädikate

OWL DL (Beschreibungslogik) und Datalog verwenden die gleichen Interpretationen:

- OWL-Individuen sind Datalog-Konstanten
- OWL-Klassen sind einstellige Datalog-Prädikate
- OWL-Rollen sind zweistellige Datalog-Prädikate

↪ \mathcal{I} kann gleichzeitig Modell sein für OWL-Ontologie und Menge von Datalog-Regeln

↪ Schlussfolgerung über OWL-Datalog-Kombination möglich

Beispiel

Kombinierte SWRL-Wissensbasis (Datalog + Beschreibungslogik):

- (1) $\text{Vegetarier}(x) \wedge \text{Fischprodukt}(y) \rightarrow \text{magNicht}(x, y)$
- (2) $\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \rightarrow \text{Unglücklich}(x)$
- (3) $\text{hatBestellt}(x, y) \rightarrow \text{Gericht}(y)$
- (4) $\text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enthält}(y, z) \rightarrow \text{magNicht}(x, y)$
- (5) $\rightarrow \text{Vegetarier}(\text{markus})$
- (6) $\text{Glücklich}(x) \wedge \text{Unglücklich}(x) \rightarrow$
- (7) $\exists \text{hatBestellt. ThaiCurry}(\text{markus})$
- (8) $\text{ThaiCurry} \sqsubseteq \exists \text{enthält. Fischprodukt}$

Beispiel

Kombinierte SWRL-Wissensbasis (Datalog + Beschreibungslogik):

- (1) $\text{Vegetarier}(x) \wedge \text{Fischprodukt}(y) \rightarrow \text{magNicht}(x, y)$
- (2) $\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \rightarrow \text{Unglücklich}(x)$
- (3) $\text{hatBestellt}(x, y) \rightarrow \text{Gericht}(y)$
- (4) $\text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enthält}(y, z) \rightarrow \text{magNicht}(x, y)$
- (5) $\rightarrow \text{Vegetarier}(\text{markus})$
- (6) $\text{Glücklich}(x) \wedge \text{Unglücklich}(x) \rightarrow$
- (7) $\exists \text{hatBestellt. ThaiCurry}(\text{markus})$
- (8) $\text{ThaiCurry} \sqsubseteq \exists \text{enthält. Fischprodukt}$

Wir können folgern: $\text{Unglücklich}(\text{markus})$

Wie schwer ist SWRL?

- ① Logisches Schließen in OWL DL ist NEXPTIME-vollständig.
- ② Logisches Schließen in OWL 2 DL ist N2EXPTIME-vollständig.
- ③ Logisches Schließen in Datalog ist EXPTIME-vollständig.

~> Wie schwer ist logisches Schließen in SWRL?

Wie schwer ist SWRL?

- ① Logisches Schließen in OWL DL ist NEXPTIME-vollständig.
- ② Logisches Schließen in OWL 2 DL ist N2EXPTIME-vollständig.
- ③ Logisches Schließen in Datalog ist EXPTIME-vollständig.

↪ Wie schwer ist logisches Schließen in SWRL?

Logisches Schließen in SWRL ist unentscheidbar
(für OWL und damit auch für OWL 2).

Unentscheidbarkeit von SWRL

SWRL ist unentscheidbar

Es gibt keinen Algorithmus, mit dem man *alle* logischen Schlüsse aus *allen* SWRL-Wissensbasen ziehen kann, selbst wenn man beliebig (endlich) viel Rechenzeit und Speicher zur Verfügung hat.

Praktisch möglich dagegen sind:

- ① Algorithmen, die alle Schlüsse aus **einem Teil der SWRL-Wissensbasen** ziehen
- ② Algorithmen, die aus allen SWRL-Wissensbasen **einen Teil der Schlüsse** ziehen

↔ Beides ist trivial möglich, wenn der entsprechende „Teil“ nur sehr klein ist.



Entscheidbare Fragmente von SWRL

Für welche Arten von SWRL-Wissensbasen kann man vollständige Inferenz-Algorithmen finden?

- Für die Menge aller SWRL-Wissensbasen, die nur aus Ontologien in OWL (2) bestehen.
- Für die Menge aller SWRL-Wissensbasen, die nur aus Datalog-Regeln bestehen.
- Für jede feste endliche Menge an SWRL-Wissensbasen.

⇒ Gibt es noch interessantere entscheidbare Fragmente?

Entscheidbare Fragmente von SWRL

Für welche Arten von SWRL-Wissensbasen kann man vollständige Inferenz-Algorithmen finden?

- Für die Menge aller SWRL-Wissensbasen, die nur aus Ontologien in OWL (2) bestehen.
- Für die Menge aller SWRL-Wissensbasen, die nur aus Datalog-Regeln bestehen.
- Für jede feste endliche Menge an SWRL-Wissensbasen.

