

Regeln für OWL

Markus Krötzsch Sebastian Rudolph

Institut AIFB · Karlsruher Institut für Technologie (KIT)

Semantic Web Technologies 1 (WS09/10)

3. Februar 2010

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

Die nichtkommerzielle Vervielfältigung, Verbreitung und Bearbeitung dieser Folien ist gestattet (→ Lizenzbestimmungen CC-BY-NC).

- 1 Einleitung und Wiederholung
- 2 Logisches Schließen in SWRL
- 3 Description Logic Rules
- 4 DL-safe Rules
- 5 Zusammenfassung

Semantic Web Technologies 1

- 1 Einleitung und XML
- 2 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 OWL 2
- 9 SPARQL – Syntax und Intuition
- 10 Semantik von SPARQL
- 11 Konjunktive Anfragen/Einführung Regelsprachen
- 12 Semantic Web – Anwendungen
- 13 **Regeln für OWL** (→ Webseite)
- 14 Ontology Engineering

Literaturhinweise siehe → Webseite dieser Vorlesung

Problem

Ontologiesprache OWL DL für verschiedene Anwendungen zu schwach

- Konjunktive Anfragen: Abfrage von Instanzen in OWL-DL-Wissensbasen
- SWRL: Erweiterung von OWL DL mit Datalog-Regeln
↪ Konjunktive Anfragen in SWRL darstellbar durch einzelne Regeln
- OWL 2: Erweiterung von OWL, logische und nicht-logische Änderungen
↪ OWL 2 DL ebenfalls mit konjunktiven Anfragen und SWRL erweiterbar

SWRL-Beispiel (Wiederholung)

Kombinierte SWRL-Wissensbasis* (Datalog + Beschreibungslogik):

- (1) $\text{Vegetarier}(x) \wedge \text{Fischprodukt}(y) \rightarrow \text{magNicht}(x, y)$
- (2) $\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \rightarrow \text{Unzufrieden}(x)$
- (3) $\text{hatBestellt}(x, y) \rightarrow \text{Gericht}(y)$
- (4) $\text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enth\u00e4lt}(y, z) \rightarrow \text{magNicht}(x, y)$
- (5) $\rightarrow \text{Vegetarier}(\text{markus})$
- (6) $\text{Zufrieden}(x) \wedge \text{Unzufrieden}(x) \rightarrow$
- (7) $\exists \text{hatBestellt.ThaiCurry}(\text{markus})$
- (8) $\text{ThaiCurry} \sqsubseteq \exists \text{enth\u00e4lt.Fischprodukt}$

Wir k\u00f6nnen folgern: $\text{Unzufrieden}(\text{markus})$

* Wir meinen mit „SWRL“ auch in dieser Vorlesung die freie Kombination von Datalog & Beschreibungslogik. Das entspricht nicht ganz urspr\u00fcnglichen Definition von SWRL f\u00fcr OWL (1) DL.

- 1 Einleitung und Wiederholung
- 2 Logisches Schließen in SWRL**
- 3 Description Logic Rules
- 4 DL-safe Rules
- 5 Zusammenfassung

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).

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.

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?

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

- 1 Einleitung und Wiederholung
- 2 Logisches Schließen in SWRL
- 3 Description Logic Rules**
- 4 DL-safe Rules
- 5 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

Klassenausdrücke

Klassenamen	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	$R_1 \sqsubseteq R_2$
Allgemeine 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 $R(x, y) \rightarrow S(x, y)$

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:

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

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:

- $\text{Zufrieden} \sqcap \text{Unzufrieden} \sqsubseteq \perp$ entspricht
 $\text{Zufrieden}(x) \wedge \text{Unzufrieden}(x) \rightarrow$
- $\exists \text{wohnort}.\exists \text{liegtIn.EULand} \sqsubseteq \text{EUBürger}$ 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 enthalt Klassenausdrucke \rightsquigarrow nicht durch Subproperty-Axiom darstellbar

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:

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

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:

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

Die Grenzen von Description Logic Rules

Nicht alle SWRL-Regeln können so dargestellt werden!

