Knowledge-Based Systems and Deductive Databases

Christoph Lofi
Institut für Informationssysteme
Technische Universität Braunschweig
http://www.ifis.cs.tu-bs.de
12.1 Description Logics
12.2 DAML+OIL
12.3 OWL
• The **World Wide Web** is a medium of documents for **people**
• **Idea:** augment Web pages with data targeted at **computers**
  – Add documents solely for computers enhanced with semantic markups
  – Find meaning of semantic data by following hyperlinks to definitions of key terms and rules for reasoning about data logically
  – Spur development of automated web services and highly functional agents
But how can knowledge be represented and how can conclusions be drawn?

- Remember: early in AI the notation with frames was introduced by Marvin Minsky at MIT

- Then the expert systems took over with different representation frameworks and (uncertain) reasoning capabilities
  - MYCIN, etc.

And... how to do it for the Web?
• Basic Web Technology
  – Uniform Resource Identifier (URI)
    • Identify items on the Web
  – Extensible Markup Language (XML)
    • Allows anyone to design own document formats (syntax)
    • Can include markup to enhance meaning of document’s content
  – Resource Description Framework (RDF)
    • Make machine-processable statements
    • Triple of URIs: subject, predicate, object
Now comes the interesting part... **How to derive new knowledge?**

- **Definitely a formal semantics** is needed
  - There is a large number of different logics
- **Searches should to be decidable**
  - Decidability often conflicts with expressiveness
- **Different applications may need different expressiveness**
  - From simple inheritance structures, to evaluating logical expressions with full negation and quantification
- **Very tight coupling** between theory and practice
  - The evaluation needs to be fast
• In the previous lectures, we have seen different knowledge inference schemes with their respective advantages and disadvantages

• **First Order Logic**
  – **Pro:**
    • Very expressive and powerful
  – **Con:**
    • Not very intuitive, knowledge is hard to model
    • Computationally challenging
      – Undecidable in worst-case
      – EXPTIME in most cases
12.1 Description Logics

• **Horn Logic** (e.g., Datalog)
  – **Pro:**
    • Computationally manageable
  – **Con:**
    • Less expressive
    • Lacks intuitive modeling features

• **Frame Systems & Semantic Networks**
  – **Pro:**
    • Intuitive modeling
    • More human centered
  – **Con:**
    • Lacks formal semantics necessary for reasoning
12.1 Description Logics

- In the late 70ties, frame systems were quite popular
  - …but lacked formal reasoning capabilities
  - **Idea**: Combine *semantic frames* with *first order logics*
    ⇒ Description Logics
  - Description logics can be defined in various degrees of *expressiveness* by using different features of first order logic
    - Different expression classes map to *different fragments of first order logic*
    - More expressiveness ⇒ Higher computational complexity
    - Subsets of description logics are usually called *description languages*
    - Still, all description languages are *decidable*
  - Languages like RDF+RDF/S (in limited extent), OWL, DAML+OIL emerged as implementation of description logics
The basic building blocks of description logics are **concepts**, **roles** and **individuals**

- Like with frame systems, think of concepts like OO classes without methods
  - Act as “blue prints” for the concept instances
  - Each concept represents a set of actual **individuals**
  - Those individuals (or members) can be recursively enumerated
- Concepts are represented by **unary predicates**
- Concepts are embedded into an hierarchical **inheritance structure**
**12.1 Description Logics**

- Furthermore, concepts can be linked to each other by using **roles**
  - Roles are represented by **binary predicates**
- Concepts and roles use a **set-theoretical interpretation**
  - **Concept**: a set of individuals of the respective domain
  - **Role**: set of pairs of individuals of the respective domain
• Basic building blocks in DL atomic concepts and atomic roles
  – Atomic concepts and roles are given by their predicate definition as enumeration of individuals/pairs of individuals
  – Using those atoms, additional concepts and roles may be described by DL expressions
    • So called complex concepts and complex roles
12.1 Description Logics

• Informal Example:
  – Atomic concepts:
    • Person, University, Professor, Lecture
  – Atomic Roles:
    • studies, teaches
  – Individuals:
    • Prof. Balke, Christoph, Student_1, Student_2
  – Complex Concept
    • Student: Someone who studies at a university
    • TeachingAssistant: Someone who teaches but is not a professor and not a student
    • HiWi: Someone who teaches and is student
  – Complex Role
    • supervises: A professor who teaches a lecture is also supervising that lecture
  – Inheritance Hierarchy
    • A Professor is a special Person, a Student is a special Person, a TeachingAssistant is a special Person, a HiWi is a special Student
    • supervises is a stronger form of teaches
12.1 Description Logics