⇒ Gibt es noch interessantere entscheidbare Fragmente?

- Description Logic Rules
- DL-safe Rules

Gliederung

- 1 Einleitung und Motivation
- 2 Regelsprachen und das Semantic Web
- 3 Datalog
- 4 Regeln für OWL: SWRL
- 5 Description Logic Rules**
- 6 Rule Interchange Format (RIF)
- 7 Zusammenfassung

Beobachtung

Manche SWRL-Regeln lassen sich bereits in OWL 2 (also der Beschreibungslogik *SROIQ*) ausdrücken.

- Identifizierung dieser **Description Logic Rules** liefert ein entscheidbares Fragment von SWRL
- Ziel: „Versteckte“ Ausdrucksstärke von OWL 2 nutzen
- Implementierung direkt durch OWL-2-Tools

Rückblick: *SROIQ* (rot = zusätzlich zu *SHOIN*)

Klassenausdrücke

Klassennamen	A, B
Konjunktion	$C \sqcap D$
Disjunktion	$C \sqcup D$
Negation	$\neg C$
Exist. Rollenrestr.	$\exists R.C$
Univ. Rollenrestr.	$\forall R.C$
Self	$\exists S.\text{Self}$
Größer-als	$\geq n S.C$
Kleiner-als	$\leq n S.C$
Nominale	$\{a\}$

Rollen

Rollennamen	R, S, T
einfache Rollen	S, T
Inverse Rollen	R^-
Universelle Rolle	U

Tbox (Klassenaxiome)

Inklusion	$C \sqsubseteq D$
Äquivalenz	$C \equiv D$

Rbox (Rollenaxiome)

Inklusion	$R_1 \sqsubseteq R_2$
Allgem. Inkl.	$R_1^{(-)} \circ \dots \circ R_n^{(-)} \sqsubseteq R$
Transitivität	$\text{Tra}(R)$
Symmetrie	$\text{Sym}(R)$
Reflexivität	$\text{Ref}(R)$
Irreflexivität	$\text{Irr}(S)$
Disjunktheit	$\text{Dis}(S, T)$

Abox (Fakten)

Klassenzugehörigkeit	$C(a)$
Rollenbeziehung	$R(a, b)$
Neg. Rollenbeziehung	$\neg S(a, b)$
Gleichheit	$a \approx b$
Ungleichheit	$a \not\approx b$



Alle SROIQ-Axiome können als SWRL-Regeln geschrieben werden:

- $C \sqsubseteq D$ entspricht $C(x) \rightarrow D(x)$
- $R \sqsubseteq S$ entspricht $R(x, y) \rightarrow S(x, y)$

Einfache Regeln mit *SROIQ*

Alle SROIQ-Axiome können als SWRL-Regeln geschrieben werden:

- $C \sqsubseteq D$ entspricht $C(x) \rightarrow D(x)$
- $R \sqsubseteq S$ entspricht $R(x, y) \rightarrow S(x, y)$

Einige Klassen können innerhalb von Regeln „zerlegt“ werden:

- $\text{Glücklich} \sqcap \text{Unglücklich} \sqsubseteq \perp$ entspricht
 $\text{Glücklich}(x) \wedge \text{Unglücklich}(x) \rightarrow$
- $\exists \text{wohnort} . \exists \text{liegtIn} . \text{EULand} \sqsubseteq \text{EUBürger}$ entspricht
 $\text{wohnort}(x, y) \wedge \text{liegtIn}(y, z) \wedge \text{EULand}(z) \rightarrow \text{EUBürger}(x)$

Einfache Regeln mit *SROIQ*

Alle SROIQ-Axiome können als SWRL-Regeln geschrieben werden:

- $C \sqsubseteq D$ entspricht $C(x) \rightarrow D(x)$
- $R \sqsubseteq S$ entspricht $R(x, y) \rightarrow S(x, y)$

Einige Klassen können innerhalb von Regeln „zerlegt“ werden:

- $\text{Glücklich} \sqcap \text{Unglücklich} \sqsubseteq \perp$ entspricht
 $\text{Glücklich}(x) \wedge \text{Unglücklich}(x) \rightarrow$
- $\exists \text{wohnort}.\exists \text{liegtIn.EULand} \sqsubseteq \text{EUBürger}$ entspricht
 $\text{wohnort}(x, y) \wedge \text{liegtIn}(y, z) \wedge \text{EULand}(z) \rightarrow \text{EUBürger}(x)$

SROIQ-Rollenaxiome liefern weitere Regeln:

- $\text{hatMutter} \circ \text{hatBruder} \sqsubseteq \text{hatOheim}$ entspricht
 $\text{hatMutter}(x, y) \wedge \text{hatBruder}(y, z) \rightarrow \text{hatOheim}(x, z)$

