Introduction to DLs, OWL, RacerPro

Semantic Web
Abox Query Answering
RacerPro  DLs  Ontologies
OWL  Logical Reasoning
Logical Models

Entailment

„Tag Cloud“
Logic-Based Knowledge Representation

- DLs: family of logics for knowledge representation (KR)
  - foundation for ontologies & Semantic Web
  - ... but what is logic-based KR?
- Logic
  - formal syntax and semantics
  - notion of entailed / logically implied formulas: $\models$
  - mechanical reasoning (inference / proof system)
- Basic idea of logic-based KR
  - knowledge base (KB) = set of formulas (axioms)
  - represents knowledge of some „agent“
  - agent uses proof system to derive conclusions from the KB which are meaningful in the environment
Purpose of (Logical) Models

- Replace "real-world reasoning" in some DOD with computational operations performed on the representations (|=)
- "real-world" reasoning may be impossible, too dangerous, too expensive, too complicated, ...
- Representation involves abstraction
  - conceptualization!
- Models have a purpose
  - conceptualization depends on purpose and DOD

Relevant aspects should be isomorphic with real world
KBs, Logical Models, Entailment

KB = Set of Formulas / Axioms
- All individuals are female or male
- Mothers are parents and woman
- A parent has a child
- Woman are female persons
- Betty is a mother

- Implicit information, things can be left unsaid (e.g., that Betty is a person)
- What holds in all models? Entailment ⊨
- The more axioms, the less models
- The less models, the more entailed formulas, the more implicit / entailed information!
- Chaos = absence of structure
KBs, Logical Models, Entailment (2)

- Some KBs have only infinite models; countable infinite models suffice
- For each first-order logic model, there is a bigger one (Löwenheim-Skolem)
- Each KB has an infinite number of models, or is contradictory
- A contradictory KB has no models (important inference problem!)
- From a contradictory KB, everything follows

\[
\begin{align*}
\forall x. (\text{female}(x) \lor \text{male}(x)) \\
\forall x. (\text{mother}(x) \rightarrow \text{parent}(x) \land \text{woman}(x)) \\
\forall x. (\text{parent}(x) \leftrightarrow \exists y. (\text{has\_child}(x, y))) \\
\forall x. (\text{woman}(x) \leftrightarrow \text{female}(x) \land \text{person}(x)) \\
\text{mother(betty)}
\end{align*}
\]
$\forall x, y, z. \text{has\_ancestor}(x, y) \land \text{has\_ancestor}(y, z)$

$\rightarrow \text{has\_ancestor}(x, z)$

$\forall x. \neg\text{has\_ancestor}(x, x)$

$\forall x. \exists y. \text{has\_ancestor}(x, y)$

$\text{betty}$
Why Description Logics?

- Formal
  - suitable as **ontology** languages (Gruber definition)
  - foundation for the Semantic Web
- Well-understood
  - Semantics, complexity, implementation techniques
- Decidable
  - unlike FOPL
- Relatively mature set of tools available
  - Reasoners: Fact++, Pellet, RacerPro
  - Editors: Protege, Swoop, RacerPorter, ...
  - Visualizers: OWLViz, OntoTrack, ...
Description Logics: Basic Notions

- Based on first order-logic
  - but variable-free and decidable
  - concept languages, class-based KR

- Central notions:
  - Concept (OWL: Class)
    - atomic or complex (concept term)
  - Role (OWL: Property, RDF: Predicate)
  - Individual
  - Container data structures:
    - TBox: Set of terminological axioms
    - ABox: Set of assertional axioms
Architecture of a DL System

TELL

betty : mother
mother ⊑ woman

ASK

betty : woman


DL System

|= betty : woman

TBox

mother ⊑ woman

ABox

betty : mother
Description Logics: Concepts

- Represent „classes“ = sets of individuals
  - atomic concepts: basic vocabulary, e.g. person
  - complex concepts: e.g. person \( \sqcap \) female

- Semantics via interpretation \( \mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}) \)
  - interpretation of a concept = set of individuals in \( \Delta^\mathcal{I} \)
  - function \( \cdot^\mathcal{I} \) maps concept \( C \) to subset of \( \Delta^\mathcal{I} \)

\[
person^\mathcal{I} \subseteq \Delta^\mathcal{I} \quad \cdot^\mathcal{I} : \mathcal{N}_C \rightarrow 2^{\Delta^\mathcal{I}}
\]

- Top and bottom: \( \top^\mathcal{I} = \Delta^\mathcal{I} \quad \bot^\mathcal{I} = \emptyset \)

- Concept constructors, e.g. conjunction
  - constraint on interpretation of complex concepts

\[
(person \sqcap female)^\mathcal{I} = person^\mathcal{I} \cap female^\mathcal{I}
\]
Illustration of Concept Semantics

Syntax

\(\text{person} \sqcap \text{female}\)

Semantics

\(\text{person}^\mathcal{I}\)

\((\text{person} \sqcap \text{female})^\mathcal{I}\)

\(\text{female}^\mathcal{I}\)

