F. Description Logics

This section is based on material from

- Ian Horrocks: http://www.cs.man.ac.uk/~horrocks/Teaching/cs646/

Description Logics

- OWL DL ist äquivalent zur Beschreibungslogik $SHOIN(D_n)$. Auf letzterer basiert also die Semantik von OWL DL.
- Unter Beschreibungslogiken (Description Logics) versteht man eine Familie von Teilsprachen der Prädikatenlogik 1. Stufe, die entscheidbar sind.
- $SHOIN(D_n)$ ist eine relativ komplexe Beschreibungslogik.
- Um einen ersten Einblick in das Prinzip der Beschreibungslogiken zu erhalten, werfen wir zum Abschluss der Vorlesung einen Blick auf etwas abgespeckte Fassungen.

Literatur:


Aside: Semantics and Model Theories

- Ontology/KR languages aim to model (part of) world
- Terms in language correspond to entities in world
- Meaning given by, e.g.:
  - Mapping to another formalism, such as FOL, with own well defined semantics
  - or a bespoke Model Theory (MT)
- MT defines relationship between syntax and interpretations
  - Can be many interpretations (models) of one piece of syntax
  - Models supposed to be analogue of (part of) world
    - E.g., elements of model correspond to objects in world
  - Formal relationship between syntax and models
    - Structure of models reflect relationships specified in syntax
  - Inference (e.g., subsumption) defined in terms of MT
    - E.g., $\mathcal{T} \models A \sqsubseteq B$ iff in every model of $\mathcal{T}$, $\text{ext}(A) \subseteq \text{ext}(B)$

Aside: Set Based Model Theory

- Many logics (including standard First Order Logic) use a model theory based on Zermelo-Frankel set theory
- The domain of discourse (i.e., the part of the world being modelled) is represented as a set (often refered as $\Delta$)
- Objects in the world are interpreted as elements of $\Delta$
  - Classes/concepts (unary predicates) are subsets of $\Delta$
  - Properties/roles (binary predicates) are subsets of $\Delta \times \Delta$ (i.e., $\Delta^2$)
  - Ternary predicates are subsets of $\Delta^3$ etc.
- The sub-class relationship between classes can be interpreted as set inclusion
- Doesn’t work for RDF, because in RDF a class (set) can be a member (element) of another class (set)
  - In Z-F set theory, elements of classes are atomic (no structure)
Aside: Set Based Model Theory Example

- Formally, the vocabulary is the set of names we use in our model of (part of) the world
  - \{Daisy, Cow, Animal, Mary, Person, Z123ABC, Car, drives, \ldots\}
- An interpretation \(\mathcal{I}\) is a tuple \(\langle \Delta, \mathcal{I} \rangle\)
  - \(\Delta\) is the domain (a set)
  - \(\mathcal{I}\) is a mapping that maps
    - Names of objects to elements of \(\Delta\)
    - Names of unary predicates (classes/concepts) to subsets of \(\Delta\)
    - Names of binary predicates (properties/roles) to subsets of \(\Delta \times \Delta\)
    - And so on for higher arity predicates (if any)

What Are Description Logics?

- A family of logic based Knowledge Representation formalisms
  - Descendants of semantic networks and KL-ONE
  - Describe domain in terms of concepts (classes), roles (relationships) and individuals
- Distinguished by:
  - Formal semantics (typically model theoretic)
    - Decidable fragments of FOL
    - Closely related to Propositional Modal & Dynamic Logics
  - Provision of inference services
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimised)

DL Architecture
**Short History of Description Logics**

**Phase 1:**
- Incomplete systems (Back, Classic, Loom, . . .)
- Based on structural algorithms

**Phase 2:**
- Development of tableau algorithms and complexity results
- Tableau-based systems for Pspace logics (e.g., Kris, Crack)
- Investigation of optimisation techniques

**Phase 3:**
- Tableau algorithms for very expressive DLs
- Highly optimised tableau systems for ExpTime logics (e.g., FaCT, DLP, Racer)
- Relationship to modal logic and decidable fragments of FOL

**Latest Developments**

**Phase 4:**
- Mature implementations
- Mainstream applications and Tools
  - Databases
    - Consistency of conceptual schemata (EER, UML etc.)
    - Schema integration
    - Query subsumption (w.r.t. a conceptual schema)
  - Ontologies and Semantic Web (and Grid)
    - Ontology engineering (design, maintenance, integration)
    - Reasoning with ontology-based markup (meta-data)
    - Service description and discovery
  - Commercial implementations
    - Cerebra system from Network Inference Ltd