Was ist mit

$\text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enth\"alt}(y, z) \rightarrow \text{magNicht}(x, y)?$

Was ist mit

$\text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enth\"alt}(y, z) \rightarrow \text{magNicht}(x, y)?$

- Regelkopf mit zwei Variablen \rightsquigarrow nicht durch Subklassen-Axiom darstellbar
- Regelrumpf enthält Klassenausdrücke \rightsquigarrow nicht durch Subproperty-Axiom darstellbar

Noch mehr Regeln (I)

Was ist mit

$\text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enth\"alt}(y, z) \rightarrow \text{magNicht}(x, y)?$

- Regelkopf mit zwei Variablen \rightsquigarrow nicht durch Subklassen-Axiom darstellbar
- Regelrumpf enthält Klassenausdrücke \rightsquigarrow nicht durch Subproperty-Axiom darstellbar

Trotzdem ist diese Regel in OWL 2 darstellbar!

Noch mehr Regeln (II)

Einfacheres Beispiel: $\text{Mann}(x) \wedge \text{hatKind}(x, y) \rightarrow \text{vaterVon}(x, y)$

Idee

Ersetze $\text{Mann}(x)$ durch ein Rollen-Atom, so dass die Regel als allgemeine Rolleninklusion mit \circ darstellbar wird.



Noch mehr Regeln (II)

Einfacheres Beispiel: $\text{Mann}(x) \wedge \text{hatKind}(x, y) \rightarrow \text{vaterVon}(x, y)$

Idee

Ersetze $\text{Mann}(x)$ durch ein Rollen-Atom, so dass die Regel als allgemeine Rolleninklusion mit \circ darstellbar wird.

Trick: mit $\exists R.\text{Self}$ kann man Klassen in Rollen umwandeln:

- Hilfsrolle R_{Mann}
- Hilfsaxiom $\text{Mann} \equiv \exists R_{\text{Mann}}.\text{Self}$
- Intuition: „Männer sind genau die Dinge, die ein R_{Mann} -Beziehung zu sich selbst haben.“

Mit diesem Hilfsaxiom kann die Regel geschrieben werden als:

$R_{\text{Mann}} \circ \text{hatKind} \sqsubseteq \text{vaterVon}$



Beispiel:

$\text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enth\"alt}(y, z) \rightarrow \text{magNicht}(x, y)$

Noch mehr Regeln (III)

Beispiel:

$\text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enthält}(y, z) \rightarrow \text{magNicht}(x, y)$

wird zu

$\text{Gericht} \equiv \exists R_{\text{Gericht}}.\text{Self}$

$\text{magNicht} \circ \text{enthält}^{-1} \circ R_{\text{Gericht}} \sqsubseteq \text{magNicht}$

Noch mehr Regeln (IV)

Nicht so einfach:

$\text{Vegetarier}(x) \wedge \text{Fischprodukt}(y) \rightarrow \text{magNicht}(x, y)$

Noch mehr Regeln (IV)

Nicht so einfach:

$\text{Vegetarier}(x) \wedge \text{Fischprodukt}(y) \rightarrow \text{magNicht}(x, y)$

Idee

Verbinde unzusammenhängende Teile im Regelrumpf durch die universelle Rolle U .

- Hilfsrollen $R_{\text{Vegetarier}}$ und $R_{\text{Fischprodukt}}$
- Hilfsaxiome $\text{Vegetarier} \equiv \exists R_{\text{Vegetarier}}.\text{Self}$ und $\text{Fischprodukt} \equiv \exists R_{\text{Fischprodukt}}.\text{Self}$

Mit diesen Hilfsaxiomen kann die Regel geschrieben werden als:

$R_{\text{Vegetarier}} \circ U \circ R_{\text{Fischprodukt}} \sqsubseteq \text{magNicht}$

Nicht alle SWRL-Regeln können so dargestellt werden!

Nicht alle SWRL-Regeln können so dargestellt werden!

Beispiel:

$\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \rightarrow \text{Unglücklich}(x)$
ist nicht in *SR_{OIQ}* darstellbar.

Mögliche Umwandlungen im Regelrumpf im Überblick

- Rollen umkehren, z.B. $\text{enthält}(y, z) \mapsto \text{enthält}^-(z, y)$
- Seitenarme „aufrollen“, z.B.
 $\text{liegtIn}(y, z) \wedge \text{EULand}(z) \mapsto \exists \text{liegtIn.EULand}(y)$
- Konzepte durch Rollen ersetzen, z.B. $\text{Mann}(x) \mapsto R_{\text{Mann}}(x, x)$
- Ketten in Rolleninklusionen umwandeln (\wedge durch \circ ersetzen)

Vorbereitung: **Regel normalisieren**

- Für jedes *Vorkommen* (!) einer Konstante a der Regel:
Füge im Rumpf $\{a\}(x)$ mit einer neuen Variable x ein und ersetze das Vorkommen von a durch x .
- Ersetze jedes Atom $R(x, x)$ durch $\exists R.\text{Self}(x)$.

