OWL 2 Rules (Part 2)

Tutorial at ESWC2009
May 31, 2009

Pascal Hitzler
Markus Krötzsch
Sebastian Rudolph

AIFB, Universität Karlsruhe (TH)
http://www.pascal-hitzler.de
http://korrekt.org
http://www.sebastian-rudolph.de
References and Pointers

See http://semantic-web-grundlagen.de/wiki/ESWC09_Tutorial for the complete set of slides, and for links to references.

Main References:


Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

- Retaining tractability III: ELP

Extending OWL with Rules

Rules inside OWL

putting it all together
Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

- Retaining tractability III: ELP

Intro

Extending OWL with Rules

Rules inside OWL

putting it all together
Motivation: OWL and Rules

- Rules (mainly, logic programming) as alternative ontology modelling paradigm.
- Similar tradition, and in use in practice (e.g. F-Logic)

- Ongoing: W3C RIF working group
  - Rule Interchange Format
  - based on Horn-logic
  - language standard forthcoming 2009

- Seek: Integration of rules paradigm with ontology paradigm
  - Here: Tight Integration in the tradition of OWL
  - Foundational obstacle: reasoning efficiency / decidability
    [naive combinations are undecidable]
Preliminaries: Datalog

Essentially Horn-rules without function symbols

general form of the rules:

\[ p_1(x_1,\ldots,x_n) \land \ldots \land p_m(y_1,\ldots,y_k) \rightarrow q(z_1,\ldots,z_j) \]

semantics either as in predicate logic
or as Herbrand semantics (see next slide)

decidable

polynomial data complexity (in number of facts)

combined (overall) complexity: ExpTime

combined complexity is P if the number of variables per rule is

globally bounded
Datalog semantics example

Example:
\[ p(x) \rightarrow q(x) \]
\[ q(x) \rightarrow r(x) \]
\[ \rightarrow p(a) \]

predicate logic semantics:

\[(\forall x) (p(x) \rightarrow r(x))\]
\[
\text{and}
\[(\forall x) (\neg r(x) \rightarrow \neg p(x))\]
\text{are logical consequences}

q(a) and r(a)
\text{are logical consequences}

Herbrand semantics

those on the left are not logical consequences

q(a) and r(a)
\text{are logical consequences}

material implication:
apply only to known constants
Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

- Retaining tractability III: ELP

Intro
Extending OWL with Rules
Rules inside OWL
Putting it all together
More rules than you ever need: SWRL

- Union of OWL DL with (binary) function-free Horn rules (with binary Datalog rules)
- undecidable
- no native tools available
- rather an overarching formalism

see http://www.w3.org/Submission/SWRL/
SWRL example (running example)

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

\text{ThaiCurry} \sqsubseteq \exists \text{contains}.\{\text{peanutOil}\}
\top \sqsubseteq \forall \text{orderedDish. Dish}

NutAllergic(x) \land 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)
SWRL example (running 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 (running example)**

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

Thailand ⊑ ∃contains. {peanutOil}

∀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,ys)  
Thailand(ys)  
Dish(ys)
SWRL example (running example)

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

\( \text{ThaiCurry} \subseteq \exists \text{contains.} \{ \text{peanutOil} \} \)
\( \top \subseteq \forall \text{orderedDish.Dish} \)

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

Conclusions:
\( \text{dislikes}(\text{sebastian},\text{peanutOil}) \)
\( \text{contains}(y_s,\text{peanutOil}) \)
\( \text{orderedDish}(\text{sebastian},y_s) \)
\( \text{ThaiCurry}(y_s) \)
\( \text{Dish}(y_s) \)
NutAllergic(sebastian)
NutProduct(peanutOil)
\exists orderedDish.ThaiCurry(sebastian)

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

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

Conclusions:
dislikes(sebastian,peanutOil)  
dislikes(sebastian,y_s)  
thaiCurry(y_s)  
Dish(y_s)  
contains(y_s,peanutOil)
SWRL example (running example)

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

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

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

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

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

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

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

Conclusion: \textbf{Unhappy(sebastian)}
Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

- Retaining tractability III: ELP

Intro
Extending OWL with Rules
Rules inside OWL
putting it all together
Retaining decidability I: DL-safety

- Reinterpret SWRL rules:
  Rules apply only to individuals which are explicitly given in the knowledge base.
  - Herbrand-style way of interpreting them

- OWL DL + DL-safe SWRL is decidable
- Native support e.g. by KAON2 and Pellet

DL-safe SWRL example

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

\text{ThaiCurry} \sqsubseteq \exists \text{contains}.\{\text{peanutOil}\}
\top \sqsubseteq \forall \text{orderedDish}.\text{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)

\text{Unhappy(sebastian)} \text{ cannot be concluded}
**DL-safe SWRL example**

NutAllergic(sebastian)

NutProduct(peanutOil)