**From RDF to OWL**

- Two languages developed to satisfy the requirements
  - OIL: developed by group of (largely) European researchers (several from EU OntoKnowledge project)
  - DAML-ONT: developed by group of (largely) US researchers (in DARPA DAML programme)
- Efforts merged to produce DAML+OIL
  - Development was carried out by “Joint EU/US Committee on Agent Markup Languages”
  - Extends (“DL subset” of) RDF
- DAML+OIL submitted to W3C as basis for standardisation
  - Web-Ontology (WebOnt) Working Group formed
  - WebOnt group developed OWL language based on DAML+OIL
  - OWL language now a W3C Recommendation (i.e., a standard like HTML and XML)

**Description Logic Family**

- DLs are a family of logic based KR formalisms
- Particular languages mainly characterised by:
  - Set of constructors for building complex concepts and roles from simpler ones
  - Set of axioms for asserting facts about concepts, roles and individuals
- $\mathcal{ALC}$ is the smallest DL that is propositionally closed
  - Constructors include booleans (and, or, not), and
  - Restrictions on role successors
  - E.g., concept describing “happy fathers” could be written:
    
    $\text{Man} \sqcap \exists \text{hasChild.Female} \sqcap \exists \text{hasChild.Male}$
    $\sqcap \forall \text{hasChild.}(\text{Rich} \sqcap \text{Happy})$
DL Concept and Role Constructors

- Range of other constructors found in DLs, including:
  - Number restrictions (cardinality constraints) on roles, e.g., \( \geq 3 \) hasChild, \( \leq 1 \) hasMother
  - Qualified number restrictions, e.g., \( \geq 2 \)
    hasChild.Female, \( \leq 1 \) hasParent.Male
  - Nominals (singleton concepts), e.g., \{Italy\}
  - Concrete domains (datatypes), e.g., hasAge.(\( \leq 21 \))
  - Inverse roles, e.g., hasChild: (hasParent)
  - Transitive roles, e.g., hasChild* (descendant)
  - Role composition, e.g., hasParent o hasBrother

OWL as DL: Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>FOL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>( C_1 \cap \ldots \cap C_n )</td>
<td>Human &amp; Male</td>
<td>( C_1(x) \land \ldots \land C_n(x) )</td>
</tr>
<tr>
<td>unionOf</td>
<td>( C_1 \cup \ldots \cup C_n )</td>
<td>Doctor &amp; Lawyer -Male</td>
<td>( C_1(x) \lor \ldots \lor C_n(x) )</td>
</tr>
<tr>
<td>complementOf</td>
<td>( {x_1} \cup \ldots \cup {x_n} )</td>
<td>{john} \cup {mary}</td>
<td>( \forall y.P(x, y) \rightarrow C(y) )</td>
</tr>
<tr>
<td>oneValuesFrom</td>
<td>( \exists P.C )</td>
<td>hasChild.Doctor</td>
<td>( \exists y.P(x, y) \land C(y) )</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>( \exists P.C )</td>
<td>hasChild.Lawyer</td>
<td>( \exists y.P(x, y) \land C(y) )</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>( \leq m P )</td>
<td>( \leq 1 ) hasChild</td>
<td>( \exists y.P(x, y) )</td>
</tr>
<tr>
<td>minCardinality</td>
<td>( \geq m P )</td>
<td>( \geq 2 ) hasChild</td>
<td>( \exists y.P(x, y) )</td>
</tr>
</tbody>
</table>

- XMLS datatypes as well as classes in \( \forall P.C \) and \( \exists P.C \)
  - E.g., \( \exists P.C \) hasAge.nonNegativeInteger
- Arbitrarily complex nesting of constructors
  - E.g., Person \( \cap \forall hasChild.Doctor \cup \exists hasChild.Doctor \)

DL Knowledge Base

- DL Knowledge Base (KB) normally separated into 2 parts:
  - TBox is a set of axioms describing structure of domain (i.e., a conceptual schema), e.g.:
    - HappyFather \( \equiv \) Man \( \cap \exists \) hasChild.Female \( \cap \ldots \)
    - Elephant \( \equiv \) Animal \( \cap \exists \) Large \( \cap \exists \) Grey
    - transitive(ancestor)
  - ABox is a set of axioms describing a concrete situation (data), e.g.:
    - John:HappyFather
    - <John,Mary>:hasChild

- Separation has no logical significance
  - But may be conceptually and implementationally convenient

