13.0 Building the Semantic Web

- 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

13.0 Building the Semantic 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 URLs: subject, predicate, object

13.0 Building the Semantic Web

- **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**?

13.0 Building the Semantic Web

- **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
13.1 Description Logics

- 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

- **Frame Systems & Semantic Networks**
  - **Pro:**
    - Intuitive modeling
    - More human centered
  - **Con:**
    - Lacks formal semantics necessary for reasoning

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

- In the late 70ties, frame systems were quite popular
  - 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+RDFS (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

- 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
13.1 Description Logics

- Informal Example:
  - Atomic concepts:
    - Person, University, Professor, Lecture
  - Atomic Roles:
    - studies, teaches
  - Individuals:
    - Prof. Balke, Christoph, Student_1, Student_2
  - Complex Concept
    - Person
  - Complex Role
    - teaches, supervises

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

Knowledge-Based Systems and Deductive Databases
- Wolf Tilo Balke & Christoph Lofi
- IfIS – TU Braunschweig

13.1 Description Logics

- 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

- 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

Knowledge-Based Systems and Deductive Databases
- Wolf Tilo Balke & Christoph Lofi
- IfIS – TU Braunschweig
13.1 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 \( \exists y_1 y_2 y_3 y_4 (R(x,y_1) \land R(x,y_2) \land R(x,y_3) \land R(x,y_4) \land y_1 \neq y_2 \land y_1 \neq y_3 \land y_1 \neq y_4 \land y_2 \neq y_3 \land y_2 \neq y_4 \land y_3 \neq y_4) \)

13.1 Description Language $\mathcal{AL}$

- $\top$: Top or Universal concept, represents the whole domain of all individuals
  - $\top \models A'$
- $\bot$: Bottom concept, represents the empty set of individuals
  - $\bot \models \emptyset$
- $\neg$: Atomic negation
  - Negation in $\mathcal{AL}$ only possible on atomic concepts
  - Due to open world assumption, resulting set is infinite
    - $\neg (\neg A') = A' \setminus A'$
- $C \cap D$: Intersection
  - All individuals which are both $C$ and $D$
  - $(C \cap D)' = C' \cap D'$

The TBox contains defining statements for complex concepts, realized by terminological axioms

- $C \sqsubseteq D$: 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
  - $C' \sqsubseteq D'$
- $C \equiv D$: 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
    - $C' \equiv D'$

13.1 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
    - $(\forall R.C)' = \{a \in D' \mid \forall b (a,b) \in R \land (a,b) \in C')$
      - Top concept
- $\exists R.T$: Limited existential restriction
  - Defines the set of all those individuals which have a relationship partner in $R$
  - e.g. $\exists$studies:Teacher: all individuals who study (somewhere)
  - Only top concept is allowed as concept
  - $(\exists R.T)' = \{a \in D' \mid \exists b ((a,b) \in R')$
13.1 Description Language $\alpha\ell$

- **Our previous example in $\alpha\ell$**
- Atomic concepts and roles are given by ABox statements
  - Person(Prof. Balke), Person(Christoph), Person(Student_A), Person(Student_B)
  - University(TU Braunschweig), Professor(Prof. Balke), Lecture(KBS),
  - studies(Student_A, TU Braunschweig), studies(Student_B, TU Braunschweig)
  - teaches(Prof. Balke, KBS), teaches(Christoph, KBS)
- Complex concepts are given by TBox statements
  - Complex Concept
    - Student $\equiv$ Person $\land$ studies.University
    - TeachingAssistant $\equiv$ Person $\land$ teaches.Lecture $\land$ ?Student $\land$ ?Professor
    - HiWi $\equiv$ Person $\land$ teaches.Lecture

13.1 Expanding $\alpha\ell$

- **Expansion $\gamma$: General complement $\neg C$**
  - $\alpha\ell$ allowed only negation of atomic concepts
  - However, often general negation is necessary (the complement), e.g.: $\HiWi \equiv$ Person $\land$ teaches.Lecture
  - LazyStudent $\equiv$ Person $\land$ $\neg$ HiWi
  - HiWi is a complex concept, thus this expression is not allowed in $\alpha\ell$
  - $(\neg C)^! = \Delta^! \setminus C^!$
  - If general complements are allowed, this results to the language $\alpha\ell C$
    - Naming convention: Start with $\alpha\ell$ and concatenate the short letters of all additional features…

13.1 Expanding $\alpha\ell$

- For description logics hold also the known **equivalences** of first order logics
  - $(C \cup D) \equiv \neg (\neg C \land \neg D)$
  - $\exists R.C \equiv \forall R.\neg C$
  - Thus, union and full existential quantification can be modeled by using the general complement and vice versa
  - Therefore, $\alpha\ell C$ has the same expressiveness as $\alpha\ell C U E$
    - Therefore, we will use $\alpha\ell C U E$ to refer also to $\alpha\ell C U E C$ or $\alpha\ell C U E U$