Vorbereitung: **Regel normalisieren**

- Für jedes *Vorkommen* (!) einer Konstante a der Regel:
Füge im Rumpf $\{a\}(x)$ mit einer neuen Variable x ein und ersetze das Vorkommen von a durch x .
- Ersetze jedes Atom $R(x, x)$ durch $\exists R.\text{Self}(x)$.

Abhängigkeitsgraph einer Regel: *Ungerichteter* Graph mit

- Knoten = Variablen der Regel
- Kanten = Rollenatome der Regel (ohne Richtung!)

Description Logic Rules: Definition

Vorbereitung: **Regel normalisieren**

- Für jedes *Vorkommen* (!) einer Konstante a der Regel:
Füge im Rumpf $\{a\}(x)$ mit einer neuen Variable x ein und ersetze das Vorkommen von a durch x .
- Ersetze jedes Atom $R(x, x)$ durch $\exists R.\text{Self}(x)$.

Abhängigkeitsgraph einer Regel: *Ungerichteter* Graph mit

- Knoten = Variablen der Regel
- Kanten = Rollenatome der Regel (ohne Richtung!)

Eine SWRL-Regel ist eine **Description Logic Rule** wenn gilt:

- ① alle Regelatome verwenden *SR**OIQ*-Konzepte und -Rollen,
- ② der Abhängigkeitsgraph der normalisierten Regel hat keine Zyklen



Beispiel

DL Rules in der früheren SWRL-Wissensbasis:

(1) $\text{Vegetarier}(x) \wedge \text{Fischprodukt}(y) \rightarrow \text{magNicht}(x, y)$

(3) $\text{hatBestellt}(x, y) \rightarrow \text{Gericht}(y)$

(4) $\text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enthält}(y, z) \rightarrow \text{magNicht}(x, y)$

(5) $\rightarrow \text{Vegetarier}(\text{markus})$

(6) $\text{Glücklich}(x) \wedge \text{Unglücklich}(x) \rightarrow$

Regel (2) $\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \rightarrow \text{Unglücklich}(x)$

ist keine DL Rule

Anmerkung: Description Logic Rules müssen nach Umwandlung in *SROIQ* natürlich auch die Bedingungen an einfache Rollen und reguläre RBoxen erfüllen!

Umwandlung von DL Rules nach *SROIQ* (I)

Eingabe: Eine Description Logic Rule

Umwandlung von DL Rules nach *SROIQ* (I)

Eingabe: Eine Description Logic Rule

- 1 Normalisiere die Regel.

Umwandlung von DL Rules nach *SROIQ* (I)

Eingabe: Eine Description Logic Rule

- 1 Normalisiere die Regel.
- 2 Für jedes Paar von Variablen x und y :
Sind x und y im Abhängigkeitsgraph nicht verbunden, d.h. es gibt keinen Pfad zwischen x und y , dann füge im Rumpf $U(x, y)$ ein.

Umwandlung von DL Rules nach *SROIQ* (I)

Eingabe: Eine Description Logic Rule

- 1 Normalisiere die Regel.
- 2 Für jedes Paar von Variablen x und y :
Sind x und y im Abhängigkeitsgraph nicht verbunden, d.h. es gibt keinen Pfad zwischen x und y , dann füge im Rumpf $U(x, y)$ ein.
- 3 Der Regelkopf hat nun die Form $D(z)$ oder $S(z, z')$.
Für jedes Atom $R(x, y)$ im Rumpf:
Falls im Abhängigkeitsgraph der Pfad von z nach y kürzer ist als der von z nach x , so ersetze $R(x, y)$ mit $R^-(y, x)$.

Umwandlung von DL Rules nach *SROIQ* (I)

Eingabe: Eine Description Logic Rule

- 1 Normalisiere die Regel.
- 2 Für jedes Paar von Variablen x und y :
Sind x und y im Abhängigkeitsgraph nicht verbunden, d.h. es gibt keinen Pfad zwischen x und y , dann füge im Rumpf $U(x, y)$ ein.
- 3 Der Regelkopf hat nun die Form $D(z)$ oder $S(z, z')$.
Für jedes Atom $R(x, y)$ im Rumpf:
Falls im Abhängigkeitsgraph der Pfad von z nach y kürzer ist als der von z nach x , so ersetze $R(x, y)$ mit $R^-(y, x)$.
- 4 Falls im Rumpf ein Atom $R(x, y)$ vorkommt, so dass y in keinem anderen zweistelligen Atom der Regel auftritt:
 - Wenn der Rumpf n einstellige Atome $C_1(y), \dots, C_n(y)$ enthält, dann definiere $E := C_1 \sqcap \dots \sqcap C_n$ und entferne $C_1(y), \dots, C_n(y)$ aus dem Rumpf. Andernfalls definiere $E := \top$.
 - Ersetze $R(x, y)$ durch $\exists R.E(x)$.