\(\text{owl:Thing}\)
Description Logics : Roles

- Represent relationships = sets of (binary) tuples
  - atomic roles : basic vocabulary, e.g. $\text{has\_child}$
  - complex roles: e.g. $\text{has\_child}^{-1}$

- Semantics via interpretation $(\Delta^\mathcal{I}, .^\mathcal{I})$
  - interpretation of a role = set of tuples from $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
  - function $.^\mathcal{I}$ maps $R$ to subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$

  $\text{has\_child}^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$

  $^\mathcal{I} : \mathcal{N}_R \mapsto 2^{\Delta^\mathcal{I} \times \Delta^\mathcal{I}}$

- Role constructors, e.g. inverse role
  - constraint on interpretation of complex roles

  $(\text{has\_child}^{-1})^\mathcal{I} = (\text{has\_child}^\mathcal{I})^{-1}$
Illustration of Role Semantics

Syntax

Semantics

\( \subseteq (\Delta^I \times \Delta^I) \)
Concept Constructors: Conjunction

- DL syntax
  \( person \cap female \)

- KRSS / Racer
  (and person female)

- OWL RDF
  
  \[
  \text{<owl:Class>}
  \text{<owl:intersectionOf}
  \text{rdf:parseType="Collection">}
  \text{<owl:Class rdf:ID="Person"/>}
  \text{<owl:Class rdf:ID="Female"/>}
  \text{</owl:intersectionOf>}
  \text{</owl:Class>}
  \]
Concept Constructors: Disjunction

- **DL syntax**
  
  \( \text{male} \sqcup \text{female} \)

- **KRSS / Racer**
  
  (or male female)

- **OWL RDF**
  
  ```xml
  <owl:Class>
    <owl:unionOf>
      rdf:parseType="Collection">
        <owl:Class rdf:about="#Male"/>
        <owl:Class rdf:about="#Female"/>
      </owl:unionOf>
    </owl:Class>
  </owl:Class>
  ```
Concept Constructors: Negation

- DL syntax
  \(-female\)
- KRSS / Racer
  (not female)
- OWL RDF

\[
(\neg female)^I = \Delta^I \setminus female^I
\]

```xml
<owl:Class>
  <owl:complementOf
    <owl:Class
      rdf:about="#Female"/>
  </owl:complementOf>
</owl:Class>
```
Concept Constructors : Existentials

- DL syntax
  \[ (\exists has\_mother\_woman)^I = \{ i \in \Delta^I | \exists j : (i, j) \in has\_mother^I, j \in woman^I \} \]

- KRSS / Racer
  (some has-mother woman)

- OWL RDF
  ```xml
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:about="#hasMother"/>
    </owl:onProperty>
    <owl:someValuesFrom>
      <owl:Class rdf:about="#Woman"/>
    </owl:someValuesFrom>
  </owl:Restriction>
  ```
Concept Constructors: Universals

- DL syntax

\[ (\forall \text{has\_mother\_woman})^\mathcal{I} = \{ i \in \Delta^\mathcal{I} \mid \forall j : (i, j) \in \text{has\_mother}^\mathcal{I} \rightarrow j \in \text{woman}^\mathcal{I} \} \]

- KRSS / Racer

(all has-mother woman)

- OWL RDF

```xml
<owl:Restriction>
  <owl:onProperty>
    <owl:ObjectProperty
      rdf:about="#hasMother"/>
  </owl:onProperty>
  <owl:allValuesFrom>
    <owl:Class rdf:about="#Woman"/>
  </owl:allValuesFrom>
</owl:Restriction>
```
Constructors: Number Restrictions

- DL syntax
  \[(\leq_1 \text{has\_mother})^I = \{ i \in \Delta^I \mid \#\{(i, j) \mid (i, j) \in \text{has\_mother}^I\} \leq 1\}\]

- KRSS / Racer
  (at-most 1 has-mother)

- OWL RDF

```xml
<owl:Restriction>
  <owl:onProperty>
    <owl:ObjectProperty>
      rdf:about="#hasMother"/>
    </owl:ObjectProperty>
  </owl:onProperty>
  <owl:maxCardinality
    rdf:datatype="..."#nonNegativeInteger">
    1
  </owl:maxCardinality>
</owl:Restriction>
```
Constructors : Number Restrictions

- **DL syntax**
  \[
  (\leq_1 has\_mother\_woman)^I = \\
  \{ i \in \Delta^I \mid \#\{ (i, j) \mid (i, j) \in has\_mother^I, j \in woman^I \} \leq 1 \}
  \]

- **KRSS / Racer**
  (at-most 1 has-mother woman)

- **OWL RDF**
  
  ```xml
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty
        rdf:about="#hasMother"/>
    </owl:onProperty>
    <owl:maxCardinality
      rdf:datatype="#nonNegativeInteger">
      1
    </owl:maxCardinality>
    <owl2:onClass rdf:resource="#Woman"/>
  </owl:Restriction>
  ```
Constructors : Number Restrictions