∃orderedDish.ThaiCurry(sebastian)

ThaiCurry ⊆ ∃contains.{peanutOil}

∀orderedDish.Dish

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

Conclusions:

\[
\begin{align*}
\text{dislikes(sebastian,peanutOil)} \\
\text{orderedDish(sebastian,}y_s) \\
\text{ThaiCurry}(y_s) \\
\text{Dish}(y_s)
\end{align*}
\]
Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog
- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules
- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2
- Retaining tractability III: ELP

Intro
Extending OWL with Rules
Rules inside OWL
putting it all together
Retaining decidability II: DL Rules

- General idea:
  Find out which rules can be encoded in OWL (2 DL) anyway

- Man(x) ∧ hasBrother(x,y) ∧ hasChild(y,z) → Uncle(x)
  - Man ⊑ ∀hasBrother.∃hasChild.⊤ ⊑ Uncle

- ThaiCurry(x) → ∃contains.FishProduct(x)
  - ThaiCurry ⊑ ∃contains.FishProduct

- kills(x,x) → suicide(x)
  - ∃kills.Self ⊑ suicide

Note: with these two axioms,

\( suicide \) is basically the same as \( kills \)
DL Rules: more examples

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

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

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

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

\[ C(x) \land R(x,a) \land S(x,y) \land D(y) \land T(y,a) \rightarrow E(x) \]
- \[ C \cap \exists R.\{a\} \cap \exists S.(D \cap \exists T.\{a\}) \subseteq E \]

![Diagram of tree-shaped bodies with duplicating nominals being ok]
DL Rules: definition

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

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

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

- Tree-shaped bodies
- First argument of the conclusion is the root
- Complex classes are allowed in the rules
  - \( \text{Mouse}(x) \land \exists \text{hasNose}.\text{TrunkLike}(y) \rightarrow \text{smallerThan}(x,y) \)
  - \( \text{ThaiCurry}(x) \rightarrow \exists \text{contains.} \text{FishProduct}(x) \)
    
    Note: This allows to reason with unknowns (unlike Datalog)
  - Allowed class constructors depend on the chosen underlying description logic!
DL Rules: definition

Given a description logic $\mathcal{D}$, the language $\mathcal{D}$ Rules consists of:
- all axioms expressible in $\mathcal{D}$,
- plus all rules with
  - tree-shaped bodies, where
  - the first argument of the conclusion is the root, and
  - complex classes from $\mathcal{D}$ are allowed in the rules.
- <plus possibly some restrictions concerning e.g. the use of simple roles – depending on $\mathcal{D}$>
Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

- Retaining tractability III: ELP

Intro
Extending OWL with Rules
Rules inside OWL
putting it all together
The rules hidden in OWL 2: SROIQ Rules

- N2ExpTime complete

- In fact, SROIQ Rules can be translated into SROIQ i.e. they don't add expressivity.
  Translation is polynomial.

- SROIQ Rules are essentially helpful syntactic sugar for OWL 2.
SROIQ Rules example

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

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

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

!not a SROIQ Rule!
Each SROIQ Rule can be written ("linearised") such that
- the body-tree is linear,
- if the head is of the form $R(x,y)$, then $y$ is the leaf of the tree, and
- if the head is of the form $C(x)$, then the tree is only the root.

\[
\text{worksAt}(x,y) \land \text{University}(y) \land \text{supervises}(x,z) \land \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)
\]

\[
\exists \text{worksAt.University}(x) \land \text{supervises}(x,z) \land \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)
\]

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

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

- Motivation: OWL and Rules
- Preliminaries: Datalog

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

- The rules hidden in OWL 2: SROIQ Rules
- **Retaining tractability I: OWL 2 EL Rules**
- Retaining tractability II: DLP 2

- Retaining tractability III: ELP

- Extending OWL with Rules
- Rules inside OWL

- putting it all together
Retaining tractability I: OWL 2 EL Rules

- EL++ Rules are PTime complete

- EL++ Rules offer expressivity which is not readily available in EL++.
OWL 2 EL Rules: normal form

- Every EL++ Rule can be converted into a normal form, where
  - occurring classes in the rule body are either atomic or nominals,
  - all variables in a rule’s head occur also in its body, and
  - rule heads can only be of one of the forms $A(x)$, $\exists R.A(x)$, $R(x,y)$, where $A$ is an atomic class or a nominal or $\top$ or $\bot$.

- Translation is polynomial.

- $\exists \text{worksAt.University}(x) \land \text{supervises}(x,z) \land \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)$

- $\text{worksAt}(x,y) \land \text{University}(y) \land \text{supervises}(x,z) \land \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)$

- $\text{ThaiCurry}(x) \rightarrow \exists \text{contains.FishProduct}(x)$
OWL 2 EL Rules in a nutshell

Essentially, OWL 2 EL Rules is

- Binary Datalog with tree-shaped rule bodies,
- extended by
  - occurrence of nominals as atoms and
  - existential class expressions in the head.