Wiederhole Schritt 4 solange es solche $R(x, y)$ gibt.

Umwandlung von DL Rules nach \mathcal{SROIQ} (II)

Die Regel kann jetzt in \mathcal{SROIQ} ausgedrückt werden:

- Falls der Regelkopf einstellig ist, dann hat die Regel die Form $C_1(x) \wedge \dots \wedge C_n(x) \rightarrow D(x)$.
Ersetze sie durch $C_1 \sqcap \dots \sqcap C_n \sqsubseteq D$.
- Falls der Regelkopf zweistellig ist, dann
 - Für jedes einstellige Atom $C(z)$ im Rumpf:
Erzeuge ein neues Axiom $C \equiv \exists R_C.\text{Self}$ (die Rolle R_C ist neu) und ersetze $C(z)$ durch $R_C(z, z)$.
 - Die Regel hat nun die Form $R_1(x, x_2) \wedge \dots \wedge R_n(x_n, y) \rightarrow S(x, y)$.
Ersetze sie durch $R_1 \circ \dots \circ R_n \sqsubseteq S$.

Diese Umformung von Regeln einer SWRL-Wissensbasis verändert ihre Erfüllbarkeit nicht.

(Natürlich dürfen Hilfssymbole wie R_C noch nirgends vorkommen.)

Konvertieren Sie folgende Regel in *SROIQ*-Axiome:

$$\text{arbeitetIn}(w, x) \wedge \text{anstellung}(w, \mathbf{FEST}) \wedge \text{Uni}(x) \\ \wedge \text{Doktorand}(y) \wedge \text{betreutVon}(y, w) \rightarrow \text{professorVon}(w, y)$$

Konvertieren Sie folgende Regel in *SROIQ*-Axiome:

$$\text{arbeitetIn}(w, x) \wedge \text{anstellung}(w, \text{FEST}) \wedge \text{Uni}(x) \\ \wedge \text{Doktorand}(y) \wedge \text{betreutVon}(y, w) \rightarrow \text{professorVon}(w, y)$$

Nächster Schritt:

Normalisiere die Regel.

Konvertieren Sie folgende Regel in *SROIQ*-Axiome:

$$\text{arbeitetIn}(\mathbf{w}, \mathbf{x}) \wedge \text{anstellung}(\mathbf{w}, \mathbf{z}) \wedge \{\mathbf{FEST}\}(\mathbf{z}) \wedge \text{Uni}(\mathbf{x}) \\ \wedge \text{Doktorand}(\mathbf{y}) \wedge \text{betreutVon}(\mathbf{y}, \mathbf{w}) \rightarrow \text{professorVon}(\mathbf{w}, \mathbf{y})$$

Nächster Schritt:

Für jedes Paar von Variablen x und y : Sind x und y im Abhängigkeitsgraph nicht verbunden, d.h. es gibt keinen Pfad zwischen x und y , dann füge im Rumpf $U(x, y)$ ein.

Konvertieren Sie folgende Regel in *SROIQ*-Axiome:

$$\text{arbeitetIn}(\mathbf{w}, \mathbf{x}) \wedge \text{anstellung}(\mathbf{w}, \mathbf{z}) \wedge \{\mathbf{FEST}\}(\mathbf{z}) \wedge \text{Uni}(\mathbf{x}) \\ \wedge \text{Doktorand}(\mathbf{y}) \wedge \text{betreutVon}(\mathbf{y}, \mathbf{w}) \rightarrow \text{professorVon}(\mathbf{w}, \mathbf{y})$$

Nächster Schritt:

Der Regelkopf hat nun die Form $D(z)$ oder $S(z, z')$. Für jedes Atom $R(x, y)$ im Rumpf: Falls im Abhängigkeitsgraph der Pfad von z nach y kürzer ist als der von z nach x , so ersetze $R(x, y)$ mit $R^-(y, x)$.

Übung

Konvertieren Sie folgende Regel in *SROIQ*-Axiome:

$$\text{arbeitetIn}(\mathbf{w}, \mathbf{x}) \wedge \text{anstellung}(\mathbf{w}, \mathbf{z}) \wedge \{\mathbf{FEST}\}(\mathbf{z}) \wedge \text{Uni}(\mathbf{x}) \\ \wedge \text{Doktorand}(\mathbf{y}) \wedge \text{betreutVon}^-(\mathbf{w}, \mathbf{y}) \rightarrow \text{professorVon}(\mathbf{w}, \mathbf{y})$$

Nächster Schritt:

Falls im Rumpf ein Atom $R(x, y)$ vorkommt, so dass y in keinem anderen zweistelligen Atom der Regel auftritt:

- Wenn der Rumpf n einstellige Atome $C_1(y), \dots, C_n(y)$ enthält, dann definiere $E := C_1 \sqcap \dots \sqcap C_n$ und entferne $C_1(y), \dots, C_n(y)$ aus dem Rumpf. Andernfalls definiere $E := \top$.
- Ersetze $R(x, y)$ durch $\exists R.E(x)$.