- DL syntax

\[(\geq 2 \text{ has\_child})^I = \{ \text{i} \in \Delta^I \mid \# \{(i, j) \mid (i, j) \in \text{has\_child}^I \} \geq 2 \}\]

- KRSS / Racer

(at-least 2 has-child)

- OWL RDF

```xml
<owl:Restriction>
  <owl:onProperty>
    <owl:ObjectProperty
      rdf:about="#hasChild"/>
  </owl:onProperty>
  <owl:minCardinality
    rdf:datatype="...#nonNegativeInteger">
    2
  </owl:minCardinality>
</owl:Restriction>
```
Constructors : Number Restrictions

- **DL syntax**
  \[
  \geq_2 \text{has\_child\_female} = \{ i \in \Delta^\mathcal{I} \mid \#\{(i, j) \mid (i, j) \in \text{has\_child}^\mathcal{I}, j \in \text{female}^\mathcal{I}\} \geq 2\}
  \]

- **KRSS / Racer**
  (at-least 2 has-child female)

- **OWL RDF**
  
  ```xml
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty>
        rdf:about="#hasChild"/
      </owl:ObjectProperty>
      <owl:minCardinality
        rdf:datatype="#nonNegativeInteger">2
      </owl:minCardinality>
    </owl:onProperty>
    <owl2:onClass rdf:resource="#Female"/>
  </owl:Restriction>
  ```
Inference Problems for Concepts

- Concept Satisfiability (Core Problem!)
  - exists some \((\Delta^\mathcal{I}, \cdot^\mathcal{I})\) such that \(C^\mathcal{I} \neq \emptyset\)?
  - then, \((\Delta^\mathcal{I}, \cdot^\mathcal{I}) \models C\) and \((\Delta^\mathcal{I}, \cdot^\mathcal{I})\) is a model of \(C\)

- Concept Subsumption („Inheritance“)
  - does \(C^\mathcal{I} \subseteq D^\mathcal{I}\) hold in all interpretations?
  - then, \(D\) subsumes \(C\) (subsumer / subsumee)
  - \(\models C \sqsubseteq D\) iff \(C \cap \neg D\) unsatisfiable

• Equivalence: \(\models C \sqsubseteq D, \models D \sqsubseteq C\)

• Disjointness
  - holds \(C^\mathcal{I} \cap D^\mathcal{I} = \emptyset\) in all interpretations?
  - iff \(C \cap D\) unsatisfiable
Description Logics : TBox Axioms

- Constrain interpretations of (atomic) concepts
  - enforce subset relationships
    \[ \mathcal{I} \models mother \sqsubseteq parent \]
    \[ \text{iff} \]
    \[ mother^\mathcal{I} \subseteq parent^\mathcal{I} \]
  - enforce equivalences
    ("definitions")
    \[ \mathcal{I} \models parent \equiv \text{person} \sqcap \exists \text{has\_child}. \top \]
    \[ \text{iff} \]
    \[ parent^\mathcal{I} = \text{person}^\mathcal{I} \sqcap (\exists \text{has\_child}. \top)^\mathcal{I} \]
  - Nowadays, arbitrary concepts in axioms (GCIs)
Description Logics: TBox Axioms (2)

- DL Syntax
  \[ \text{mother} \sqsubseteq \text{parent} \]

- KRSS / Racer
  (implies mother parent)
  (define-primitive-concept
   mother parent)

- OWL
  
  `<owl:Class rdf:about="Mother">
  <rdfs:subClassOf>
    <owl:Class rdf:about="Parent"/>
  </rdfs:subClassOf>
  </owl:Class>

- DL Syntax
  \[ \text{parent} \equiv \text{person} \sqcap \exists \text{has\_child}. \top \]

- KRSS / Racer
  (equivalent parent
   (and person ... ))
  (define-concept parent
   (and person ... ))

- OWL
  
  `<owl:Class rdf:about="Parent">
  <owl:equivalentClass>
    <owl:Class rdf:about="Person"/>
    <owl:Class>
      <owl:intersectionOf
        ...
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  </owl:Class>`
DLs as First Order Logic

- Concepts: FOPL formulas with one free variable
  \[ \text{person}(x) \land \exists y. \text{has\_child}(x, y) \]

- Roles: binary atoms with two free variables
  \[ \text{has\_child}(x, y) \]

- Individuals: constants
  \[ \text{betty} \]

- Axioms
  \[ \forall x. (\text{parent}(x) \leftrightarrow \text{person}(x) \land \exists y. \text{has\_child}(x, y)) \]
  \[ \forall x. (\text{mother}(x) \rightarrow \text{parent}(x)) \]
  \[ \forall x, y. (\text{has\_child}(x, y) \leftrightarrow \text{has\_child}(y, x)) \]
  \[ \forall x, y, z. (\text{has\_descendant}(x, y) \land \text{has\_descendant}(y, z) \rightarrow \text{has\_descendant}(x, z)) \]
Inference Problems for TBoxes

- Concept satisfiability (disjointness, subsumption, equivalence) w.r.t. a Tbox, e.g.
  - parent ∩ ¬mother unsat. w.r.t. TBox
  - woman ∩ ∃has_child.female ⊆ parent due to TBox

- Reasoning example:
  woman ∩ ∃has_child.female ⊆ parent iff
  woman ∩ ∃has_child.female ∩ ¬parent unsat. iff
  woman ∩ ∃has_child.female ∩ ¬(person ∩ ∃has_child.⊥) iff
  person ∩ ⋯ ∩ ∃has_child.female ∩ ¬person unsat. AND
  person ∩ ⋯ ∩ ∃has_child.female ∩ ∀has_child.⊥ unsat.