- The existentials really make the difference.

- Arguably the better alternative to OWL 2 EL (aka EL++)?
  - (which is covered anyway)
Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
  
  Retaining tractability II: DLP 2

- Retaining tractability III: ELP

Intro
Extending OWL with Rules
Rules inside OWL
putting it all together
Retaining tractability II: DLP 2

- DLP 2 is
  - DLP (aka OWL 2 RL) extended with
  - DL rules, which use
    - left-hand-side class expressions in the bodies and
    - right-hand-side class expressions in the head.

- Polynomial transformation into 5-variable Horn rules.

- PTime.

- Quite a bit more expressive than DLP / OWL 2 RL ...
Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

- Retaining tractability III: ELP

Putting it all together
Retaining tractability III: ELP (aka putting it all together)

- ELP is
  - OWL 2 EL Rules +
  - a generalisation of DL-safety +
  - variable-restricted DL-safe Datalog +
  - role conjunctions (for simple roles).

- PTime complete.
- Contains OWL 2 EL and OWL 2 RL.
- Covers variable-restricted Datalog.
DL-safe variables

- A generalisation of DL-safety.
- DL-safe variables are special variables which bind only to named individuals (like in DL-safe rules).
- DL-safe variables can replace individuals in EL++ rules.

\[ C(x) \land R(x,x_s) \land S(x,y) \land D(y) \land T(y,x_s) \rightarrow E(x) \]

with \( x_s \) a safe variable is allowed, because

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

is an EL++ rule.

- Duplicating nominals is ok.
Variable-restricted DL-safe Datalog

- n-Datalog is Datalog, where the number of variables occurring in rules is globally bounded by $n$.

- complexity of n-Datalog is PTime (for fixed $n$)
  - (but exponential in $n$)

- in a sense, this is cheating.
- in another sense, this means that using a few DL-safe Datalog rules together with an EL++ rules knowledge base shouldn't really be a problem in terms of reasoning performance.
Role conjunctions

- orderedDish(x,y) ∧ dislikes(x,y) \rightarrow Unhappy(x)

- In fact, role conjunctions can also be added to OWL 2 DL without increase in complexity.

Retaining tractability III: ELP
(aka putting it all together)

\[ ELP_n \text{ is} \]
\[
\begin{align*}
\text{OWL 2 EL Rules generalised by DL-safe variables +} \\
\text{DL-safe Datalog rules with at most n variables +} \\
\text{role conjunctions (for simple roles).}
\end{align*}
\]

\[ \text{PTime complete (for fixed n).} \]
\[
\begin{align*}
\text{exponential in n} \\
\text{Contains OWL 2 EL and OWL 2 RL.} \\
\text{Covers all Datalog rules with at most n variables. (!)}
\end{align*}
\]
ELP example

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

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

[okay]
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)

[okay – role conjunction]

not an EL++ rule
ELP example

- dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)
  as SROIQ rule translates to

  \[
  \text{Dish} \equiv \exists \text{dish.Self} \\
  \text{dislikes} \circ \text{contains} \sim \circ \text{dish} \sqsubseteq \text{dislikes}
  \]

  but we don't have inverse roles in ELP!

- solution: make z a DL-safe variable:

  dislikes(x,!z) ∧ Dish(y) ∧ contains(y,!z) → dislikes(x,y)

  this is fine 😊
DL-safe 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)
contains(y,s,peanutOil)
orderedDish(sebastian,y,s)
dislikes(sebastian,y,s)
ThaiCurry(y_s)
Dish(y_s)
ELP example

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

\( \text{ThaiCurry} \subseteq \exists \text{contains.}{\text{peanutOil}} \)
\( \top \subseteq \forall \text{orderedDish.} \text{Dish} \)

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

Conclusion: Unhappy(sebastian)
ELP Reasoner ELLY

- Implementation currently being finalised.
- Based on IRIS Datalog reasoner.
- In cooperation with STI Innsbruck (Barry Bishop, Daniel Winkler, Gulay Unel).

Legend:
R ... Rule
H ... Head
B ... Body
A ... Atom
L ... Literal
The Big Picture

- ELP

- OWL 2 EL Rules

- OWL 2 EL

- OWL 2 = SROIQ Rules

- >ExpTime

- tractable
Closed World and ELP

- There's an extension of ELP using (non-monotonic) closed-world reasoning – based on a well-founded semantics for hybrid MKNF knowledge bases.

The Big Picture II

ELP

OWL 2 EL Rules

OWL 2
= SROIQ Rules

>ExpTime

tractable
data-tractable

hybrid ELP (local closed world)


http://www.w3.org/Submission/SWRL/


References


Thanks!

http://semantic-web-grundlagen.de/wiki/ESWC09_Tutorial