Wiederhole Schritt 4 solange es solche $R(x, y)$ gibt.

Konvertieren Sie folgende Regel in *SRIOIQ*-Axiome:

$$\exists \text{arbeitetIn.Uni}(\mathbf{w}) \wedge \exists \text{anstellung.}\{FEST\}(\mathbf{w}) \\ \wedge \text{Doktorand}(\mathbf{y}) \wedge \text{betreutVon}^-(\mathbf{w}, \mathbf{y}) \rightarrow \text{professorVon}(\mathbf{w}, \mathbf{y})$$

Nächster Schritt:

Für jedes einstellige Atom $C(z)$ im Rumpf:

Erzeuge ein neues Axiom $C \equiv \exists R_C.\text{Self}$ (die Rolle R_C ist neu) und ersetze $C(z)$ durch $R_C(z, z)$.

Übung

Konvertieren Sie folgende Regel in *SR0IQ*-Axiome:

$$\exists R_1.\text{Self} \equiv \exists \text{arbeitetIn.Uni}$$

$$\exists R_2.\text{Self} \equiv \exists \text{anstellung.}\{FEST\}$$

$$\exists R_3.\text{Self} \equiv \text{Doktorand}$$

$$\begin{aligned} & R_1(w, w) \wedge R_2(w, w) \wedge R_3(y, y) \\ \wedge \text{betreutVon}^-(w, y) & \rightarrow \text{professorVon}(w, y) \end{aligned}$$

Nächster Schritt:

Die Regel hat nun die Form $R_1(x, x_2) \wedge \dots \wedge R_n(x_n, y) \rightarrow S(x, y)$.
Ersetze sie durch $R_1 \circ \dots \circ R_n \sqsubseteq S$.

Lösung:

$\exists R_1.\text{Self} \equiv \exists \text{arbeitetIn.Uni}$

$\exists R_2.\text{Self} \equiv \exists \text{anstellung.}\{FEST\}$

$\exists R_3.\text{Self} \equiv \text{Doktorand}$

$R_1 \circ R_2 \circ \text{betreutVon}^- \circ R_3 \sqsubseteq \text{professorVon}$

Gliederung

- 1 Einleitung und Motivation
- 2 Regelsprachen und das Semantic Web
- 3 Datalog
- 4 Regeln für OWL: SWRL
- 5 Description Logic Rules
- 6 Rule Interchange Format (RIF)**
- 7 Zusammenfassung

Rule Interchange Format (RIF)

- engl.: Rule Interchange Format (RIF)
- **am 22. Juni 2010 als W3C-Standard verabschiedet**
- entwickelt um Regeln auszutauschen
- **Fokus liegt auf dem Austausch** nicht ein Format für alle Regelsprachen
- einzelne Sprache kann nicht verschiedene Paradigmen und Bedürfnisse für Regelbenutzungen erfüllen
- **Familie von Sprachen**, die **Dialekte** (dialects) genannt werden
- RIF ist einheitlich (uniform) und erweiterbar (extensible)

- Konzentration auf zwei Arten von Dialekten:
 - ① logikbasiert (z.B. Prädikatenlogik, Logikprogrammierung)
 - ② “rules with actions” (z.B. Produktionsregeln)
- RIF bietet Framework zur Definition eigener Dialekte
- RIF mit RDF und OWL kompatibel:
 - kann semantisch kombiniert werden mit OWL/RDF
 - RDF-Syntax für RIF vorhanden

Vorstellung der RIF Dokumente 1/2

Dokument	Beschreibung
RIF-BLD: The Basic Logic Dialect	definite Hornklauseln, Standard-Prädikatenlogik-Semantik
RIF-PRD: The Production Rule Dialect	deckt Vielzahl von Produktionsregelsystemen ab
RIF-Core: The Core Dialect	ermöglicht Regelaustausch zwischen Systemen mit Logikregeln und Produktionsregeln verwenden
RIF-FLD: The Framework for Logic Dialect	logisches Erweiterungsframework um Anstrengungen zu minimieren neue Logikdialekte zu definieren
RIF-RDF+OWL: RDF und OWL Kompatibilität	Kombination von RIF mit RDF oder OWL