- Person
  - Professor
  - TeachingAssistant
  - Student
    - HiWi
    - studies
  - University
    - studies
- Lecture
  - teaches
  - teaches
- Professor
  - supervises
  - teaches
12.1 Description Logics

• In the following, we will formally describe different description languages

• As a note, all description languages use two important assumptions
  – **Open World Assumption**
    • Interpretation domain $\Delta^I$ is infinite
    • Lack of knowledge does not imply the negation of the given fact
  – **Ambiguous Name Assumption**
    • Two concepts with different names may be equivalent
      – i.e. different names do not guarantee different concepts
• DL knowledge bases consist of **two types** of expressions

– **ABox** statements (*assertion box*): provides assertions on the individuals with respect to the vocabulary
  • i.e. which individual is **member** of which concept
  • Typical reasoning tasks involve checking for assertion **consistency** (satisfiability) and checking whether a certain individual is an **instance** of a given concept
  • Thus, the ABox provides the known **facts**
12.1 Description Logics

- **TBox statements** (*terminology box*): Defines the vocabulary of the knowledgebase
  - Used *description language* controls the complexity of the TBox
  - Provides the *model-theoretic foundation* for later reasoning
  - Defines *complex concepts* and *complex roles*
  - Typical reasoning tasks for TBox is checking for concept or role *subsumption*
For providing TBox statements, different Description Languages are available

- Languages differ with respect to their features
- Each new feature adds additional complexity and expressiveness
- Description languages are named and classified by their feature sets
- Most basic description language
  - $\mathcal{AL}$: Attribute Language
• Description Languages use a variable free syntax
  – Variables are modeled implicitly
    • e.g. The description logic expression $C \cap D$ can be translated into $C(x) \land D(x)$
  – So, why do we need a new syntax?
    • Variable free syntax is much shorter and simpler, e.g. the short DL statement $\geq 4 R$ translates to
      $$\exists y_1, y_2, y_3, y_4 (R(x, y_1) \land R(x, y_2) \land R(x, y_3) \land R(x, y_4) \land y_1 \neq y_2 \land y_1 \neq y_3 \land y_1 \neq y_4 \land y_2 \neq y_3 \land y_2 \neq y_4 \land y_3 \neq y_4$$
12.1 Description Language $\mathcal{AL}$

- **Description Language $\mathcal{AL}$**
  - Minimal description language with practical applicability

- **Allowed syntactical constructs and their interpretations**
  - **Atomic concepts** (denoted by $A$ and $B$)
    - $A^I \subseteq \Delta^I$, $B^I \subseteq \Delta^I$
  - **Atomic roles** (denoted by $R$)
    - $R^I \subseteq \Delta^I \times \Delta^I$
  - **Complex concepts** (denoted by $C$ and $D$)
    - $C^I \subseteq \Delta^I$, $D^I \subseteq \Delta^I$
– $\top$: **Top** or **Universal concept**, represents the whole domain of all individuals  
  • $\top^I = \Delta^I$

– $\bot$: **Bottom concept**, represents the empty set of individuals  
  • $\bot^I = \emptyset$

– $\neg A$: **Atomic negation**  
  • Negation in $\mathcal{AL}$ only possible on **atomic concepts**  
  • Due to open world assumption, resulting set is **infinite**  
  • $(\neg A)^I = \Delta^I \setminus A^I$

– $C \cap D$: **Intersection**  
  • All individuals which are both $C$ and $D$  
  • $(C \cap D)^I = C^I \cap D^I$
12.1 Description Language $\mathcal{AL}$

- **$\forall R.C$ : Value Restriction**
  - Defines the set of all those individuals which are in relationship $R$ with individuals in $C$ (and only those)
  - e.g. $\forall$ studies.University : all individuals who study only at universities
  - $(\forall R.C)^I = \{ a \in \Delta^I | \forall b ((a, b) \in R^I \rightarrow b \in C^I ) \}$

- **$\exists R.T$ : Limited existential restriction**
  - Defines the set of all those individuals which have a relationship partner in $R$
  - e.g. $\exists$ studies.$T$ : all individuals who study (somewhere)
  - Only top concept is allowed as concept
  - $(\exists R.T)^I = \{ a \in \Delta^I | \exists b ((a, b) \in R^I ) \}$
The TBox contains defining statements for complex concepts, realized by **terminological axioms**