RDFS Syntax

E.g., Person \( \cap \forall hasChild.Doctor \cup \exists hasChild.Doctor \):

\[
<\text{owl:Class}>
<\text{owl:intersectionOf}\ rdf:parseType="collection">
<\text{owl:Class} rdf:about="#Person"/>
<\text{owl:Restriction}>
<\text{owl:onProperty} rdf:resource="#hasChild"/>
<\text{owl:toClass}>
<owl:unionOf rdf:parseType="collection">
<owl:Class rdf:about="#Doctor"/>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasChild"/>
<owl:hasClass rdf:resource="#Doctor"/>
</owl:Restriction>
</owl:unionOf>
</owl:toClass>
</owl:Restriction>
</owl:intersectionOf>
\]
OWL as DL: Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\sqcap$ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\sqcap$ Male</td>
</tr>
<tr>
<td>disjointWith</td>
<td>$C_1 \sqsubseteq \neg C_2$</td>
<td>Male $\sqsubseteq \neg$ Female</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>${x_1} \equiv {x_2}$</td>
<td>President_Bush $\equiv$ G.W.Bush</td>
</tr>
<tr>
<td>differentFrom</td>
<td>${x_1} \sqsubseteq \neg {x_2}$</td>
<td>John $\subseteq \neg$ Peter</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasChild $\equiv$ hasParent$^{-}$</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \subseteq P$</td>
<td>ancestor$^+$ $\subseteq$ ancestor</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>$T \sqsubseteq \lessdot 1P$</td>
<td>T $\sqsubseteq$ x hasMother</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>$T \sqsubseteq \lessdot 1P^-$</td>
<td>T $\sqsubseteq$ x hasSSN$^-$</td>
</tr>
</tbody>
</table>

- Axioms (mostly) reducible to inclusion ($\sqsubseteq$)
  - $C \equiv D$ iff both $C \sqsubseteq D$ and $D \sqsubseteq C$

- Obvious FOL equivalences
  - E.g., $C \equiv D$ iff $\forall x. \ C(x) \leftrightarrow D(x)$,
  - $C \sqsubseteq D$ iff $\forall x. \ C(x) \Rightarrow D(x)$

Why Separate Classes and Datatypes?

- Philosophical reasons:
  - Datatypes structured by built-in predicates
  - Not appropriate to form new datatypes using ontology language
- Practical reasons:
  - Ontology language remains simple and compact
  - Semantic integrity of ontology language not compromised
  - Implementability not compromised — can use hybrid reasoner

XML Schema Datatypes in OWL

- OWL supports XML Schema primitive datatypes
  - E.g., integer, real, string, …
- Strict separation between “object” classes and datatypes
  - Disjoint interpretation domain $\Delta_D$ for datatypes
    - For a datavalue $d$ holds $d \sqsubseteq \Delta_D$
    - and $\Delta_D \sqcap \Delta^I = \emptyset$
  - Disjoint “object” and datatype properties
    - For a datatype property $P$ holds $P^D \subseteq \Delta^I \times \Delta_D$
    - For object property $S$ and datatype property $P$ hold $S^I \sqcap P^D = \emptyset$
  - Equivalent to the “$(D_n)$” in $SHOIN(D_n)$

OWL DL Semantics

- Mapping OWL to equivalent DL ($SHOIN(D_n)$):
  - Facilitates provision of reasoning services (using DL systems)
  - Provides well defined semantics
- DL semantics defined by interpretations: $\mathcal{I} = (\Delta^I, \mathcal{I})$,
  where
  - $\Delta^I$ is the domain (a non-empty set)
  - $\mathcal{I}$ is an interpretation function that maps:
    - Concept (class) name $\mathcal{A}$ to subset $\mathcal{A}^I$ of $\Delta^I$
    - Role (property) name $\mathfrak{R}$ to binary relation $\mathfrak{R}^I$ over $\Delta^I$
    - Individual name $\downarrow i$ to element $i^I$ of $\Delta^I$
**Interpretation function** \( \mathcal{I} \) extends to **concept expressions** in the obvious way, i.e.:

\[
\begin{align*}
(C \cap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\
(C \cup D)^\mathcal{I} &= C^\mathcal{I} \cup D^\mathcal{I} \\
(-C)^\mathcal{I} &= \Delta^\mathcal{I} \setminus C^\mathcal{I} \\
\{x\}^\mathcal{I} &= \{x^\mathcal{I}\} \\
\exists R.C)^\mathcal{I} &= \{x \mid \exists y. (x, y) \in R^\mathcal{I} \land y \in C^\mathcal{I}\} \\
(\forall R.C)^\mathcal{I} &= \{x \mid \forall y. (x, y) \in R^\mathcal{I} \Rightarrow y \in C^\mathcal{I}\} \\
(n R)^\mathcal{I} &= \{x \mid \#\{y \mid (x, y) \in R^\mathcal{I}\} \leq n\} \\
(\geq n R)^\mathcal{I} &= \{x \mid \#\{y \mid (x, y) \in R^\mathcal{I}\} \geq n\}
\end{align*}
\]