13.1 Expanding $\alpha\ell$

- **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 $\alpha\ell C$ is the least expressive common description language and has limited expressiveness
    - Additional features can expand $\alpha\ell C$

13.1 Expanding $\alpha\ell$

- Expansion $\mathcal{U}$: Union $C \cup D$
  - Allows to union two complex concepts
  - $(C \cup D)^! = C^! \cup D^!$
  - Results to $\alpha\ell C U$

- Expansion $\mathcal{E}$: Full existential quantification $\exists R.C$
  - In contrast to limited existential quantification in $\alpha\ell$, any concept is allowed in existential quantification
  - $\exists (R.T) = \{a \in \Delta^! | \exists b ((a, b) \in R \land b \in C^!}\}$
13.1 Expanding $\mathcal{AL}$

- **Expansion $\mathcal{N}$:** Number Restriction $\geq n R$
  - Comes in two flavors:
    - At-least-Restriction: $\geq n R.C$
    - At-most-Restriction: $\leq m R.C$
  - $\geq n R.C \prod \leq m R.C$: 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.
  - $(\geq n R.C) = \{a \in A | |\{(b | (a, b) \in R\}| \geq n\}$
  - $(\leq n R.C) = \{a \in A | |\{(b | (a, b) \in R\}| \leq n\}$

13.1 Description Language $\mathcal{AL}$

- 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. $\text{Person}(\text{Christoph}), \text{TeachingAssistant} \equiv \text{Person} \land \text{Teaches. Lecture} \equiv \text{Professor} \land \text{TeachingAssistant}(\text{Christoph})$
  - 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.

13.1 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 $C \not\models \bot$
      - Example of unsatisfiable concept: $\text{Student} \land \neg \text{Professor}$

13.1 Description Language $\mathcal{AL}$

- Terminological equivalence statements are called **definitorial**, if they are acyclic after expression optimization:
  - $C \equiv D \land E$
    - is acyclic and definitorial
  - $C \equiv D \lor E$
    - is cyclic and not definitorial
  - $C \equiv D \lor E \land \neg (C \cap \neg C)$
    - is cyclic and definitorial
  - Why? Query can be simplified to $C \equiv D \lor E \land (C \cap \neg C)$$
    - Thus, despite containing a cycle, it can be simplified to an acyclic expression

13.1 Complexity of $\mathcal{AL}$

- **Subsumption:** $\Sigma \models C \subseteq D$
  - Is the concept $C$ a sub-concept of $D$, i.e. is for every model $C \subseteq D$.
  - Example of a subsumed concept: $\text{Student} \subseteq \text{Person}$
- **Instance Checking:** $\Sigma \models C(a)$
  - Is an instance of $C$, i.e. is $C(a)$ satisfied in every model of $\Sigma$
- **Retrieval:** $[a | \Sigma \models C(a)]$
  - Returns all individuals being member of a certain concept
- **Realization:** $[C | \Sigma \models C(a)]$
  - Returns all concepts realizing a given individual
13.1 Complexity of \(\mathcal{ALC}\)

- Most of these problems are reducible (with overhead) to general satisfiability

**Reasoning complexity** for \(\mathcal{ALC}\) languages

<table>
<thead>
<tr>
<th>Language</th>
<th>(\equiv C \sqsubseteq D)</th>
<th>(\equiv 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>(\text{ExpTime})</td>
<td>(\text{ExpTime})</td>
</tr>
</tbody>
</table>

13.1 Description Language \(\mathcal{S}\)

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

13.1 Expanding \(\mathcal{S}\)

- **Expansion \(\mathcal{H}\): Role hierarchies \(R \sqsubseteq S\)**
  - Allows the construction of role hierarchies using the inclusion \(R \sqsubseteq S\)
  - \((R \sqsubseteq S) = R \sqsubseteq S\)
  - 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 \(\mathcal{H}\)

13.1 Complexity of \(\mathcal{ALC}\)

- Up to now, we have considered basic description languages based on \(\mathcal{AL}\)
  - They allow definition of facts via ABox statements and definition of complex concepts via 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 RBox
  - Construction of complex roles is not supported by \(\mathcal{AL}\)

13.1 Description Language \(\mathcal{S}\)

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

13.1 Expanding \(\mathcal{S}\)

- 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 \(R \sqcap S\), union \(R \cup S\), negation \(\neg\) and composition \(R \circ S\)
    - \((R \sqcap S) = R \sqcap S\)
    - \((R \cup S) = R \cup S\)
    - \((\neg R) = \neg R\)
    - \((R \circ S) = \{(a, c) \in A^2 \mid \exists b (a, b) \in R \land (b, c) \in S\}\)
  - Examples:
    - Atomic roles: attendsLecture, mother, parent
    - Complex roles: skipsLecture \(\equiv\) ~attendsLecture, grandmother \(\equiv\) parent \(\circ\) mother
13.1 Expanding $\mathcal{S}$

