Description Logic:
A Formal Foundation for
Ontology Languages and Tools

Ian Horrocks
<iian.horrocks@comlab.ox.ac.uk>
Information Systems Group
Oxford University Computing Laboratory
What Are Description Logics?
What Are Description Logics?

• Decidable fragments of First Order Logic

Thank you for listening

Any questions?
What Are Description Logics?

- A family of logic based Knowledge Representation formalisms
  - Originally descended from semantic networks and KL-ONE
  - Describe domain in terms of concepts (aka classes), roles (aka properties, relationships) and individuals

[Quillian, 1967]
What Are Description Logics?

• Modern DLs (after Baader et al) distinguished by:
  – Fully fledged logics with formal semantics
    • Decidable fragments of FOL (often contained in $C_2$)
    • Closely related to Propositional Modal/Dynamic Logics & Guarded Fragment
  – Computational properties well understood (worst case complexity)
  – Provision of inference services
    • Practical decision procedures (algorithms) for key problems
      (satisfiability, subsumption, query answering, etc)
    • Implemented systems (highly optimised)

• The basis for widely used ontology languages
Web Ontology Language OWL (2)

- **W3C recommendation(s)**
- Motivated by Semantic Web activity
  
  Add meaning to web content by annotating it with terms defined in ontologies
- Supported by tools and infrastructure
  
  - APIs (e.g., OWL API, Thea, OWLink)
  - Development environments (e.g., Protégé, Swoop, TopBraid Composer, Neon)
  - Reasoners & Information Systems (e.g., Pellet, Racer, HermiT, Quonto, …)
- Based on Description Logics (**SHOIN**/**SROIQ**)
DL Syntax

• **Signature**
  
  – **Concept** (aka class) names, e.g., Cat, Animal, Doctor
    • Equivalent to FOL unary predicates
  
  – **Role** (aka property) names, e.g., sits-on, hasParent, loves
    • Equivalent to FOL binary predicates
  
  – **Individual** names, e.g., Felix, John, Mary, Boston, Italy
    • Equivalent to FOL constants
DL Syntax

- Operators
  - Many kinds available, e.g.,
    - Standard FOL Boolean operators ($\land$, $\lor$, $\neg$)
    - Restricted form of quantifiers ($\exists$, $\forall$)
    - Counting ($\geq$, $\leq$, $=$)
    - ...
DL Syntax

• Concept expressions, e.g.,
  – Doctor ⊔ Lawyer
  – Rich ⊓ Happy
  – Cat ⊓ ∃ sits-on.Mat

• Equivalent to FOL formulae with one free variable
  – Doctor(x) ∨ Lawyer(x)
  – Rich(x) ∧ Happy(x)
  – ∃y.(Cat(x) ∧ sits-on(x, y))
DL Syntax

• Special concepts
  – \( T \) (aka top, Thing, most general concept)
  – \( \bot \) (aka bottom, Nothing, inconsistent concept)

used as abbreviations for
  – \( (A \sqcup \neg A) \) for any concept \( A \)
  – \( (A \sqcap \neg A) \) for any concept \( A \)
DL Syntax

• Role expressions, e.g.,
  – $\text{loves}^-$
  – $\text{hasParent} \circ \text{hasBrother}$

• Equivalent to FOL formulae with two free variables
  – $\text{loves}(y, x)$
  – $\exists z. (\text{hasParent}(x, z) \land \text{hasBrother}(z, y))$
DL Syntax

• “Schema” Axioms, e.g.,
  - Rich ⊆ ¬Poor  (concept inclusion)
  - Cat ∩ ∃sits-on.Mat ⊆ Happy  (concept inclusion)
  - BlackCat ≡ Cat ∩ ∃hasColour.Black  (concept equivalence)
  - sits-on ⊆ touches  (role inclusion)
  - Trans(part-of)  (transitivity)

• Equivalent to (particular form of) FOL sentence, e.g.,
  - ∀x.(Rich(x) → ¬Poor(x))
  - ∀x.(Cat(x) ∧ ∃y.(sits-on(x,y) ∧ Mat(y)) → Happy(x))
  - ∀x.(BlackCat(x) ↔ (Cat(x) ∧ ∃y.(hasColour(x,y) ∧ Black(y))))
  - ∀x,y.(sits-on(x,y) → touches(x,y))
  - ∀x,y,z.((sits-on(x,y) ∧ sits-on(y,z)) → sits-on(x,z))
DL Syntax