- **C ⊑ D: Inclusion**
  - C is included in D (and is thus a sub-concept)
  - Each individual in C is also an individual in D
  - Provide further information on how concepts and roles are related
  - C\textsubscript{I} ⊑ D\textsubscript{I}

- **C ≡ D: Equivalence**
  - C is equivalent to D (and is thus identifies the same individuals)
  - Each individual in C is also an individual in D and vice versa
  - Equivalence can be used to define new complex concepts
  - C\textsubscript{I} = D\textsubscript{I}
• Terminological axioms can also be used to model and check assertions about classes
  – E.g., disjoint classes: \( D \cap E \equiv \bot \)
  – Such checks are quite often used for
    • Checking the **consistency** of an ontology and knowledge
    • Checking for **unintended relationships** between classes
    • **Automatically classifying** instances in classes
    • Finding **inconsistencies** when designing large ontologies (especially if multiple authors are involved)
• Our previous example in $\mathcal{AL}$
• Atomic concepts and roles are given by ABox statements
  – $\text{Person}(\text{Prof. Balke}), \text{Person}(\text{Christoph}), \text{Person}(\text{Student}_A), \text{Person}(\text{Student}_B)$ 
    $\text{University}(\text{TU Braunschweig}), \text{Professor}(\text{Prof. Balke}), \text{Lecture}(\text{KBS}),$ 
  – $\text{studies}(\text{Student}_A, \text{TU Braunschweig}), \text{studies}(\text{Student}_B, \text{TU Braunschweig})$ 
  – $\text{teaches}(\text{Prof. Balke}, \text{KBS}), \text{teaches}(\text{Christoph}, \text{KBS})$
• Complex concepts are given by TBox statements
  – Complex Concept
    • $\text{Student} \equiv \text{Person} \sqcap \forall \text{studies.University}$ 
    • $\text{TeachingAssistant} \equiv \text{Person} \sqcap \forall \text{teaches.Lecture} \sqcap \neg \text{Student} \sqcap \neg \text{Professor}$ 
    • $\text{HiWi} \equiv \text{Student} \sqcap \forall \text{teaches.Lecture}$
Typical reasoning queries are mainly of classifying nature

- E.g., return all students, HiWis or ResearchAssistants
  - Those concepts have not been explicitly defined by ABox statements, but can only be derived using TBox descriptions

Note that $\mathcal{AL}$ is the least expressive common description language and has limited expressiveness

- Additional features can expand $\mathcal{AL}$
12.1 Expanding $\mathcal{AL}$

• **Expansion $\mathcal{C}$: General complement $\neg \mathcal{C}$**
  
  – $\mathcal{AL}$ allowed only negation of atomic concepts
  
  – However, often general negation is necessary (the complement), e.g.:
    
    • $\text{HiWi} \equiv \text{Student} \sqcap \forall \text{teaches.Lecture}$
    
    • $\text{LazyStudent} \equiv \text{Student} \sqcap \neg \text{HiWi}$
      
      – HiWi is a complex concept, thus this expression is not allowed in $\mathcal{AL}$
    
  – $(\neg \mathcal{C})^I = \Delta^I \setminus C^I$
  
  – If general complements are allowed, this results to the language $\mathcal{ALC}$

    • Naming convention: Start with $\mathcal{AL}$ and concatenate the short letters of all additional features…
12.1 Expanding $\mathcal{AL}$

• Expansion $\mathcal{U}$: Union $C \sqcup D$
  – Allows to union two complex concepts
  – $(C \sqcup D)^I = C^I \cup D^I$
  – Results to $\mathcal{ALU}$

• Expansion $\mathcal{E}$:
  **Full existential quantification $\exists R.C$**
  – In contrast to limited existential quantification in $\mathcal{AL}$, any concept is allowed in existential quantification
  – $(\exists R.C)^I = \{a \in \Delta^I \mid \exists b ((a, b) \in R^I \land b \in C^I)\}$
12.1 Expanding \( \mathcal{AL} \)

• For description logics hold also the known equivalences of first order logics
  
  – \((C \sqcup D) \equiv \neg (\neg C \sqcap \neg D)\)
  
  – \(\exists \, R.\, C \equiv \neg \forall \, R.\, \neg C\)
  
  – Thus, union and full existential quantification can be modeled by using the general complement and vice versa

  – Therefore, \( \mathcal{ALC} \) has the same expressiveness as \( \mathcal{ALUE} \)
    
