

Regeln und OWL

Markus Krötzsch

Institut für Angewandte Informatik und
Formale Beschreibungsverfahren (AIFB)

Folien zur Vorlesung
Intelligente Systeme im WWW

Pascal Hitzler, York Sure, Markus Krötzsch

Sommer 2005

Warum man Regeln braucht

OWL hat verschiedene Einschränkungen:

- OWL ist oft **nicht ausdrucksstark genug**. Beispiel:
 $(\forall X)(\forall Y)(\forall Z)\text{bruder}(Y, Z) \wedge \text{vater}(X, Y) \rightarrow \text{onkel}(X, Z)$
nicht in OWL darstellbar
- OWL ist **statisch**: es dient zur Repräsentation von Wissen, nicht zur Programmierung.
- OWL ist entscheidbar: es kann grundsätzlich nicht alles Programmierbare ausdrücken (*Halteproblem*).
- OWL wird nicht „abgearbeitet“, es ist **nicht prozedural**: Erweiterungen (Built-ins) sind nur schwer zu realisieren.

Was sind Regeln?

- ① Logische Regeln (FOL):
 - „ $F \rightarrow G \equiv \neg F \vee G$ “
 - Logische Erweiterung der Wissensbasis \rightsquigarrow **statisch**
 - Open World
 - **Deklarativ** (beschreibend)
- ② Prozedurale Regeln (Business Rules):
 - „*If X then Y else Z*“
 - Ausführbare Maschinen-Anweisungen \rightsquigarrow **dynamisch**
 - **Operational** (Bedeutung = Effekt bei Ausführung)
- ③ Logikprogrammierung et al. (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**

Algorithmische Auswertung von Regeln

Forward-Chainig vs. Backward-Chaining:

- *Forward*: Evaluierung Bottom-Up

Antezedenz \rightarrow Konsequenz

Z.B. Prozesse innerhalb eines Arbeitsablaufs, prozedurale Regeln

- *Backward*: Evaluierung Top-Down

Konsequenz \leftarrow Antezedenz

Z.B. Rekursive Wissensrepräsentationen,
Logikprogrammierung (SLD-Resolution)

\rightsquigarrow Erweiterungen und Built-Ins müssen Abarbeitungsstrategie berücksichtigen

Verschiedene Vorschläge für OWL+Rules

- ① Regeln als FOL-Erweiterung von OWL (deklarativ):
Wichtigster Vertreter: **SWRL**
- ② Regeln *on top of* OWL (semi-deklarativ):
z.B. Einbindung von OWL-Wissensbasen in
Logikprogramme, zum Beispiel über Anfragen an externen
OWL-Reasoner
- ③ Regeln neben OWL (semi-deklarativ):
Möglicher Vertreter: **F-Logik**
Interaktion mit OWL durch gemeinsames Fragment

Aller Voraussicht nach wird noch 2005 eine W3C Working Group eingerichtet, die – vermutlich innerhalb von 3 Jahren – einen geeigneten Standard entwickeln soll.

Pro und Contra Regelerweiterungen (1)

FOL-Erweiterung von OWL

Vorteile

- Klare Semantik
- Nahtlose Integration in OWL
- Homogenes, voll deklaratives System

Nachteile

- Viele Nachteile von OWL bestehen weiterhin

Regeln *on top of* OWL

Vorteile

- Programmieren möglich (operationale Aspekte)
- Built-Ins

Nachteile

- verschiedene Semantiken
- heterogenes System
- unhandlich

Pro und Contra Regelerweiterungen (2)

Regeln neben OWL

Vorteile

- Programmieren möglich
- Built-Ins
- Homogenes System

Nachteile

- Semantik unklar oder sogar unvereinbar
- neuer inkompatibler Standard

Hornklauseln als Regeln

- *Klauseln* sind Disjunktionen von atomaren Aussagen oder negierten atomaren Aussagen
- *Hornklauseln* sind Klauseln mit genau einem nicht-negierten Atom:

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

↔ Semantisch äquivalente Schreibweise als Regel:

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

- Ein *definites* (oder *positives*) *Logikprogramm* ist eine endliche Menge solcher Hornklauseln
- Hohe Bedeutung von Hornklauseln, da (verhältnismäßig) effiziente Algorithmen zur Auswertung

Lloyd-Topor-Transformationen (zwei)

Konjunktionen im Kopf sind implizit erlaubt, da sie in Hornklauseln umgewandelt werden können.

Beispiel: $H_1 \wedge H_2 \leftarrow A$ ist äquivalent zu

$$(H_1 \leftarrow A) \wedge (H_2 \leftarrow A)$$

Gleiches gilt für **Disjunktionen im Rumpf**.