- ... only that simple for simple (unfoldable) TBoxes
Inference Problems for TBoxes (2)

- **TBox coherence check**
  - unsatisfiable concept names or roles other than $\bot$? e.g.
    \[ C \sqsubseteq \neg C, \ldots \]

- **TBox satisfiability**
  - all concepts names unsatisfiable? e.g.
    \[ C \equiv \neg C, \ldots \]

- **Taxonomy computation**
  - compute most specific subsumers and most general subsumees for names

\[
\begin{align*}
\text{woman} & \equiv \text{person} \sqcap \text{female} \\
\text{parent} & \equiv \text{person} \sqcap \exists \text{has_child}. \top \\
\text{mother\_w\_daughter} & \equiv \text{woman} \sqcap \exists \text{has_child}. \text{female}
\end{align*}
\]
More Tbox Axioms: Role Declarations

- Sub / super roles $\text{has\_mother} \sqsubseteq \text{has\_parent}$
- Transitive roles $\text{transitive} (\text{has\_descendant})$
  - no number restrictions f. trans. Roles (or roles with trans. subroles) allowed!
- Functional roles $\text{functional} (\text{has\_mother})$
- Being inverses $\text{has\_parent} \equiv \text{has\_child}^{-1}$
- Domain & range restrictions
  - $\text{domain} (\text{has\_mother}) = \text{person}$
    \[ \exists \text{has\_mother}. \top \sqsubseteq \text{person} \]
  - $\text{range} (\text{has\_mother}) = \text{mother}$
    \[ \top \sqsubseteq \forall \text{has\_mother}. \text{mother} \]
Syntax of Role Declarations

(define-primitive-role
  has-descendant
  :transitive t)

(define-primitive-role
  has-child
  :parent has-descendant)

(define-primitive-role
  has-parent
  :inverse has-child)

(define-primitive-role
  has-mother
  :parent has-parent
  :domain person
  :range mother
  :feature t)
TBox Patterns : Covering Axioms

- **Disjointness** $\text{plant} \sqcap \neg \text{animal}(= \text{animal} \sqcap \neg \text{plant})$
  - OWL axiom $<\text{owl:disjointWith}>$
- **Covering axioms** $\text{living thing} \equiv \text{plant} \sqcup \text{animal}$
  - OWL axiom $<\text{owl:disjointUnion}>$
- **Global axioms** $\top \sqsubseteq \text{plant} \sqcup \text{animal}$
- **Global consistency condition** $\text{plant} \sqcap \text{animal} \equiv \bot$
Implicit Subsumptions

• In an axiom $D \subseteq X$
  
  - $D$ is **sufficient** for $X$ ("given $D$, $X$ follows")
  - $X$ is **necessary** for $D$ ("without $X$, $D$ cannot hold")
  - Since $\neg X \subseteq \neg D$, also $\neg X$ is sufficient for $\neg D$
  - no sufficient conditions for $D$!
  - But: $D \equiv X \cap D^*$ for some fresh $D^*$

• However, $C \subseteq D$ can never hold, since
  
  $C \cap \neg D$ unsat. iff $C \cap \neg (X \cap D^*)$ unsat. iff
  
  $(C \cap \neg X)$ unsat. AND $(C \cap \neg D^*)$ unsat.
  
  the latter can never happen, since $D^*$ was fresh

• Thus: no **implicit** subsumption without proper
  
  sufficient conditions for subsumer ($D$)
Implicit Subsumptions (2)

color

red XOR green XOR blue
red ⊆ ¬ green
red ⊆ ¬ blue
green ⊆ ¬ blue
yellow ⊆ ¬ color
yellow ⊆ ¬ green
yellow ⊆ ¬ red

disjoint

?
Implicit Subsumptions (3)

\[ \text{color} \equiv \text{red} \sqcup \text{green} \sqcup \text{blue} \]
\[ \text{red} \sqsubseteq \neg \text{green} \]
\[ \text{red} \sqsubseteq \neg \text{blue} \]
\[ \text{green} \sqsubseteq \neg \text{blue} \]
\[ \text{yellow} \sqsubseteq \neg \text{color} \]
\[ \text{yellow} \sqsubseteq \neg \text{green} \]
\[ \text{yellow} \sqsubseteq \neg \text{red} \]
Implicit Subsumptions (4)