• “Data” Axioms (aka Assertions or Facts), e.g.,
  – BlackCat(Felix) (concept assertion)
  – Mat(Mat1) (concept assertion)
  – Sits-on(Felix,Mat1) (role assertion)

• Directly equivalent to FOL “ground facts”
  – Formulae with no variables
DL Syntax

- A set of axioms is called a **TBox**, e.g.:

  \{Doctor \sqsubseteq \text{Person}, \\
  \text{Parent} \equiv \text{Person} \sqcap \exists \text{hasChild.P}
  \text{erson}, \\
  \text{HappyParent} \equiv \text{Parent} \sqcap \forall \text{hasChild}
  \}

- A set of facts is called an **ABox**, e.g.:

  \{\text{HappyParent(John),} \\
  \text{hasChild(John,Mary)}\}

- A **Knowledge Base** (KB) is just a TBox plus an Abox

  - Often written $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

  **Note**

  Facts sometimes written

  \begin{align*}
  \text{John:HappyParent,} \\
  \text{John hasChild Mary,} \\
  \langle \text{John,Mary} \rangle: \text{hasChild}
  \end{align*}
The DL Family

• Many different DLs, often with “strange” names
  – E.g., $\textbf{EL}$, $\textbf{ALC}$, $\textbf{SHIQ}$

• Particular DL defined by:
  – Concept operators ($\cap$, $\cup$, $\neg$, $\exists$, $\forall$, etc.)
  – Role operators ($\cdot$, $\circ$, etc.)
  – Concept axioms ($\sqsubseteq$, $\equiv$, etc.)
  – Role axioms ($\sqsubseteq$, Trans, etc.)
The DL Family

• E.g., $\mathcal{EL}$ is a well known “sub-Boolean” DL
  – Concept operators: $\sqcap, \neg, \exists$
  – No role operators (only atomic roles)
  – Concept axioms: $\subseteq, \equiv$
  – No role axioms

• E.g.:

  $\text{Parent} \equiv \text{Person} \sqcap \exists \text{hasChild}.\text{Person}$
The DL Family

- **$\mathcal{ALC}$** is the smallest propositionally closed DL
  - Concept operators: $\sqcap$, $\sqcup$, $\neg$, $\exists$, $\forall$
  - No role operators (only atomic roles)
  - Concept axioms: $\sqsubseteq$, $\equiv$
  - No role axioms
- E.g.:

  $$\text{ProudParent} \equiv \text{Person} \sqcap \forall \text{hasChild.(Doctor} \sqcup \exists \text{hasChild.Doctor})$$
The DL Family

- $S$ used for $ALC$ extended with (role) transitivity axioms
- Additional letters indicate various extensions, e.g.:
  - $\mathcal{H}$ for role hierarchy (e.g., hasDaughter $\sqsubseteq$ hasChild)
  - $\mathcal{R}$ for role box (e.g., hasParent $\circ$ hasBrother $\sqsubseteq$ hasUncle)
  - $\mathcal{O}$ for nominals/singleton classes (e.g., \{Italy\})
  - $\mathcal{I}$ for inverse roles (e.g., isChildOf $\equiv$ hasChild$^-$)
  - $\mathcal{N}$ for number restrictions (e.g., $\geq$ hasChild, $\leq$ hasChild)
  - $\mathcal{Q}$ for qualified number restrictions (e.g., $\geq$ hasChild.\text{Doctor})
  - $\mathcal{F}$ for functional number restrictions (e.g., $\leq$ hasMother)

- E.g., $SHIQ = S +$ role hierarchy + inverse roles + QNRs
The DL Family

• Numerous other extensions have been investigated
  – Concrete domains (numbers, strings, etc)
  – DL-safe rules (Datalog-like rules)
  – Fixpoints
  – Role value maps
  – Additional role constructors (\(\cap, \cup, \neg, \circ, \text{id}, \ldots\))
  – Nary (i.e., predicates with arity >2)
  – Temporal
  – Fuzzy
  – Probabilistic
  – Non-monotonic
  – Higher-order
  – ...
DL Semantics

Via translation to FOL, or directly using FO model theory:

- **Interpretation function** $\mathcal{I}$
- **Interpretation domain** $\Delta^I$