Beispiel: $H \leftarrow A_1 \vee A_2$ ist äquivalent zu

$$(H \leftarrow A_1) \wedge (H \leftarrow A_2)$$

Horn-Fragment von OWL: Description Logic Programs

Idee

DLP-Axiome sind OWL-Axiome, die sich als Hornklauseln darstellen lassen.

Beispiel: Umwandlung von OWL-Axiomen in *Klauselform* (wie in Logikteil besprochen). Entsteht eine Hornklausel ohne Skolemsymbole (Funktionen/Konstanten), so gehört das OWL-Axiom zu DLP.

- Käse \sqsubseteq PizzaBelag \mapsto PizzaBelag(X) \leftarrow Käse(X)
- Manchmal sind intelligentere Umwandlungen nötig:
Käse \sqsubseteq PizzaBelag \sqcup PizzaBelag \mapsto
PizzaBelag(X) \vee PizzaBelag(X) \leftarrow Käse(X)
 \rightsquigarrow äquivalent zu PizzaBelag(X) \leftarrow Käse(X)

Description Logik Programs

Uneingeschränkt erlaubt sind:*

`Class, Thing, subclassOf, Property,
subPropertyOf, domain, range, Individual,
equivalentClass, equivalentProperty, sameAs,
differentFrom, AllDifferent, DatatypeProperty,
ObjectProperty, inverseOf, TransitiveProperty,
SymmetricProperty, FunctionalProperty,
Inverse-FunctionalProperty, intersectionOf`
(* Benötigt Gleichheit = und *Integrity Constraints*)

Andere OWL-Konstrukte sind nur eingeschränkt erlaubt.

Horn-Checker und Übersetzer von OWL nach Prolog, sowie weiterführende Links und Literatur sind erhältlich unter <http://logic.aifb.uni-karlsruhe.de>.

Beispiele

- „Männer und Frauen sind Erwachsene“
OWL: $\text{mann} \sqcup \text{frau} \sqsubseteq \text{erwachsen}$
FOL : $(\text{mann}(X) \rightarrow \text{erwachsen}(X)) \wedge$
 $(\text{frau}(X) \rightarrow \text{erwachsen}(X))$
- „Ein Waisenkind ist ein Kind verstorbener Eltern“
OWL: $\text{weise} \sqsubseteq \text{kind} \sqcap \forall \text{kindVon.tot}$
FOL : $\text{weise}(X) \wedge \text{kindVon}(X, Y) \rightarrow \text{kind}(X) \wedge \text{tot}(Y)$

DLP und F-Logik

DLP ist *syntaktische* Teilmenge von F-Logik.

Problem

Closed World vs. *Open World*

↪ Einige logische Aussagen werden als „falsch“ eingeordnet, ohne dass dies logisch beweisbar wäre.

↪ F-Logik beantwortet Anfragen negativ, die in DLP als „unbekannt“ gelten würden. Dafür sind die negativen Aussagen von F-Logik nicht endgültig sicher: mit Erweiterung der Ontologie können sie revidiert werden!

Erweiterte Semantic-Web-Schichttorte oder „Twin Tower“?

Aktuelle Forschung: Kontrolliertes Zusammenspiel der verschiedenen Paradigmen ermöglichen

DLP: Aktueller Status

- Syntaktischer Kompromiss zwischen F-Logic und OWL.
- Effizientes Fragment, genügt für viele state-of-the-art Anwendungen.
- Verwendet in aktuellen großen Forschungsprojekten, z.B. SEKT (EU) und SMARTWEB (BMBF).
- Mittelfristig nicht ausdrucksstark genug.

AIFB DLP Seite: <http://logic.aifb.uni-karlsruhe.de>

Semantic Web Rules Language (SWRL)

Idee

Erweiterung von OWL durch eingeschränkte **FOL-Regeln** der Form

$$\langle \text{antecedent} \rangle \Rightarrow \langle \text{consequent} \rangle$$

erweitert durch **Built-Ins**.

Einschränkung: $\langle \text{antecedent} \rangle$ und $\langle \text{consequent} \rangle$ sind **Konjunktionen** von Ausdrücken der Form

- 1 $C(x)$ oder $P(x, y)$, mit C/P OWL-Klasse/-Property
- 2 `sameAs(x, y)`, `differentfrom(x, y)`, `builtIn(r, x, ...)`

wobei x, y Variablen, OWL-Individuen oder Elemente einer OWL *concrete domain* sind.