\[ \text{color} \sqsubseteq \text{red} \sqcup \text{green} \sqcup \text{blue} \]
\[ \text{red} \sqsubseteq \neg \text{green} \]
\[ \text{red} \sqsubseteq \neg \text{blue} \]
\[ \text{green} \sqsubseteq \neg \text{blue} \]
\[ \text{yellow} \sqsubseteq \text{color} \]
\[ \text{yellow} \sqsubseteq \neg \text{green} \]
\[ \text{yellow} \sqsubseteq \neg \text{red} \]
Implicit Subsumptions (5)

\[
\begin{align*}
\text{color} & \Rightarrow \text{red} \sqcup \text{green} \sqcup \text{blue} \\
\text{red} & \sqsubseteq \neg \text{green} \\
\text{red} & \sqsubseteq \neg \text{blue} \\
\text{green} & \sqsubseteq \neg \text{blue} \\
\text{yellow} & \sqsubseteq \neg \text{color} \\
\text{yellow} & \sqsubseteq \neg \text{green} \\
\text{yellow} & \sqsubseteq \neg \text{red}
\end{align*}
\]
Implicit Subsumption Relationships

\[ \text{color} \equiv \text{red} \sqcup \text{green} \sqcup \text{blue} \]
\[ \text{red} \sqsubseteq \neg \text{green} \]
\[ \text{red} \sqsubseteq \neg \text{blue} \]
\[ \text{green} \sqsubseteq \neg \text{blue} \]
\[ \text{yellow} \sqsubseteq \neg \text{red} \]
\[ \text{yellow} \sqsubseteq \neg \text{green} \]
\[ \text{disjoint} \]

\[ \text{yellow} \sqcap \neg \text{blue} \models \]
\[ \text{yellow} \sqcap \neg \text{green} \sqcap \neg \text{red} \sqcap \neg \text{color} \sqcap \neg \text{blue} \models \]
\[ \text{yellow} \sqcap \neg \text{green} \sqcap \neg \text{red} \sqcap \neg \text{blue} \sqcap \neg \text{red} \sqcap \text{blue} \models \]
\[ \text{yellow} \sqcap \neg \text{green} \sqcap \neg \text{red} \sqcap \neg \text{blue} \sqcap \text{blue} \models \bot \]
# DL Naming Schema

<table>
<thead>
<tr>
<th>DL</th>
<th>Expressive Means</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ALC$</td>
<td>$\sqcap, \sqcup, \exists R.C, \forall R.C$</td>
</tr>
<tr>
<td>$S(ALC_{R^+})$</td>
<td>$ALC + R^+$ for transitively closed roles</td>
</tr>
<tr>
<td>$ALCI$</td>
<td>$ALC + I$ for inverse roles</td>
</tr>
<tr>
<td>$ALCH$</td>
<td>$ALC + \mathcal{H}$ for role hierarchies</td>
</tr>
<tr>
<td>$ALCN$</td>
<td>$ALC + \mathcal{N}$ for number restrictions</td>
</tr>
<tr>
<td>$ALCQ$</td>
<td>$ALC + Q$ for qualified number restrictions</td>
</tr>
<tr>
<td>$ALCO$</td>
<td>$ALC + O$ for nominals</td>
</tr>
<tr>
<td>$OWL = SHOIN(D^-)$</td>
<td>$D^-$ for datatypes</td>
</tr>
<tr>
<td>$RACERPRO = SHIQ(D^-)$</td>
<td>$D^-$ for concrete domains</td>
</tr>
<tr>
<td>$OWL2 = SROIQ(D^+)$</td>
<td>$\mathcal{R}$ for complex role inclusions</td>
</tr>
</tbody>
</table>
Individuals and Relationships: ABox

- Abox = set of ABox assertions (axioms)
- Instance and role assertions (plus same-as, different-from, ...)

\{ betty : person, (betty, charles) : has\_child \}

(instance betty person)
(related betty charles has-child)

\(<\text{Person rdf:ID="betty"}>
\quad <\text{hasChild rdf:resource="#charles"}/>
\quad </\text{Person}>\)

- $\mathcal{I}$ maps individuals to elements in $\Delta^\mathcal{I}$

\[
\begin{align*}
\mathcal{I} &\models \text{betty : person iff } \text{betty}^\mathcal{I} \in \text{person}^\mathcal{I} \\
\mathcal{I} &\models (\text{betty, charles} : \text{has\_child} \iff (\text{betty}^\mathcal{I}, \text{charles}^\mathcal{I}) \in \text{has\_child}^\mathcal{I})
\end{align*}
\]
ABox Inference Services

- Abox satisfiability (w.r.t. a possibly empty TBox)
  - does the Abox have a model?
    \{betty : \neg parent, betty : person, (betty, charles) : has\_child\}

- Individual realization
  - compute the (most specific) concept names an individual is an instance of, e.g. in
    \{betty : person, (betty, charles) : has\_child\}
  it is realized that betty is an instance of parent

- Instance checking: is betty and instance of parent?