      • Therefore, we will use \( \mathcal{ALC} \) to refer also to \( \mathcal{ALUEC} \) or \( \mathcal{ALUE} \)
• Modeling differences for existential and universal quantification
  – In our example, we defined a TA as \( \text{TeachingAssistant} \equiv \text{Person} \sqcap \neg \text{Student} \sqcap \neg \text{Professor} \sqcap \forall \text{teaches.Lecture} \)
    • This actually means: All persons which are no student, no professor, teach something, and \textbf{everything they teach is a lecture}
    • Thus, as soon as a person also teaches a lab course, he is not a TA anymore…
  – Better expression: \( \text{TeachingAssistant} \equiv \text{Person} \sqcap \neg \text{Student} \sqcap \neg \text{Professor} \sqcap \exists \text{teaches.Lecture} \)
    • All persons which are no student, no professor and teach at least one lecture
    • Thus, for this simple statement you already need \( \mathcal{ALC} \)
12.1 Expanding $\mathcal{AL}$

- Expansion $\mathcal{N}$: **Number Restriction** $\geq n R$
  - Comes in two flavors:
    - At-least-Restriciton: $\geq n R.C$
    - At-most-Restriciton: $\leq m R.C$
  - $\geq n R.C \cap \leq m R.C$: This restricts that each individual which participates in the relationship $R$ needs to be related to at least $m$ and at most $n$ other individuals
    - $(\geq n R.C)^I = \{ a \in \Delta^I \mid \| \{ b \mid (a, b) \in R^I \} \| \geq n \}$
    - $(\leq n R.C)^I = \{ a \in \Delta^I \mid \| \{ b \mid (a, b) \in R^I \} \| \leq n \}$
Terminological equivalence statements are called \textit{definitorial}, if they are acyclic after expression optimization

- $C \equiv D \cap E$
  • is \textit{acyclic} and \textit{definitorial}

- $C \equiv D \cup \exists R.C$
  • is \textit{cyclic} and \textit{not definitorial}

- $C \equiv D \cup \exists R.(C \cap \neg C)$
  • is \textit{cyclic} and \textit{definitorial}

- Why? Query can be simplified to $C \equiv D \cup \exists R.(\bot) \equiv D$
  - Thus, despite containing a cycle, it can be simplified to an acyclic expression
• If all definitions in a TBox are **definitorial**, we call the TBox **acyclic**
  – The case of acyclic TBoxes can be reduced to a case with an empty TBox by “**expanding**” the TBox into explicit ABox statements
    • e.g.
      \[
      \text{Person}(\text{Christoph}), \text{TeachingAssistant} \equiv \text{Person} \sqcap \\
      \forall \text{teaches. Lecture} \sqcap \neg \text{Student} \sqcap \neg \text{Professor} \\
      \Rightarrow \text{Person}(\text{Christoph}), \text{TeachingAssistant}(\text{Christoph})
      \]
  – In case of weak languages, this significantly increases the space and time complexity
    • For more expressive languages, it does not matter complexity-wise, if a TBox is used or not
• **Cyclic TBoxes** allow to recursively define concepts
  
  – e.g. TomsAncestors ≡ Tom ⊔ ∃parent. TomsAncestor
  
  – Recursive definitions usually **increase the complexity significantly**
  
  – Recursive definition of concepts lead to the already known **fixpoint semantics**
  
  – This feature can be used to model **transitive roles**
12.1 Complexity of $\mathcal{AL}$

- **Complexity of $\mathcal{AL}$**
- Commonly, several reasoning problems can be examined
  - **General satisfiability**: $\Sigma \not\models \bot$
    - Is the whole system satisfiable, i.e. does $\Sigma$ have a model
  - **Concept satisfiability**: $\Sigma \not\models C \equiv \bot$
    - Is a given concept $C$ satisfiable, i.e. is there a model such that $C^I \neq \emptyset$
    - Example of unsatisfiable concept: Student \(\cap \neg\)Person
12.1 Complexity of $\mathcal{ALC}$

- **Subsumption**: $\Sigma \models C \sqsubseteq D$
  - Is the concept $C$ a sub-concept of $D$, i.e. is for every model $C^I \subseteq D^I$
  - Example of a subsumed concept: Student $\sqsubseteq$ Person

- **Instance Checking**: $\Sigma \models C(a)$
  - Is $a$ an instance of $C$, i.e. is $C(a)$ satisfied in every model of $\Sigma$

- **Retrieval**: $\{a \mid \Sigma \models C(a)\}$
  - Return all individuals being member of a certain concept

