Knowledge Representation for the Semantic Web

Winter Quarter 2010

Slides 11 – 03/08/2010

Pascal Hitzler
Kno.e.sis Center
Wright State University, Dayton, OH
http://www.knoesis.org/pascal/
Slides are based on

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies

Chapman & Hall/CRC, 2010

Flyer with special offer is available.

http://www.semantic-web-book.org
Today: Wrap-Up

- Main Messages
- What Is Semantics – Revisited
- OWL At Its Expressive Limits
- A Very Personal Semantic Web History
- Making OWL Fit For Practice
Main Messages

• How to model in RDF and OWL

• What is model-theoretic semantics

• How to compute logical consequences in RDF and OWL
Today: Wrap-Up

- Main Messages
- What Is Semantics – Revisited
- OWL At Its Expressive Limits
- A Very Personal Semantic Web History
- Making OWL Fit For Practice
What Is Semantic Web Semantics?

• Opinions Differ. Here’s my take.

• Semantic Web requires a *computable* semantics.

• I.e., the semantics must be a formal entity which is clearly defined and automatically computable.

• Ontology languages provide this by means of their formal semantics.

• Semantic Web Semantics is given by a relation – the *logical consequence* relation.

• Note: This is considerably more than saying that the semantics of an ontology is the set of its logical consequences!
Today: Wrap-Up

- Main Messages
- What Is Semantics – Revisited
- OWL At Its Expressive Limits
- A Very Personal Semantic Web History
- Making OWL Fit For Practice
OWL At Its Expressive Limits

• There are a lot of things that cannot be said in OWL.

• We will talk about a few such things and general ideas how to address them.
If I ask for soccer team members, I also want to get the goalkeepers listed ...

If I ask for cities, I also want all capitals listed ...

covered by OWL

inheritance reasoning
Less Simple Reasoning

What was again the name of that Russian researcher who worked on resolution-based calculi for EL?

Are lobsters spiders?

What is "Käuzchen" in English?

Covered by OWL – given enough data

Answering requires merging of knowledge from many websites and using background knowledge.
Sophisticated Application Needs

The conclusions from the sensor data are uncertain. How do I process that? *uncertainty reasoning*

Merging different sources yields inconsistencies. How do we deal with that? *paraconsistent reasoning*

Thinkpads run Windows, unless explicitly stated otherwise ... *default reasoning*

*not covered by OWL*
Some things you can say in OWL

*Rules* are often considered an intuitive form of knowledge representation

- **Man**(x) \(\land\) **hasBrother**(x,y) \(\land\) **hasChild**(y,z) → **Uncle**(x)
  - Man \(\cap\) \(\exists\)hasBrother.\(\exists\)hasChild. \(\top\) ⊆ Uncle

- **ThaiCurry**(x) → \(\exists\)contains.FishProduct(x)
  - ThaiCurry ⊆ \(\exists\)contains.FishProduct

- **kills**(x,x) → **suicide**(x)                     suicide(x) → **kills**(x,x)
  - \(\exists\)kills.Self ⊆ suicide  suicide ⊆ \(\exists\)kills.Self

*Note:* with these two axioms, 

suicide is basically the same as kills
Some things you can say in OWL

- **NutAllergic(x) ∧ NutProduct(y) → dislikes(x,y)**
  - NutAllergic ≡ ∃nutAllergic.Self
  - NutProduct ≡ ∃nutProduct.Self
  - nutAllergic ◦ U ◦ nutProduct ⊆ dislikes

- **dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)**
  - Dish ≡ ∃dish.Self
  - dislikes ◦ contains¬ ◦ dish ⊆ dislikes

- **worksAt(x,y) ∧ University(y) ∧ supervises(x,z) ∧ PhDStudent(z) → professorOf(x,z)**
  - ∃worksAt.University ≡ ∃worksAtUniversity.Self
  - PhDStudent ≡ ∃phDStudent.Self
  - worksAtUniversity ◦ supervises ◦ phDStudent ⊆ professorOf
DL Rules: definition

- Tree-shaped bodies
- First argument of the conclusion is the root

\[ C(x) \land R(x,a) \land S(x,y) \land D(y) \land T(y,a) \rightarrow E(x) \]
\[ C \sqcap \exists R.\{a\} \sqcap \exists S.\{D \sqcap \exists T.\{a\}\} \subseteq E \]
DL Rules: definition