Nicht alle SWRL-Regeln können so dargestellt werden!

Beispiel:

$\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \rightarrow \text{Unzufrieden}(x)$
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

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) $\text{Zufrieden}(x) \wedge \text{Unzufrieden}(x) \rightarrow$

Regel (2) $\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \rightarrow \text{Unzufrieden}(x)$
ist keine DL Rule

Anmerkung: Description Logic Rules müssen nach Umwandlung in *SROIQ* natürlich auch die Bedingungen and 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, 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, 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 *SR0IQ* (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, 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.

Die Regel kann jetzt in *SRFIQ* 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.)

- 1 Einleitung und Wiederholung
- 2 Logisches Schließen in SWRL
- 3 Description Logic Rules
- 4 DL-safe Rules**
- 5 Zusammenfassung

Beobachtung

Datalog ist entscheidbar, weil Regeln nur auf endlich viele Arten angewendet werden müssen: Variablen stehen nur für Konstanten.

- Variablen in SWRL könnten für unendlich viele „geschlussfolgerter“ Individuen stehen.
- Ziel: Regeln durch Datalog-Prädikate „absichern“ um Variablenbelegung einzuschränken
- **DL-safe Rules** als weiteres entscheidbares Fragment von SWRL

Diesmal enthalten Regeln auch Nicht-DL-Atome:

- Ein **Datalog-Atom** ist ein Atom, dessen Prädikatsymbol in keinem *SROIQ*-Axiom der Wissensbasis vorkommt.

Diesmal enthalten Regeln auch Nicht-DL-Atome:

- Ein **Datalog-Atom** ist ein Atom, dessen Prädikatsymbol in keinem *SROIQ*-Axiom der Wissensbasis vorkommt.

Eine SWRL-Regel ist **DL-safe** wenn gilt:

- Jede Variable der Regel kommt auch in einem Datalog-Atom im Rumpf vor.

⇒ Variablenbelegungen für Datalog-Atome können letztlich nur Konstanten entsprechen!

Beispiel:

$\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \rightarrow \text{Unzufrieden}(x)$

\rightsquigarrow nicht DL-safe, wenn „hatBestellt“ und „magNicht“ in DL-Axiomen vorkommen.

Beispiel:

$\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \rightarrow \text{Unzufrieden}(x)$
 \rightsquigarrow nicht DL-safe, wenn „hatBestellt“ und „magNicht“ in DL-Axiomen vorkommen.

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

$\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \wedge O(x) \wedge O(y) \rightarrow$
 $\text{Unzufrieden}(x)$

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

\rightsquigarrow Regel nur noch auf bekannte OWL-Individuen anwendbar

DL-safe Rules in der Praxis

- OWL 2 DL mit DL-safe Rules ist entscheidbar
- Naive Implementierung: jede Regel ist ausdrückbar durch viele Regeln, indem man alle Variablen durch Konstanten ersetzt (auf jede denkbare Weise!)
- Keine größere *theoretische* Komplexität der Berechnung

Implementierungen (02.02.2010):

- HermiT: DL-safe Rules für OWL 2 (→ Webseite)
- KAON2: sehr effiziente Umsetzung von DL-safe Rules, Unterstützung für disjunktive Regeln, beschränkt auf Teil von OWL 1 (→ 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)

Ein kombiniertes Beispiel (I)

SROIQ + Description Logic Rules + DL-safe Rules weiterhin entscheidbar:

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

$$(2) \text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \\ \wedge O(x) \wedge O(y) \rightarrow \text{Unzufrieden}(x)$$

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

$$(4) \text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enth\u00e4lt}(y, z) \rightarrow \text{magNicht}(x, y)$$