- **Realization**: $\{C \mid \Sigma \models C(a)\}$
  - Return all concepts realizing a given individual
12.1 Complexity of $\mathcal{ALC}$

- Most of these problems are reduceable (with overhead) to general satisfiability
- **Reasoning complexity** for $\mathcal{AL}$ languages

<table>
<thead>
<tr>
<th>Language</th>
<th>$\models C \sqsubseteq D$</th>
<th>$\models C(a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{AL}$</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>$\mathcal{ALE}$</td>
<td>NP</td>
<td>PSPACE</td>
</tr>
<tr>
<td>$\mathcal{ALC}$</td>
<td>PSPACE</td>
<td>PSPACE</td>
</tr>
<tr>
<td>$\mathcal{ALC}$ cyclic TBox</td>
<td>ExpTIME</td>
<td>ExpTIME</td>
</tr>
</tbody>
</table>
12.1 Complexity of $\mathcal{ALC}$

• Up to now, we have considered basic description languages based on $\mathcal{AL}$
  – They allow definition of facts via $\mathcal{ABox}$ statements and definition of complex concepts via $\mathcal{TBox}$ definitions
    • Suitable for concept hierarchies and respective classification problem
  – Starting from here, more powerful description languages emerged which focus on additionally modeling complex roles
    • Complex role definitions are collected in the $\mathcal{RBox}$
12.1 Description Language $\mathcal{S}$

- Languages using RBox build on top of complete $\mathcal{AL}$ language
  - Use $\mathcal{ALCUE}$ as base language
- The first and simplest RBox feature is role transitivity $R^+$
  - Transitive roles implicitly also contain the full transitive closure of those facts given in the ABox
  - $(R^+)^I = \bigcup_{i \geq 1} (R^I)^i$
  - As the naming scheme has become a little bit clumsy, the short name $\mathcal{S}$ is introduced: $\mathcal{S} = \mathcal{ALCUE} +$ transitive roles
12.1 Description Language $\mathcal{S}$

- **Example of $\mathcal{S}$**
  - **ABox**
    - $\text{parent}(\text{Thomas, John}); \text{parent}(\text{Mary, John});$
      $\text{parent}(\text{George, Thomas}); \text{parent}(\text{Sonja, Thomas});$
      $\text{parent}(\text{Peter, Mary}); \text{parent}(\text{Karen, Mary});$
  - **RBox**
    - $\text{ancestor} \equiv \text{parent}^+$
  - **TBox**
    - $\text{TomsAncestors} \equiv \exists \text{Tom.ancestor}$

- **Like the base language $\mathcal{AL}$, also $\mathcal{S}$ can be expanded with additional features**
  - **Many of those features are just convenience features and do not directly increase expressiveness**
12.1 Expanding $\mathcal{S}$

- Expansion $\mathcal{H}$: **Role hierarchies $R \sqsubseteq S$**
  - Allows the construction of *role hierarchies* using the **inclusion** $R \sqsubseteq S$
    - $(R \subseteq S)^I = R^I \subseteq S^I$
  - Role hierarchies alone just add additional **restrictions** on role individuals in the ABox and allow for more **reasoning capabilities**
    - e.g. ABox consistency, role subsumption, role membership, etc.
    - The construction of complex roles is not supported by $\mathcal{H}$
12.1 Expanding $S$

• Additionally, further complex role definition constructs are possible which may optionally be added to the language
  
  – They do not have own feature symbols
  
  – Complex role definitions using intersection $R \cap S$, union $R \cup S$, negation $\neg R$, and composition $R \circ S$

    • $(R \cap S)^I = R^I \cap S^I$
    • $(R \cup S)^I = R^I \cap S^I$
    • $(\neg R)^I = \Delta^I \times \Delta^I \setminus R^I$
    • $(R \circ S)^I = \{(a, c) \in \Delta^I \times \Delta^I \mid \exists b (a, b) \in R^I \land (b, c) \in S^I\}$

    • Examples:
      
      – Atomic roles: attendsLecture, mother, parent
      
      – Complex roles: skipsLecture $\equiv \neg$attendsLecture
                      grandmother $\equiv$ parent $\circ$ mother
12.1 Expanding $\mathcal{S}$