- Tree-shaped bodies
- First argument of the conclusion is the root

\[ C(x) \land R(x,a) \land S(x,y) \land D(y) \land T(y,a) \rightarrow V(x,y) \]

\[ C \sqcap \exists R.\{a\} \sqsubseteq \exists R1.\text{Self} \]
\[ D \sqcap \exists T.\{a\} \sqsubseteq \exists R2.\text{Self} \]
\[ R1 \circ S \circ R2 \sqsubseteq V \]
DL Rules: definition

- Tree-shaped bodies
- First argument of the conclusion is the root
- Complex classes are allowed in the rules
  - Mouse(x) ∧ ∃hasNose.TrunkLike(y) → smallerThan(x,y)
  - ThaiCurry(x) → ∃contains.FishProduct(x)

Note: This allows to reason with unknowns (unlike rules)

- Allowed class constructors depend on the chosen underlying description logic!

SROIQ Rules can be transformed back into SROIQ!
Outside SROIQ Rules

\[ \text{Pair}(p) \land \text{consistOf}(p, x) \land \text{consistOf}(p, y) \land \text{differentFrom}(x, y) \land \]
\[ \text{River}(r) \land \text{inBetween}(r, p) \land \text{rightBankOf}(x, r) \rightarrow \text{leftBankOf}(y, r). \]

- Cannot be expressed in SROIQ (is not a SROIQ Rule).
- Extending OWL with such more general rules leads to undecidability.

[Example due to Dong-Po Deng, presented at GeoS2009]
Pair(p) \land \text{consistOf}(p, x) \land \text{consistOf}(p, y) \land \text{differentFrom}(x, y) \land 
\text{River}(r) \land \text{inBetween}(r, p) \land \text{rightBankOf}(x, r) \rightarrow \text{leftBankOf}(y, r).

- Read rule as a first-order predicate logic formula.

Semantically okay, but leads to undecidability in combination with OWL.
Pair(p) \land \text{consistOf}(p, x) \land \text{consistOf}(p, y) \land \text{differentFrom}(x, y) \land
\text{River}(r) \land \text{inBetween}(r, p) \land \text{rightBankOf}(x, r) \rightarrow \text{leftBankOf}(y, r).

- Semantically restrict rule, such that it applies only to individuals which are explicitly contained in the knowledge base. I.e., those with known URIs.

- DL-safe SWRL combined with OWL is decidable.

- Formalism supported, e.g., by Pellet.
SWRL example

NutAllergic(sebastian)
NutProduct(peanutOil)
∃orderedDish.ThaiCurry(sebastian)

ThaiCurry ⊑ ∃contains.{peanutOil}
T ⊑ ∀orderedDish.Dish

NutAllergic(x) ∧ NutProduct(y) → dislikes(x,y)
dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)
orderedDish(x,y) ∧ dislikes(x,y) → Unhappy(x)
SWRL example

NutAllergic(sebastian)
NutProduct(peanutOil)
∃orderedDish.ThaiCurry(sebastian)

ThaiCurry ⊑ ∃contains.{peanutOil}
T ⊑ ∀orderedDish.Dish

NutAllergic(x) ∧ NutProduct(y) → dislikes(x,y)
dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)
orderedDish(x,y) ∧ dislikes(x,y) → Unhappy(x)

Conclusions:
dislikes(sebastian,peanutOil)
SWRL example

NutAllergic(sebastian)
NutProduct(peanutOil)
∃orderedDish.ThaiCurry(sebastian)

ThaiCurry ⊑ ∃contains.{peanutOil}

T ⊑ ∀orderedDish.Dish

NutAllergic(x) ∧ NutProduct(y) → dislikes(x,y)
dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)
orderedDish(x,y) ∧ dislikes(x,y) → Unhappy(x)

Conclusions:
dislikes(sebastian,peanutOil)
orderedDish(sebastian,y_s)
ThaiCurry(y_s)
Dish(y_s)
SWRL example

NutAllergic(sebastian)
NutProduct(peanutOil)
\exists orderedDish.ThaiCurry(sebastian)

\textit{ThaiCurry} \subseteq \exists contains.\{peanutOil\}
\top \subseteq \forall orderedDish.Dish