Vorstellung der RIF Dokumente 2/2

Dokument	Beschreibung
RIF-DTB: Datatypes und Build-ins	enthält Datentypen, Funktionen und Prädikate für RIF-Dialekte
RIF+XML-Data:	spezifiziert, wie RIF mit XML Datenquellen kombiniert werden kann (Import, Semantik)
RIF-OWLRL: OWL 2 RL in RIF	Axiomatisierung von OWL 2 RL in RIF
RIF in RDF	umkehrbares Mapping von RIF auf RDF
RIF-UCR: Use Cases and requirements	Sammlung von Anwendungsfällen
RIF-Test: Test Cases	Konformitätstests für RIF-Implementierungen

- ist der einfachste RIF Dialekt
- Ein Core Dokument besteht aus
 - Direktiven wie Import Prefixeinstellung für URIs
 - eine Sequenz von logischen Schlussfolgerungen

RIF Core Beispiel

```
Document(  
  Prefix(cpt http://example.com/concepts#)  
  Prefix(person http://example.com/people#)  
  Prefix(isbn http://.../isbn/)  
  
  Group  
  (  
    Forall ?Buyer ?Book ?Seller (  
      cpt:buy(?Buyer ?Book ?Seller):- cpt:sell(?Seller ?Book ?Buyer)  
    )  
    cpt:sell(person:John isbn:000651409X person:Mary)  
  )  
)
```

Daraus lässt sich folgende Beziehung ableiten:

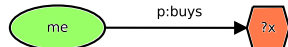
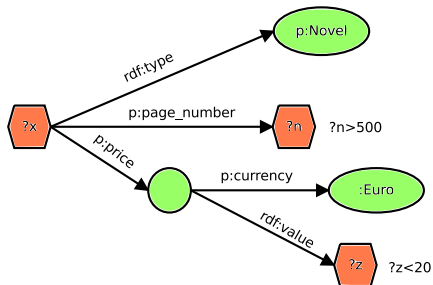
```
cpt:buy(person:Mary isbn:00065409X person:John)
```

- Datalog als Basis
- enthält Schnittmenge von RIF-BLD (Basic Logic Dialect) und RIF-PRD (Production Rule Dialect)
- einige Erweiterungen: Datentypen (RIF-DTB), IRIs, Teile von F-Logik
- Forward-Chaining möglich

- Typisches Szenario:
 - die Daten der Anwendung sind in RDF verfügbar
 - die Regeln für die Daten werden durch RIF beschrieben
 - ein RIF Prozessor erzeugt neue Beziehungen
- RDF + RIF kompatibel:
 - RDF Triples sind in RIF Repräsentierbar

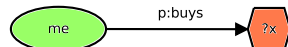
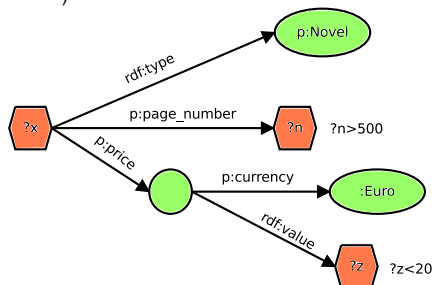
Beispiel in Turtle-basierter Syntax

```
{  
  ?x rdf:type p:Novel;  
  p:page_number ?n;  
  p:price [  
    p:currency :Euro;  
    rdf:value ?z  
  ].  
  ?n > "500"^^xsd:integer.  
  ?z < "20.0"^^xsd:double.  
}  
=>  
{ <me> p:buys ?x }
```



Das Gleiche mit RIF Präsentationssyntax

```
Document (
  Prefix ...
  Group (
    Forall ?x ?n ?z (
      <me>[p:buys->?x] :-
      And (
        ?x rdf:type p:Novel
        ?x[p:page_number->?n p:price->_abc]
        _abc[p:currency->:Euro rdf:value->?z]
        External ( pred:numeric-greater-than(?n "500"^^xsd:integer) )
        External ( pred:numeric-less-than(?z "20.0"^^xsd:double) )
      )
    )
  )
)
```



Neue Beziehungen entdecken...

```
Forall ?x ?n ?z (  
<me>[p:buys->?x] :-  
  And(  
    ?x rdf:type p:Novel  
    ?x[p:page_number->?n p:price->_abc]  
    _abc[p:currency->:Euro rdf:value->?z]  
    External ( pred:numeric-greater-than(?n "500"^^xsd:integer) )  
    External ( pred:numeric-less-than(?z "20.0"^^xsd:double) )  
  )  
)
```

Neue Beziehungen entdecken...

```
Forall ?x ?n ?z (  
<me>[p:buys->?x] :-  
  And(  
    ?x rdf:type p:Novel  
    ?x[p:page_number->?n p:price->_abc]  
    _abc[p:currency->:Euro rdf:value->?z]  
    External ( pred:numeric-greater-than(?n "500"^^xsd:integer) )  
    External ( pred:numeric-less-than(?z "20.0"^^xsd:double) )  
  )  
)
```

in Kombination mit:

```
<http://.../isbn/...> a p:Novel;  
  p:page_number "600"^^xsd:integer;  
  p:price [rdf:value "15.0"^^xsd:double ; p:currency :Euro] .
```