• Expansion $\mathcal{J}$: Role inverses $R^-$
  – An inverse role is obtained when the arguments are swapped
  – $(R^-)^I = \{(b, a) \in \Delta^I \times \Delta^I \mid (a, b) \in R^I\}$
  – Example:
    - Assertion: $\text{teaches}(\text{Prof. Balke, KBS})$
    - Complex role: $\text{isToughtBy} \equiv \text{teaches}^-$
    - Results to: $\text{isToughtBy} (\text{KBS, Prof. Balke})$
12.1 Expanding $\mathcal{S}$

- **Expansion $\mathcal{F}$: Functionality $\preceq 1 R$**
  - Functionality restricts the maximum number of role relation sources to 1
  - Thus, the role becomes a **function**
    - Thus, the second role argument is **fully functional dependent** on the first one
    - $R(x, y) \iff f(x) = y$
  - **Example:**
    - *age* or *mother* are functional (you can only have one age and one mother)
12.1 Expanding $\mathcal{S}$

• **Expansion $\emptyset$: Nominals**
  
  – Numerals are just a convenience feature for defining concepts as sets
  
  – The members of the concept are simply enumerated
    
    • Degrees $\equiv \{\text{BSC-Inf, MSC-Inf, BSC-WiInf, MSC-WiInf}\}$
    
    • Degrees$^I = \{\text{BSC-Inf}^I, \text{MSC-Inf}^I, \text{BSC-WiInf}^I, \text{MSC-WiInf}^I\}$
## 12.1 Language Summary

- **Summary of description language features**

<table>
<thead>
<tr>
<th>Short</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{AL}$</td>
<td>Base attribute language</td>
</tr>
<tr>
<td>$\mathcal{C}$</td>
<td>General complement $\neg \mathcal{C}$</td>
</tr>
<tr>
<td>$\mathcal{U}$</td>
<td>Union $\mathcal{C} \sqcup \mathcal{D}$</td>
</tr>
<tr>
<td>$\mathcal{E}$</td>
<td>Full existential quantification $\exists \mathcal{R}.\mathcal{C}$</td>
</tr>
<tr>
<td>$\mathcal{N}$</td>
<td>Unqualified number restrictions $\geq n \mathcal{R}$ and $\leq n \mathcal{R}$</td>
</tr>
<tr>
<td>$\mathcal{S}$</td>
<td>$\mathcal{ALC}$ + transitive roles $\mathcal{R}^+$</td>
</tr>
<tr>
<td>$\mathcal{H}$</td>
<td>Role hierarchies $\mathcal{R} \sqsubseteq \mathcal{S}$</td>
</tr>
<tr>
<td>$\mathcal{J}$</td>
<td>Role inverses $\mathcal{R}^-$</td>
</tr>
<tr>
<td>$\mathcal{F}$</td>
<td>Functionality $\leq 1 \mathcal{R}$</td>
</tr>
<tr>
<td>$\mathcal{O}$</td>
<td>Nominals</td>
</tr>
<tr>
<td>$\mathcal{R}$ (not in lecture)</td>
<td>Complex role inclusions $\mathcal{R} \circ \mathcal{S} \sqsubseteq \mathcal{S}$ and $\mathcal{R} \circ \mathcal{S} \sqsubseteq \mathcal{R}$</td>
</tr>
<tr>
<td>$\mathcal{Q}$ (not in lecture)</td>
<td>Qualified number restrictions $\geq n \mathcal{R}.\mathcal{C}$ and $\leq n \mathcal{R}.\mathcal{C}$</td>
</tr>
</tbody>
</table>
### 12.1 Complexity

- **Reasoning complexity for $\mathcal{AL}$ and $\mathcal{S}$**

<table>
<thead>
<tr>
<th>Language</th>
<th>$\models C \subseteq D$</th>
<th>$\models C(a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{AL}$</td>
<td>P-comp.</td>
<td>P-comp.</td>
</tr>
<tr>
<td>$\mathcal{ALE}$</td>
<td>NP-comp.</td>
<td>PSPACE-comp.</td>
</tr>
<tr>
<td>$\mathcal{ALC}$</td>
<td>PSPACE-comp.</td>
<td>PSPACE-comp.</td>
</tr>
<tr>
<td>$\mathcal{ALC}$ cyclic TBox</td>
<td>ExpTIME-comp.</td>
<td>ExpTIME-comp.</td>
</tr>
<tr>
<td>$\mathcal{S}$</td>
<td>ExpTIME-comp.</td>
<td>ExpTIME-comp.</td>
</tr>
<tr>
<td>$\mathcal{SHIF}$</td>
<td>ExpTIME-comp.</td>
<td>ExpTIME-comp.</td>
</tr>
<tr>
<td>$\mathcal{SHOIN}$</td>
<td>NExpTIME-comp.</td>
<td>NExpTIME-comp.</td>
</tr>
</tbody>
</table>
12.1 Description Logics