- Expansion $\mathcal{I}$: Role inverses $R^{-1}$
  - An inverse role is obtained when the arguments are swapped
  - $(R^{-1})^I = \{(b,a) \in \Delta^I \times \Delta^I \mid (a,b) \in R^I\}$
  - Example:
    - Assertion: teaches(Prof. Balke, KBS)
    - Complex role: isToughtBy $\equiv$ teaches
    - Results to: isToughtBy (KBS, Prof. Balke)

- Expansion $\mathcal{F}$: Functionality $\leq 1 R$
  - 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
    - $R(x, y) \Leftrightarrow f(x) = y$
    - Example:
      - age or mother are functional (you can only have one age and one mother)

13.1 Expanding $\mathcal{S}$

- Expansion $\mathcal{O}$: Nominals
  - Numerals are just a convenience feature for defining concepts as sets
  - The members of the concept are simply enumerated
    - Degrees $\equiv \{\text{BSC-Inf}, \text{MSC-Inf}, \text{BSC-WiInf}, \text{MSC-WiInf}\}$
    - Degrees $^I = \{\text{BSC-Inf}^I, \text{MSC-Inf}^I, \text{BSC-WiInf}^I, \text{MSC-WiInf}^I\}$

13.1 Language Summary

- Summary of description language features

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

13.1 Complexity

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

<table>
<thead>
<tr>
<th>Language</th>
<th>$\vdash C D$</th>
<th>$\vdash C(o)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{AL}$</td>
<td>P-comp.</td>
<td>P-comp.</td>
</tr>
<tr>
<td>$\mathcal{AL}$</td>
<td>NP-comp.</td>
<td>PSPACE-comp.</td>
</tr>
<tr>
<td>$\mathcal{AL}$</td>
<td>PSAlP-comp.</td>
<td>PSPACE-comp.</td>
</tr>
<tr>
<td>$\mathcal{AL}$ cyclic TBox</td>
<td>ExpTIME-comp.</td>
<td>ExpTIME-comp.</td>
</tr>
<tr>
<td>$\mathcal{AK}$</td>
<td>ExpTIME-comp.</td>
<td>ExpTIME-comp.</td>
</tr>
<tr>
<td>$\mathcal{SN}$</td>
<td>ExpTIME-comp.</td>
<td>ExpTIME-comp.</td>
</tr>
<tr>
<td>$\mathcal{SN}$</td>
<td>NExpTIME-comp.</td>
<td>NExpTIME-comp.</td>
</tr>
</tbody>
</table>

13.1 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/
13.2 DAML+OIL

- 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

13.2 DAML

- 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

13.2 OIL

- 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)
13.2 OIL

• **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

13.3 OWL

• 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/

• 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 $\mathcal{SH}(\mathcal{F}(D))$
    • And is thus ExpTIME-complete
    • Features:
      - Concepts (Complement, Union, Existential Quantification, Universal Quantification)
      - Roles (Transitive, Inverse, Functional)
      - Additional features for data types (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
13.3 OWL

- **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 $\mathcal{SHOIN}(D)$
  - And is thus \textit{NExpTIME}-complete
- **Features:**
  - Concepts (Complement, Union, Existential Quantification, Universal Quantification, Nominals)
  - Roles (Transitive, Hierarchy, Inverse, Functional, Unqualified restriction)
  - Additional features for data types ($D$)

- 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!

Detour

- Classes are defined using `owl:Class`
  - `owl:Class` is a subclass of `rdfs:Class`
- Disjointness ($C \sqcap \neg D \equiv \bot$) is defined using `owl:disjointWith`

Detour

- `owl:equivalentClass` defines equivalence of classes
  - i.e. $C \equiv D$
- `owl:Thing` is the most general class, which contains everything (corresponds to $T$)
- `owl:Nothing` is the empty class (corresponds to $\bot$)

Detour

- 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.
13.3 OWL

- 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>
```

13.3 OWL

- OWL datatypes properties make 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>
```

13.4 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

13.4 Protégé - TBox

Class Hierarchy

Annotation & Comments

Assertions

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

13.4 Protégé - RBox

Role Hierarchy

Special Features

Annotation & Comments

Assertions

13.4 Protégé - ABox

Individual list

Concept Assertions

Role Assertions
13.4 Protégé - Reasoner

- **Query Classes**
- **Query Result**
- **Query Type**

13.4 Protégé – OWL/XML

13 Next Lecture

- **The Wisdom of Crowds**
  - Folksonomies
  - Social software