Neue Beziehungen entdecken...

```
Forall ?x ?n ?z (
<me>[p:buys->?x] :-
  And(
    ?x rdf:type p:Novel
    ?x[p:page_number->?n p:price->_abc]
    _abc[p:currency->:Euro rdf:value->?z]
    External ( pred:numeric-greater-than(?n "500"^^xsd:integer) )
    External ( pred:numeric-less-than(?z "20.0"^^xsd:double) )
  )
)
```

in Kombination mit:

```
<http://.../isbn/...> a p:Novel;
  p:page_number "600"^^xsd:integer;
  p:price [rdf:value "15.0"^^xsd:double ; p:currency :Euro] .
```

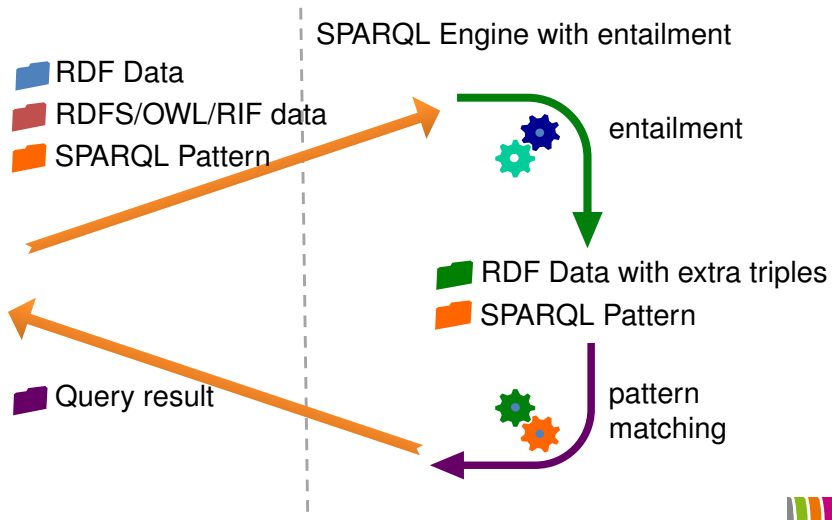
führt zu:

```
<me> p:buys <http://.../isbn/...> .
```

Was ist mit OWL 2 RL?

- OWL 2 RL steht für OWL Rule Language
- OWL 2 RL ist die Schnittmenge von RIF Core und OWL
 - Folgerungen in OWL RL können mit RIF Regeln ausgedrückt werden
 - RIF Core engines können sich wie OWL RL engines verhalten
 - wie beschrieben im Dokument RIF-OWLRL kann OWL 2 RL direkt in RIF verarbeitet werden

Ausblick: Kombination von SPARQL 1.1 und RIF



Gliederung

- 1 Einleitung und Motivation
- 2 Regelsprachen und das Semantic Web
- 3 Datalog
- 4 Regeln für OWL: SWRL
- 5 Description Logic Rules
- 6 Rule Interchange Format (RIF)
- 7 Zusammenfassung

Prädikatenlogische Regelerweiterungen für OWL DL

- Datalog als gut bekannter Formalismus
- Kombination mit OWL möglich: SWRL
- Semantik durch Erweiterung der beschreibungslogischen Interpretation von OWL
- SWRL ist unentscheidbar

Description Logic Rules

- in OWL 2 ausdrückbares SWRL-Fragment
- indirekte Unterstützung durch alle OWL-2-Tools
- Definition und Algorithmus basieren auf Abhängigkeitsgraph

RIF (Rule Interchange Format)

- W3C-Standard zum Austausch von Regeln
- erweiterbare Familie von Sprachen

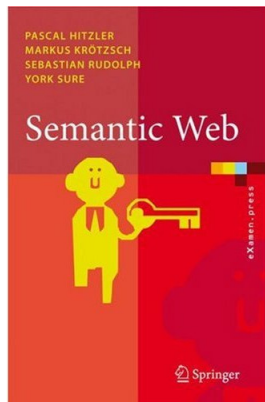
Auch relevant:

- **SPARQL 1.1 Entailment Regimes**
- **konjunktive Anfragen für OWL-DL**
- **DL-safe rules** (Variablen können nur Konstanten als Werte annehmen)

Pascal Hitzler
Markus Krötzsch
Sebastian Rudolph
York Sure

Semantic Web Grundlagen

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



Weitere Quellen:

- Ivan Herman “Introduction to Semantic Web Technologies” (2010 Semantic Technology Conference)
- W3C-Spezifikationsseiten z.B. zu RIF