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

$$(6) \text{Zufrieden}(x) \wedge \text{Unzufrieden}(x) \rightarrow$$

$$(7) \exists \text{hatBestellt}.\text{ThaiCurry}(\text{markus})$$

$$(8) \text{ThaiCurry} \sqsubseteq \exists \text{enth\u00e4lt}.\text{Fischprodukt}$$

$$(9) O(\text{markus})$$

Ein kombiniertes Beispiel (I)

SROIQ + Description Logic Rules + DL-safe Rules weiterhin entscheidbar:

- (1) $\text{Vegetarier}(x) \wedge \text{Fischprodukt}(y) \rightarrow \text{magNicht}(x, y)$
- (2) $\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \wedge O(x) \wedge O(y) \rightarrow \text{Unzufrieden}(x)$
- (3) $\text{hatBestellt}(x, y) \rightarrow \text{Gericht}(y)$
- (4) $\text{magNicht}(x, z) \wedge \text{Gericht}(y) \wedge \text{enth\u00e4lt}(y, z) \rightarrow \text{magNicht}(x, y)$
- (5) $\rightarrow \text{Vegetarier}(\text{markus})$
- (6) $\text{Zufrieden}(x) \wedge \text{Unzufrieden}(x) \rightarrow$
- (7) $\exists \text{hatBestellt}.\text{ThaiCurry}(\text{markus})$
- (8) $\text{ThaiCurry} \sqsubseteq \exists \text{enth\u00e4lt}.\text{Fischprodukt}$
- (9) $O(\text{markus})$

Wir k\u00f6nnen **nicht** folgern: $\text{Unzufrieden}(\text{markus})$

Ein kombiniertes Beispiel (II)

SROIQ + Description Logic Rules + DL-safe Rules weiterhin entscheidbar:

- (1) $\text{Vegetarier}(x) \wedge \text{Fischprodukt}(y) \rightarrow \text{magNicht}(x, y)$
- (2) $\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \wedge O(x) \wedge O(y) \rightarrow \text{Unzufrieden}(x)$
- (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) $\text{Zufrieden}(x) \wedge \text{Unzufrieden}(x) \rightarrow$
- (7) $\text{hatBestellt}(\text{markus}, \text{redThaiCurry})$
 $\text{ThaiCurry}(\text{redThaiCurry})$
- (8) $\text{ThaiCurry} \sqsubseteq \exists \text{enthält.Fischprodukt}$
- (9) $O(\text{markus}) \quad O(\text{redThaiCurry})$

Ein kombiniertes Beispiel (II)

SROIQ + Description Logic Rules + DL-safe Rules weiterhin entscheidbar:

- (1) $\text{Vegetarier}(x) \wedge \text{Fischprodukt}(y) \rightarrow \text{magNicht}(x, y)$
- (2) $\text{hatBestellt}(x, y) \wedge \text{magNicht}(x, y) \wedge O(x) \wedge O(y) \rightarrow \text{Unzufrieden}(x)$
- (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) $\text{Zufrieden}(x) \wedge \text{Unzufrieden}(x) \rightarrow$
- (7) $\text{hatBestellt}(\text{markus}, \text{redThaiCurry})$
 $\text{ThaiCurry}(\text{redThaiCurry})$
- (8) $\text{ThaiCurry} \sqsubseteq \exists \text{enthält.Fischprodukt}$
- (9) $O(\text{markus}) \quad O(\text{redThaiCurry})$

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

- 1 Einleitung und Wiederholung
- 2 Logisches Schließen in SWRL
- 3 Description Logic Rules
- 4 DL-safe Rules
- 5 Zusammenfassung**

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

DL-safe Rules

- SWRL-Fragment, in dem Variablen nur Konstanten als Werte annehmen
- Unterstützung durch einige OWL-Reasoner
- frei mit Description Logic Rules kombinierbar

Pascal Hitzler
Markus Krötzsch
Sebastian Rudolph

Foundations of Semantic Web Technologies

CRC Press 2009, 455 S., Hardcover
ISBN: 9781420090505

Aktuelle Literaturhinweise online:
[Vorlesung 13](#)