- Role filler checking: is charles a filler (successor) of the has\_child role of betty?
Abox Inference Services (2)

• Abox retrieval services
  
  - Instance retrieval
    \[ \text{concept}\_\text{instances}(\text{parent}) = \{\text{betty}\} \]
    \[(\text{concept}\_\text{instances}\ \text{parent}) \rightarrow (\text{betty})\]
  
  - Role filler retrieval
    \[ \text{role}\_\text{fillers}(\text{betty}, \text{has}\_\text{child}) = \{\text{charles}\} \]
    \[(\text{individual}\_\text{fillers}\ \text{betty}\ \text{has}\_\text{child}) \rightarrow (\text{charles})\]
  
  - ... and some more

• Recent research focus: ABox query answering
  
  - RDF QLs: SPARQL, RQL, ...
  
  - „true“ DL QLs: nRQL, OWLQL, ...
Assuming all blocks are red or green - is there a green block on the table which is next to a red one?

{ \text{block \sqsubseteq red} \sqcup \text{green}, \text{next-to} \equiv \text{next-to}^{-1} \}

{ t : \text{table}, lb : \text{green} \sqcap \text{block}, rb : \text{red} \sqcap \text{block},
  mb : \text{block}, ob : \text{green} \sqcap \text{block},
  (lb, t) : \text{on-table}, (ml, t) : \text{on-table}, (rb, t) : \text{on-table},
  (gb, ob) : \text{next-to}, (ob, rb) : \text{next-to} } 

Ask for instances of the concept

\text{table} \sqcap

\exists \text{on-table}^{-1} \cdot (\text{block} \sqcap \text{green} \sqcap

\exists \text{next-to}. (\text{red} \sqcap \text{block}))
There are two possibilities. If the middle block is red, then the green left block is next to a red one. But...

Model 1

\[ \exists_{on_table}^{-1}. (block \cap green \cap \exists_{next_to}. (red \cap block)) \]
... if the middle block is green, then it is also next to the right Block, which is red. So, yes, there ALWAYS EXISTS such a block on the table!

Model 1

Model 2

∃on_table⁻¹. (block ∩ green ∩ ∃next_to.(red ∩ block))
However:

\[
\text{concept\_instances}( \text{table} \cap \\
\exists on\_table^{-1}. (\text{block} \cap \text{green} \cap \\
\exists next\_to.(\text{red} \cap \text{block}))) = \{t\}
\]

\[
\text{concept\_instances}( \text{block} \cap \text{green} \cap \\
\exists next\_to.(\text{red} \cap \text{block}))) = \{\}
\]

- Unlike SQL, instance retr. queries can cope with
  - incomplete information (case analysis)
  - have to consider ALL models, not only one (rel.DB)
  - only the existence of such a block is entailed
Full Conjunctive Queries

[Diagram]

\[ \text{ans}(x) \leftarrow \text{table}(x), \text{on\_table}(y, x), \]
\[ \text{block}(y), \text{green}(y), \]
\[ \text{next\_to}(y, z), \text{red}(z), \text{block}(z). \]

Answer: \( x = t \)

- However, no answer for head \( \text{ans}(y) \leftarrow \ldots \)
  - distinguished variables in head: binding must hold in ALL models ("certain answer")
  - other variables: treated as existentially quantified
Grounded Conjunctive Queries

\[ ans(x) \leftarrow \text{table}(x), \text{on\_table}(y, x), \]
\[ \text{block}(y), \text{green}(y), \]
\[ \text{next\_to}(y, z), \text{red}(z), \text{block}(z). \]

Gives no answer in nRQL!

- In grounded conjunctive queries
  - ALL variables are distinguished; a binding is only established iff it holds in ALL models
  - grounding: subst. variables $\leftrightarrow$ entailed assertions
Grounded CQs vs. Full CQs

\( \text{ans}(x) \leftarrow (table \cap \exists \text{on_table}^{-1}.\ldots)(x) \)

This grounded CQ can be used instead of the full CQ (a simple instance retrieval query)

- This „rolling up“ into nested \( \exists R.\exists S.\ldots \) works only for non-cyclic queries
  - note that variables may introduce coreferences
  - no automatic rolling up in nRQL
Further Differences with Databases

• Open World Semantics

\[ \text{ans}(x) \leftarrow (\forall \text{on\_table}. (\text{block} \cap \text{green}))(x) \]
\[ \text{ans}(x) \leftarrow (\leq_2 \text{on\_table})(x) \]

give no answers

• the model / world is not closed - models with additional red blocks or even non-blocks exist, but can be excluded: \( t : \leq_2 \text{on\_table} \)
• the two blocks are the only ones \( \rightarrow \) all are green
• DB would conclude all blocks are green (CWA/NAF)
Even GCQs are not so easy...

**TBox:**
- functional($f_1$)
- transitive($r_1$)
- functional($g_1$)
- $g_1 \sqsubseteq f_1$
- $g_1 \sqsubseteq r_1$

**ABox:**

- $\exists g_1. \top$
- $\exists g_1. \top$

- $g_1, f_1$
- $g_1, f_1$
- $g_1, f_1, r_1$
- $g_1, f_1, r_1$
- $g_1, f_1, r_1$

- $r_1$

- $\text{ans}(x, y) \leftarrow r_1(x, y)$

- Yes!