• Further material
  – Daniele Nardi, Ronald J. Brachman. *An Introduction to Description Logics*
  – Franz Baader, Werner Nutt. *Basic Description Logics*
  – Complexity of Description Languages
    • [http://www.cs.man.ac.uk/~ezolin/dl/](http://www.cs.man.ac.uk/~ezolin/dl/)
The theory of description logics did not yet specify its actual application on semantic markups:

- Basically the Semantic Web is a collection of RDF/S statements.
- How can these statements be evaluated to derive new knowledge?

Two major standards for the actual application were developed in parallel:

- DAML and OIL.
- Around 2000 combined into DAML+OIL.
12.2 DAML+OIL

• The pure structural modeling is obviously not enough
  – Needed is a logic layer on top of RDF allowing for inference
  – Some type of description logic is a promising possibility
• From 1999-2006 the DARPA Agent Markup Language (DAML) program was designed to facilitate the concept of the Semantic Web

– Jim Hendler (University of Maryland, College Park)

– RDF-based markup language for agents

– Important focus on tools for the intuitive formulation of knowledge
• DAML RDF Editor
• DAML RDF Editor
The **Ontology Inference Layer** (OIL) is a proposal for a web-based representation and inference layer for ontologies:

- Provides widely used modeling primitives from frame-based languages
- Combined with the formal semantics and reasoning services provided by description logics
- Compatible with RDF Schema (RDFS), and includes a precise semantics for describing term meanings (and thus also for describing implied information)
OIL was an academically driven initiative defining a **logical foundation** for the Semantic Web

- Dieter Fensel, Frank van Harmelen (VU Amsterdam)
- Ian Horrocks (University of Manchester)
- Deborah McGuinness (Stanford)
- …
• OIL presented a **layered approach** to a standard ontology language
  
  – Each additional layer **adds functionality** and **complexity** to the previous layer
    • Like seen in the description logics layers
  
  – Agents who can only process a lower layer can still **partially understand** ontologies that are expressed in any of the higher layers (dumb down principle)
• **Core OIL** coincides largely with RDF Schema
  – With the exception of the reification features of RDF Schema
  – Even simple RDF Schema agents are able to process the OIL ontologies, and pick up as much of their meaning as possible with their limited capabilities

• **Standard OIL** is a language intended to capture the necessary modeling primitives
  – That are **well understood** thereby allowing the semantics to be precisely specified
  – That provide **adequate expressive power** and complete inference to be viable
• **Instance OIL** includes a thorough individual integration
  
  – While Standard OIL included modeling constructs that allow individual fillers to be specified in term definitions, Instance OIL includes a full-fledged database capability

• **Heavy OIL** may include additional representational (and reasoning) capabilities.

• For further information
  
  – [http://www.w3.org/TR/daml+oil-reference](http://www.w3.org/TR/daml+oil-reference)
After DAML+OIL a common effort to standardize an ontology language for the Web was made.

The result is the Web ontology language (OWL)

- OWL is a fragment of first order logic
- Became a W3C recommendation in 2004
- [http://www.w3.org/TR/owl-features/](http://www.w3.org/TR/owl-features/)
• Basically the OWL language comprises three complexity classes
  – **OWL Lite** was originally intended to support those users primarily needing a classification hierarchy and simple constraints
    • Corresponds to $\mathcal{SHIF}(\mathcal{D})$
      – And is thus **ExpTIME-complete**
    – **Features**:
      - **Concepts** (Complement, Union, Existential Quantification, Universal Quantification)
      - **Roles** (Transitive, Hierarchy, Inverse, Functional)
      - Additional features for data types ($\mathcal{D}$)
    • The initial hope was that it would be easy to support users with simple tools; but that proved wrong, since many OWL DL features can be built by complex OWL Lite expressions
    • Thus, OWL Lite is not widely used
– **OWL DL** was designed to provide the maximum expressiveness possible while retaining computational completeness and decidability