\textit{NutAllergic(x)} \land \textit{NutProduct(y)} \rightarrow \textit{dislikes(x,y)}
\textit{dislikes(x,z)} \land \textit{Dish(y)} \land \textit{contains(y,z)} \rightarrow \textit{dislikes(x,y)}
\textit{orderedDish(x,y)} \land \textit{dislikes(x,y)} \rightarrow \textit{Unhappy(x)}

Conclusions:
\textit{dislikes(sebastian,peanutOil)}
\textit{orderedDish(sebastian,y_s)}
\textit{ThaiCurry(y_s)}
\textit{Dish(y_s)}
contains(y_s,peanutOil)
SWRL example

Does not work under DL-safety!

\[ \text{NutAllergic(sebastian)} \]
\[ \text{NutProduct(peanutOil)} \]
\[ \exists \text{orderedDish.ThaiCurry(sebastian)} \]

\[ \text{ThaiCurry} \subseteq \exists \text{contains.}{\text{peanutOil}} \]
\[ \text{T} \subseteq \forall \text{orderedDish.Dish} \]

\[ \text{NutAllergic(x)} \land \text{NutProduct(y)} \rightarrow \text{dislikes(x,y)} \]
\[ \text{dislikes(x,z)} \land \text{Dish(y)} \land \text{contains(y,z)} \rightarrow \text{dislikes(x,y)} \]
\[ \text{orderedDish(x,y)} \land \text{dislikes(x,y)} \rightarrow \text{Unhappy(x)} \]

Conclusions:
\[ \text{dislikes(sebastian,peanutOil)} \]
\[ \text{contains(y_s,peanutOil)} \]
\[ \text{dislikes(sebastian,y_s)} \]
\[ \text{orderedDish(sebastian,y_s)} \]
\[ \text{ThaiCurry(y_s)} \]
\[ \text{Dish(y_s)} \]
SWRL example

NutAllergic(sebastian)
NutProduct(peanutOil)
∃orderedDish.ThaiCurry(sebastian)

ThaiCurry ⊆ ∃contains.{peanutOil}
T ⊆ ∀orderedDish.Dish

NutAllergic(x) ∧ NutProduct(y) → dislikes(x,y)
dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)
orderedDish(x,y) ∧ dislikes(x,y) → Unhappy(x)

Conclusions:
dislikes(sebastian,peanutOil)
orderedDish(sebastian,y_s)
ThaiCurry(y_s)
Dish(y_s)
contains(y_s,peanutOil)
dislikes(sebastian,y_s)
Unhappy(sebastian)
SWRL example

\[
\begin{align*}
\text{NutAllergic(sebastian)} \\
\text{NutProduct(peanutOil)} \\
\exists \text{orderedDish.ThaiCurry(sebastian)}
\end{align*}
\]

\[
\begin{align*}
\text{ThaiCurry} \subseteq \exists \text{contains.}\{\text{peanutOil}\} \\
\top \subseteq \forall \text{orderedDish.Dish}
\end{align*}
\]

\[
\begin{align*}
\text{NutAllergic(x)} \land \text{NutProduct(y)} & \rightarrow \text{dislikes(x,y)} \\
\text{dislikes(x,z)} \land \text{Dish(y)} \land \text{contains(y,z)} & \rightarrow \text{dislikes(x,y)} \\
\text{orderedDish(x,y)} \land \text{dislikes(x,y)} & \rightarrow \text{Unhappy(x)}
\end{align*}
\]

Conclusion: \text{Unhappy(sebastian)}
Alternative Semantics

• SWRL and DL-safe SWRL are essentially based on the same style of model-theoretic semantics.

• If we want to deal with inconsistencies, uncertainty, or default reasoning, we have to modify the semantic approach.

• How to modify a semantics?
  – Redefine the notion of *model*!
Sophisticated Application Needs

The conclusions from the sensor data are uncertain. How do I process that?

uncertainty reasoning

Merging different sources yields inconsistencies. How do we deal with that?

paraconsistent reasoning

Thinkpads run Windows, unless explicitly stated otherwise ...

default reasoning
Paraconsistent Reasoning

• Modification: Use four truth values instead of two.
  \{true, false\} \rightarrow \{true, false, none, both\}

• Idea: “both” captures inconsistency.