- expand existentials
- g1 has f1 as parent role
- merge f1 successors
- f1 has r1 as superrole
- r1 is transitive!
NAF Negation and Projection

- Blocks for which we can prove they are not green:
  \[ \text{ans}(x) \leftarrow \text{block}(x), (\neg \text{green})(x) \Rightarrow \{d\} \]

- Blocks for which we cannot prove they are green:
  \[ \text{ans}(x) \leftarrow \text{block}(x), \neg \text{green}(x) \Rightarrow \{d, e\} \]

- Tables f.w.w.c. prove they have only green blocks:
  \[ \text{ans}(x) \leftarrow \text{table}(x), (\forall \text{on_table}^{-1}.(\text{block} \cap \text{green}))(x) \Rightarrow \{\} \]

- f.w.w. CANNOT prove they have NON-green blocks:
  \[
  \begin{align*}
  \text{ans}_1(x) & \leftarrow \text{table}(x), \text{on_table}(y, x), \neg \text{green}(x) \\
  \text{ans}(x) & \leftarrow \text{table}(x), \neg \text{ans}(x) \Rightarrow \{t1\}
  \end{align*}
  \]
NAF Negation and Projection (2)

Tables for which we CANNOT prove the have NON-green blocks (all known blocks on table are green):

(retrieve (?x)
  (and (?x table)
    (neg (project-to (?x)
      (and (?y ?x on-table)
        (neg (?y green)))))))))

nRQL is the only and first practical DL QL which can process that kind of queries; new project-to
Illustration of nRQL Semantics

\[(\text{retrieve } (?x))\]
\[(\text{and } (?x \text{ table})\]
\[(\text{neg } (\text{project-to } (?x))\]
\[(\text{and } (?y ?x \text{ on-table})\]
\[(\text{neg } (?y \text{ green})))))\]
\[(((?X \text{ T1})))\]
Illustration of nRQL Semantics (2)

\[(\text{retrieve } (?x) \, \text{and } (?x \text{ table}) \, \text{and } (\neg (\text{project-to } (?x) \, \text{and } (\neg (?y \, ?x \text{ on-table}) \, \neg (?y \, \text{green})))))))\]
Illustration of nRQL Semantics (3)

\[ \text{green} \equiv \neg \text{red} \]

\[
\begin{align*}
\text{t1} & : a \quad b \\
\text{t2} & : c \quad d \quad e
\end{align*}
\]

(retrieve (?x) 
(and (?x table) 
(neg (project-to (?x) 
(and (?y ?x on-table) 
(neg (?y green)))))
)

(((?X T1))))
Illustration of nRQL Semantics (4)

\[\text{green} \equiv \neg \text{red}\]

\[\begin{align*}
\text{t1} & : \{a, b\} \\
\text{t2} & : \{c, d, e\}
\end{align*}\]

\[
\text{(retrieve (?x)} \\
\text{(and (?x table)} \\
\text{\quad (neg (project-to (?x)} \\
\text{\quad \quad (and (?y ?x on-table)} \\
\text{\quad \quad \quad (neg (?y green))))))}
\]

\[\text{(((?X T1)))}\]
Illustration of nRQL Semantics (5)

\(\text{green} \equiv \neg \text{red}\)

\((\text{retrieve} \ (?x)\)

\((\text{and} \ (?x \ \text{table})\)

\((\neg \ (\text{project-to} \ (?x)\)

\((\text{and} \ (?y \ ?x \ \text{on-table})\)

\((\neg \ (?y \ \text{green})\)))))

\(((?X \ T1)))\)
Ontologies & Semantic Web

• Ontologies
  - formal description of a domain of discourse (DOD)
  - „An ontology is an explicit and formal specification of a (shared) conceptualization“
    • formal: prerequisite for computerized reasoning
    • conceptualization: classes and relationship, abstraction of DOD (e.g., parent, woman, ...)
    • shared: common understanding of terms
      - common base terms
      - shared conceptual notions (e.g., what constitutes a parent)
      - make these notions explicit in a formal description language
Ontologies & Semantic Web (2)

Semantic Web

• today: unstructured HTML documents

• tomorrow: explicit content descriptions ("meta data") for web resources
  - "ontologies for the web"

• Smarter search for pages, services, ...
  - e.g., "recognize" companies with DL professors!

• Can deal with semantic heterogeneity
  - e.g., $\text{semWebCompany} \leftrightarrow \text{DLCompany}$

Racer Systems was founded in 2004 by Volker Haarslev, Ralf Müller, Kay Hidde, and Michael Wessel.