**Knowledge Base Semantics**

- An **interpretation** \( \mathcal{I} \) satisfies (models) an axiom \( \Lambda (\mathcal{I} \models \Lambda) \):
  - \( \mathcal{I} \models C \subseteq D \) iff \( C^\mathcal{I} \subseteq D^\mathcal{I} \)
  - \( \mathcal{I} \models C \equiv D \) iff \( C^\mathcal{I} = D^\mathcal{I} \)
  - \( \mathcal{I} \models R \subseteq S \) iff \( R^\mathcal{I} \subseteq S^\mathcal{I} \)
  - \( \mathcal{I} \models R \equiv S \) iff \( R^\mathcal{I} = S^\mathcal{I} \)
  - \( \mathcal{I} \models x \in D \) iff \( x^\mathcal{I} \in D^\mathcal{I} \)
  - \( \mathcal{I} \models (x, y) \in R \) iff \( (x^\mathcal{I}, y^\mathcal{I}) \in R^\mathcal{I} \)

- **\( \mathcal{I} \) satisfies a Tbox** \( \mathcal{T} (\mathcal{I} \models \mathcal{T}) \) iff **\( \mathcal{I} \) satisfies every axiom** \( \Lambda \) in \( \mathcal{T} \)

- **\( \mathcal{I} \) satisfies an Abox** \( \mathcal{A} (\mathcal{I} \models \mathcal{A}) \) iff **\( \mathcal{I} \) satisfies every axiom** \( \Lambda \) in \( \mathcal{A} \)

- **\( \mathcal{I} \) satisfies a KB** \( \mathcal{K} (\mathcal{I} \models \mathcal{K}) \) iff **\( \mathcal{I} \) satisfies both** \( \mathcal{T} \) and \( \mathcal{A} \)

**DL Knowledge Bases (Ontologies)**

- An **OWL ontology maps to a DL Knowledge Base** \( \mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle \)
  - \( \mathcal{T}(\text{Tbox}) \) is a set of axioms of the form:
    - \( C \subseteq D \) (concept inclusion)
    - \( C \equiv D \) (concept equivalence)
    - \( R \subseteq S \) (role inclusion)
    - \( R \equiv S \) (role equivalence)
    - \( R^+ \subseteq R \) (role transitivity)
  - \( \mathcal{A}(\text{Abox}) \) is a set of axioms of the form
    - \( x \in D \) (concept instantiation)
    - \( (x, y) \in R \) (role instantiation)

**Inference Tasks**

- Knowledge is **correct** (captures intuitions)
  - \( C \) subsumes \( D \) w.r.t. \( \mathcal{K} \) iff for every model \( \mathcal{I} \) of \( \mathcal{K} \), \( C^\mathcal{I} \subseteq D^\mathcal{I} \)

- Knowledge is **minimally redundant** (no unintended synonyms)
  - \( C \) is equivalent to \( D \) w.r.t. \( \mathcal{K} \) iff for every model \( \mathcal{I} \) of \( \mathcal{K} \), \( C^\mathcal{I} = D^\mathcal{I} \)

- Knowledge is **meaningful** (classes can have instances)
  - \( C \) is satisfiable w.r.t. \( \mathcal{K} \) iff there exists some model \( \mathcal{I} \) of \( \mathcal{K} \) s.t. \( C^\mathcal{I} \neq \emptyset \)

- **Querying** knowledge
  - \( x \) is an instance of \( C \) w.r.t. \( \mathcal{K} \) iff for every model \( \mathcal{I} \) of \( \mathcal{K} \), \( x^\mathcal{I} \in C^\mathcal{I} \)
  - \( (x, y) \) is an instance of \( R \) w.r.t. \( \mathcal{K} \) iff for every model \( \mathcal{I} \) of \( \mathcal{K} \), \( (x^\mathcal{I}, y^\mathcal{I}) \in R^\mathcal{I} \)

- Knowledge base **consistency**
  - A KB \( \mathcal{K} \) is consistent iff there exists some model \( \mathcal{I} \) of \( \mathcal{K} \)
DL Reasoning

- Tableau algorithms used to test satisfiability (consistency)
- Try to build a tree-like model $I$ of the input concept $C$

- Decompose $C$ syntactically
  - Apply tableau expansion rules
  - Infer constraints on elements of model

- Tableau rules correspond to constructors in logic ($\land$, $\lor$, etc)
  - Some rules are nondeterministic (e.g., $\lor$, $\leq$)
  - In practice, this means search

- Stop when no more rules applicable or clash occurs
  - Clash is an obvious contradiction, e.g., $A(x), \neg A(x)$

- Cycle check (blocking) may be needed for termination

- $C$ satisfiable iff rules can be applied such that a fully expanded clash free tree is constructed

Highly Optimised Implementation

- Naive implementation leads to effective non-termination
- Modern systems include MANY optimisations
- Optimised classification (compute partial ordering)
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order

- Optimised subsumption testing (search for models)
  - Normalisation and simplification of concepts
  - Absorption (rewriting) of general axioms
  - Davis-Putnam style semantic branching search
  - Dependency directed backtracking
  - Caching of satisfiability results and (partial) models
  - Heuristic ordering of propositional and modal expansion
  - ...