- OWL DL includes all OWL language constructs, but they can be used only under certain restrictions
- Corresponds to $\mathcal{SHOIN}(\mathcal{D})$
  - And is thus $\text{NExpTIME}$-complete
- **Features**:
  - Concepts (Complement, Union, Existential Quantification, Universal Quantification, Nominals)
  - Roles (Transitive, Hierarchy, Inverse, Functional, Unqualified restriction)
  - Additional features for data types ($\mathcal{D}$)
• **OWL Full** is based on a different semantics from OWL Lite or DL, and was designed to preserve the compatibility with RDF Schema
  - For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right
    - Thus OWL DL is based on second order logic and thus beyond FOL
  - OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary
  - It is unlikely that any reasoning software will be able to support complete reasoning for OWL Full, since it is not decidable!
Thus, OWL is just an **XML syntax** to encapsulate their respective description logic languages (or second order logic in case of OWL-FULL)

In the following slides, we just provide short examples of the syntax, further fun with OWL will be in the exercises 😊
12.3 OWL

- Classes are defined using `owl:Class`
  - `owl:Class` is a subclass of `rdfs:Class`

- Disjointness (\(C \cap D \equiv \bot\)) is defined using `owl:disjointWith`

```xml
<owl:Class rdf:about="#associateProfessor">
  <owl:disjointWith rdf:resource="#professor"/>
  <owl:disjointWith rdf:resource="#assistantProfessor"/>
</owl:Class>
```
• **owl:equivalentClass** defines equivalence of classes
  
  – i.e. $C \equiv D$

```xml
<owl:Class rdf:ID="faculty">
  <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```

• **owl:Thing** is the most general class, which contains everything (corresponds to $\top$)

• **owl:Nothing** is the empty class (corresponds to $\bot$)
In OWL there are two kinds of properties (in DL roles)

- **Object properties**, which relate objects to other objects
  - E.g. taughtBy, supervises

- **Data type properties**, which relate objects to datatype values (this corresponds to the RDF literals)
  - E.g. phone, title, age, etc.
• Object properties have a domain and range, additional constraints known from DL are possible
  – E.g. inverse, transitive, hierarchical, etc.

```xml
<owl:ObjectProperty rdf:ID="teaches">
    <rdfs:range rdf:resource="#course"/>
    <rdfs:domain rdf:resource="#academicStaffMember"/>
    <owl:inverseOf rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
```
• **OWL datatype properties** makes use of XML Schema data types

```xml
<owl:DatatypeProperty rdf:ID="age">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"/>
</owl:DatatypeProperty>
```
12.4 Protégé

• One of the major tools for building Ontologies is Protégé of Stanford and Manchester University
  – Open Source http://protege.stanford.edu/
  – Mainly supports ontology languages like RDF and OWL
  – Additional plug-ins extend Protégé’s functionality

• Protégé supports multiple views, representing TBox, ABox, and RBox
12.4 Protégé - TBox

Class Hierarchy

HiWi ⊆ Student
HiWi ≡ Student ⊓ ∃teaches.Lecture
Student ≡ ∃studiesAt.University

Annotations & Comments

Assertions
12.4 Protégé - RBox
12.4 Protégé - ABox

Individual list

Concept Assertions

Annotation & Comments

Role Assertions
12.4 Protégé - Reasoner

Query Classes

Query Result

Query Type
12.4 Protégé – OWL/XML

```xml
<SubClassOf>  
  <Class URI="http://ifis.cs.tu-bs.de/onto/university.owl">
    <SubClassOf>
      <Class URI="&university;HiWi"/>
      <Class URI="&university;Student"/>
    </SubClassOf>
  </Class>
</SubClassOf>  
<SubClassOf>
  <Class URI="&university;HiWi"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty URI="&university;teaches"/>
    <Class URI="&university;Lecture"/>
  </ObjectSomeValuesFrom>
</SubClassOf>  
<SubClassOf>
  <Class URI="&university;HiWi"/>
  <Declaration>
    <Class URI="&university;Lecture"/>
  </Declaration>
  <Declaration>
    <Class URI="&university;Person"/>
  </Declaration>
</SubClassOf>  
<SubClassOf>
  <Class URI="&university;Professor"/>
  <Class URI="&university;Person"/>
</SubClassOf>  
<DisjointClasses>
  <Class URI="&university;Professor"/>
  <Class URI="&university;TeachingAssistant"/>
</DisjointClasses>  
<Declaration>
  <Class URI="&university;Professor"/>
</Declaration>  
<SubClassOf>
  <Class URI="&university;Student"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty URI="&university;studiesAt"/>
    <Class URI="&university;University"/>
  </ObjectSomeValuesFrom>
</SubClassOf>  
<Declaration>
  <Class URI="&university;Student"/>
</Declaration>
```
• The Wisdom of Crowds
  – Folksonomies
  – Social software