$\models \text{company}(\text{Racer Systems})$

$\models \text{person}(\text{V. Haarslev})$

$\models \text{DL}\_\text{professor}(\text{V. Haarslev})$

$\text{name} \text{Racer Systems}$

$\text{founded} 2004$

$\text{founder}$

$\text{professor}$

$\text{name} \text{Volker Haarslev}$

$\text{specialField} \text{DL}$

$\text{specialField}$

$\text{professor}$

$\ldots$

$\text{semwebCompany}$
RDF, RDF Schema, SPARQL

- **Graph data model**
  - Edges = (subject, predicate, object) triples
  - Nodes (subject, object): URIs, literals (e.g., strings)
  - Triples "annotate" Web resources → "meta data" for the web
  - RDF vocabulary = RDF predicates in a namespace

- **Why not XML?**
  - XML: trees only
  - RDF (meta) data representation is more canonical (XML too semi-structured, no attribute vs. child element problem → retrieval prob. in XQuery)
  - Shallow reasoning in RDFS(++)
RDF, RDF Schema, SPARQL (2)

- RDF XML (other syntaxes exist)

```xml
<rdf:Description rdf:ID="#emp123">
  <rdf:type rdf:resource="&uni;Employee"/>
  <uni:name>M. Wessel</uni:name>
  <uni:teaches rdf:resource="#course56"/>
</rdf:Description>

<uni:Course rdf:ID="course56">
  <uni:homepage rdf:resource="www...." />
  <uni:participant> ... </uni:participant>
</uni:Course>

select ?x ?y where
{ ?x rdf:type uni:Employee .
  ?x uni:teaches
  ?y }
```
RDF, RDF Schema, SPARQL (3)

- Expressive means for simple „ontologies“
  - Classes
    - `rdfs:Class` (rdf:type already in RDF!)
    - `rdfs:subClassOf`
    - classes and individuals in one graph, no definitions
  - Properties
    - `rdfs:subPropertyOf`, `rdfs:domain`, `rdfs:range`
  - Reification of (s,p,o) triples as nodes:
    - `rdf:subject`, `rdf:predicate`, `rdf:object`
  - Utility properties
    - `rdfs:seeAlso`, `rdfs:comment`, `rdfs:label`, ...
  - RDFS(++): transitive & inverse properties, ...
RDF and OWL

- OWL XML is based on RDF and RDF Schema
  - „ABox“ as in RDF with `rdf:Description`, `rdf:type`
  - but also classes and their descriptions are nodes!
  - OWL specializes RDF Schema predicates, but also restricts possible combinations of predicates to ensure decidability
Species of OWL

- OWL Full
- OWL DL
- OWL Lite
- OWL2
  - What's new?
Rules in SWRL and nRQL

- Certain things cannot be expressed in OWL
  - no defined roles
  - famous example:
    \[ \forall x, y, z : \text{has\_brother}(x, y), \text{has\_child}(y, z) \rightarrow \text{has\_uncle}(z, x) \]
  - possible with SWRL rule
  - or nRQL rule
    (prepare-abox-rule
     (and (?x ?y has-brother)
      (?y ?z has-child))
     ((related ?z ?x has-uncle)))
  - decidable, if DL-safe (rules are only applied to named individuals in \( \Delta^I \))
  - more expressive rules in nRQL
Non-monotonic Rules in nRQL

• find *mother* without known children
  (retrieve (?x)
   (and (?x mother)
    (neg (project-to (?x)
       (?x ?y has-child)))))

• add an explicit child
  (firerule
   (and (?x mother)
    (neg (project-to (?x)
       (?x ?y has-child)))))
   ((related ?x (new-ind child-of ?x) has-child))

• not possible with SWRL

• no automatic rule application strategy in nRQL
Application: Sudoku

\[
\text{pairwise_disjoint}(C_1, C_2, C_3, C_4) \\
\top \subseteq (C_1 \sqcup C_2 \sqcup C_3 \sqcup C_4) \cap (C_1' \rightarrow \forall R. \neg C_1) \cap (C_2' \rightarrow \forall R. \neg C_2) \cap (C_3' \rightarrow \forall R. \neg C_3) \cap (C_4' \rightarrow \forall R. \neg C_4) \cap \ldots
\]
ABox construction

- by hand (OK for 2x2, but for 3x3 ?)
- transitive + symmetric role? →
- use different „backward“ role for other direction, qualification over common parent role
- use a rule to create inverse edges

\[ C_4 \rightarrow \forall R. \neg C_4 \]
### Sudoku (3)

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
Q_1 \subseteq R \\
Q_2 \subseteq R \\
\text{transitive}(Q_1) \\
\text{transitive}(Q_2) \\
Q_1(x, y) \rightarrow Q_2(y, x)
\end{align*}
\]
Sudoku (4)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

(retrieve (?x) (?x c1))
(((?X I41)) ((?X I12)) ((?X I24)) ((?X I33)))

(retrieve (?x) (?x c2))
(((?X I34)) ((?X I42)) ((?X I11)) ((?X I23)))

(retrieve (?x) (?x c3))
(((?X I14)) ((?X I31)) ((?X I43)) ((?X I22)))

(retrieve (?x) (?x c4))
(((?X I21)) ((?X I13)) ((?X I44)) ((?X I32)))
RacerPro & Friends

- To obtain a free educational trial version of RacerPro, please visit
  http://www.racer-systems.com/products/download/education.phtml

- Demo Session
  - Protege 3.4 Beta (OWL)
  - RacerPorter / people+pets.owl