• Unicorn(beauty)
  Unicorn \sqsubseteq Fictitious
  Unicorn \sqsubseteq Animal
  Animal \sqsubseteq \neg Fictitious

  would, e.g., result in the truth value “both” for Fictitious(beauty).

• Problems: Paraconsistency or bugfixing? Which of various related approaches to take? How well does it work in practice?
Uncertainty Reasoning

• Modification: Use the real unit interval as set of truth values.

• 0 is interpreted as “false”
• 1 is interpreted as “true”

• Define how to combine them. E.g.,

  HighQuality(a) is 0.7 true
  Expensive(a) is 0.8 true
  HighQuality $\sqcap$ Expensive $\sqsubseteq$ Buyable

  how “much” true is Buyable(a)?

• Problems: Different choices for combination. Does it match the intuition? Is this probabilistic or fuzzy? How reliable are the values? And it’s computationally (much) more expensive.
Default Reasoning

- Thinkpads “normally” run Windows. I.e., this is the default assumption (to be assumed unless there is evidence to the contrary).

- Thinkpad \( \sqsubseteq \forall \text{runsOS.WindowsOS} \)
  Thinkpad(myThinkpad) 
  Thinkpad(yourThinkpad) 
  runsOS(yourThinkpad,linux) 
  \( \neg \text{WindowsOS(linux)} \)

is contradictory. How do we capture the default?
Default Reasoning

- Thinkpad ⊆ ∀runsOS.WindowsOS ⊆ ExceptionThing
  Thinkpad(myThinkpad)
  Thinkpad(yourThinkpad)
  runsOS(yourThinkpad,linux)
  ¬WindowsOS(linux)

+ a semantics which “minimizes” ExceptionThing.
IAW, something is only in ExceptionThing if it is necessarily
contained in it (e.g., to avoid a contradiction).

- This idea is called *circumscription* and is due to John McCarthy
  [1980] (not for DLs, obviously).
  There exist other approaches which accomplish the same thing
  in other ways.

- Problem: Computationally very expensive.
Default Reasoning

• $\text{Thinkpad} \subseteq \forall \text{runsOS.WindowsOS} \cup \text{ExceptionThing}$
  $\text{Thinkpad(myThinkpad)}$
  $\text{Thinkpad(yourThinkpad)}$
  $\text{runsOS(yourThinkpad,linux)}$
  $\neg \text{WindowsOS(linux)}$

  + a semantics which "minimizes" ExceptionThing.
  IAW, something is only in ExceptionThing if it is necessarily contained in it (e.g., to avoid a contradiction).

• From all models $I$ of the KB, select those models, for which ExceptionThing$^I$ is minimal.
  Take these as the circumscribed models.
  Define logical consequence as usual.
Today: Wrap-Up

- Main Messages
- What Is Semantics – Revisited
- OWL At Its Expressive Limits
- A Very Personal Semantic Web History
- Making OWL Fit For Practice
A Very Personal Semantic Web History

- 2002: Growing Semantic Web Hype (I wasn’t there)

- 2004: Will it every work?

- 2006: It’s probably not going to work.

- 2008: Industry is catching on and RDF will work. But OWL won’t.

- 2010: Many major IT companies’ R&D departments investigate OWL or even have their own OWL reasoner.
Today: Wrap-Up

- Main Messages
- What Is Semantics – Revisited
- OWL At Its Expressive Limits
- A Very Personal Semantic Web History
- Making OWL Fit For Practice
• The use of formal semantics for RDF and OWL still hasn’t produced prominent applications with clear-cut added value compared with other methods/technologies. [More precisely: Such a thing hasn’t been made public.]

• Hindrances:
  – Scalability of reasoning isn’t very good (yet).
  – Few people can really model well in OWL.
  – High-quality ontologies are expensive to produce.
  – Real-life data often isn’t clean enough for reasoning.
Making OWL Fit For Practice

• Researchers have to work on:
  
  – **Scalability**, including alternative reasoning methods. Don’t get fixed on soundness/completeness/decidability.
  
  – Dissemination and education.
  
  – Methods for making real-life data fit for formal semantics.
  
  – Developing clear-cut use cases for formal semantics.

*It is essential, to leave the ivory tower!*
Some References


Some References