Built-Ins existieren für Strings, Zahlen, Zeiten, Listen, etc.

SWRL: Semantik

SWRL-Regeln werden als Hornklauseln im Sinne der FOL interpretiert.

↪ SWRL ist ein Fragment von FOL.

Sowohl $\langle \text{antecedent} \rangle$ als auch $\langle \text{consequent} \rangle$ können aus 0 Atomen bestehen (d.h. leer sein):

- $\langle \text{antecedent} \rangle$ leer:
 $\langle \text{consequent} \rangle$ muss erfüllt sein.
- **Leeres** $\langle \text{consequent} \rangle$:
 $\langle \text{antecedent} \rangle$ darf nicht erfüllt sein.

Variablen werden manchmal durch ein vorgestelltes Fragezeichen ausgewiesen.

Beispiele

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

Bewertung von SWRL

Vorteile

- nahtlose Integration mit OWL
- klare und einfache Semantik
- Fragment von FOL

Nachteile

- unentscheidbar
- keine Unterstützung semi-deklarativer Programmieraspekte
- Fragment von FOL

DL-safe rules

Idee

SWRL-ähnliche Regeln soweit einschränken, dass Entscheidbarkeit garantiert ist.

- Beliebige funktionenfreie Hornklauseln erlaubt, wobei OWL-Klassen und -Rollennamen (DL-Atome) eingebaut werden dürfen
- Regeln müssen **DL-safe** sein: Jede Variable muss auch in einem nicht-DL-Ausdruck im Rumpf auftreten!
- Semantik: Auswertung als Fragment von FOL.

Grundidee: Erzwingen von DL-safeness

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 DL-Individuen a angelegt wird.

Beispiel

Wissensbasis

father(cain, adam)

father(abel, adam)

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

hates(romulus, remus)

hates(cain, abel)

BadChild(X) \leftarrow father(X , Z), father(Y , Z), hates(X , Y)

BadChild_s(X) \leftarrow father(X , Z), father(Y , Z), hates(X , Y),
 $O(X)$, $O(Y)$, $O(Z)$

$O(\text{cain})$ $O(\text{abel})$ $O(\text{remus})$...

für alle Individuen der DL-Wissensbasis (A-Box)

Logische Konsequenzen:

BadChild(cain), BadChild(romulus), BadChild_s(cain), ...

Aber: BadChild_s(romulus) **keine** logische Konsequenz

DL-safe Rules und KAON2

- KAON2 (kaon2.semanticweb.org) ist ein Softwarepaket zum Bearbeiten von OWL-DL und SWRL Ontologien, entwickelt durch FZI und AIFB.
- System basiert auf Transformation von OWL nach *Datalog* (ein Pradigma der Logikprogrammierung).
 - ~> DL-safe Rules können untransformiert hinzugefügt werden.
- Einschränkung: Es dürfen dabei nur „simple roles“ verwendet werden.

Kapselung von OWL

OWL-Wissen ist **statisch**. Um damit zu programmieren, muss man Ausdrucksmöglichkeiten einer Programmiersprache hinzufügen.

Idee

Verwende logikbasierte (semi-deklarative) Programmiersprache (z.B. Prolog) mit prozeduralen Aspekten. Anfragen an OWL geschehen mit Hilfe von Built-ins, die einen OWL-Reasoner aufrufen.

↪ Resultat: Logikprogrammierung mit gekapselter OWL-Reasoning-Funktionalität. Ein **hybrides System**.

Ausblick: Wohin geht die Entwicklung?

- OWL ist als Standard etabliert, auch bereits außerhalb des Semantic-Web. Es wird in Theorie und Praxis verwendet.
- F-Logic spielt eine wichtige Rolle bei der praktischen Umsetzung semantischer Technologien.
- Die Industrie hat großes Interesse an der schnellen Standardisierung von (einfachen) Business- und Process-Rules, vermutlich *on top of* OWL.
- Die Praxis zeigt: Es wird alles irgendwie benutzt und weiterentwickelt werden. Die Standards (alte und neue) werden sich aber weiter verbreiten.

Literatur

- Hornklauseln/Prolog: J.W. Lloyd, **Foundations of logic programming** (second, extended edition). Springer-Verlag, 1987.
- DLP: <http://logic.aifb.uni-karlsruhe.de> für Links und Literatur
- SWRL: <http://www.w3.org/Submission/SWRL/>
- DL-safe Rules: B. Motik, U. Sattler, R. Studer, **Query Answering for OWL-DL with Rules**. In: Proc. of the 3rd Int. Semantic Web Conference (ISWC 2004), Hiroshima, Japan, November, 2004.