**Individuals** $i^I \in \Delta^I$
- John
- Mary

**Concepts** $C^I \subseteq \Delta^I$
- Lawyer
- Doctor
- Vehicle

**Roles** $r^I \subseteq \Delta^I \times \Delta^I$
- hasChild
- owns
DL Semantics

- Interpretation function extends to concept expressions in the obvious(ish) way, e.g.:

\[
(C \cap D) = C \cap D \\
(C \cup D) = C \cup D \\
(\neg C) = \Delta \setminus C \\
\{x\} = \{x\} \\
(\exists R.C) = \{x | \exists y. \langle x, y \rangle \in R \land y \in C \} \\
(\forall R.C) = \{x | \forall y. \langle x, y \rangle \in R \Rightarrow y \in C \} \\
(\leq n R) = \{x | \#\{y | \langle x, y \rangle \in R\} \leq n\} \\
(\geq n R) = \{x | \#\{y | \langle x, y \rangle \in R\} \geq n\} 
\]
DL Semantics

• Given a model $M = \langle D, \cdot^I \rangle$
  
  - $M \models C \subseteq D$ iff $C^I \subseteq D^I$
  - $M \models C \equiv D$ iff $C^I = D^I$
  - $M \models C(a)$ iff $a^I \in C^I$
  - $M \models R(a, b)$ iff $\langle a^I, b^I \rangle \in R^I$
  - $M \models \langle \mathcal{T}, \mathcal{A} \rangle$ iff for every axiom $ax \in \mathcal{T} \cup \mathcal{A}$, $M \models ax$
DL Semantics

• Satisfiability and entailment
  – A KB $\mathcal{K}$ is satisfiable iff there exists a model $M$ s.t. $M \models \mathcal{K}$
  – A concept $C$ is satisfiable w.r.t. a KB $\mathcal{K}$ iff there exists a model $M = \langle D, \cdot^I \rangle$ s.t. $M \models \mathcal{K}$ and $C^I \neq \emptyset$
  – A KB $\mathcal{K}$ entails an axiom $ax$ (written $\mathcal{K} \models ax$) iff for every model $M$ of $\mathcal{K}$, $M \models ax$ (i.e., $M \models \mathcal{K}$ implies $M \models ax$)
DL Semantics

E.g.,

\[ T = \{ \text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists \text{hasChild}.\text{Person}, \]
\[ \text{HappyParent} \equiv \text{Parent} \sqcap \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \}
\]

\[ A = \{ \text{John:HappyParent}, \text{John hasChild Mary}, \text{John hasChild Sally}, \]
\[ \text{Mary:} \neg \text{Doctor}, \text{Mary hasChild Peter}, \text{Mary:}(\leq 1 \text{ hasChild}) \}
\]

- \( \mathcal{K} \models \text{John:Person} ? \)
- \( \mathcal{K} \models \text{Peter:Doctor} ? \)
- \( \mathcal{K} \models \text{Mary:HappyParent} ? \)
- What if we add “Mary hasChild Jane”?
  - \( \mathcal{K} \models \text{Peter = Jane} \)
- What if we add “HappyParent \equiv \text{Person} \sqcap \exists \text{hasChild}.\text{Doctor}”?
  - \( \mathcal{K} \models \text{HappyPerson} \sqsubseteq \text{Parent} \)
DL and FOL

• Most DLs are subsets of C2
  – But reduction to C2 may be (highly) non-trivial
    • Trans(R) naively reduces to $\forall x, y, z. R(x, y) \land R(y, z) \rightarrow R(x, z)$

• Why use DL instead of C2?
  – Syntax is succinct and convenient for KR applications
  – Syntactic conformance guarantees being inside C2
    • Even if reduction to C2 is non-obvious
  – Different combinations of constructors can be selected
    • To guarantee decidability
    • To reduce complexity
  – DL research has mapped out the decidability/complexity landscape in great detail
    • See Evgeny Zolin’s DL Complexity Analyzer
      http://www.cs.man.ac.uk/~ezolin/dl/
Complexity of reasoning in Description Logics

Base description logic: Atributive Language with Complements

\[
\text{ALC} := \bot \mid A \mid C \cap D \mid C \cup D \mid \exists C \mid \forall C
\]

Concept constructors:
- \( \mathcal{R} \) - functionality: \((\leq 1 \mathcal{R})\)
- \( \mathcal{N} \) - (unqualified) number restrictions: \((\geq n \mathcal{R}), (\leq n \mathcal{R})\)
- \( \mathcal{Q} \) - qualified number restrictions: \((\geq 1 \mathcal{R}, \mathcal{C}), (\leq 1 \mathcal{R}, \mathcal{C})\)
- \( \mathcal{O} \) - nominals: \(\{a\} \) or \(\{a_1, \ldots, a_n\}\) ("one-of" constructor)
- \( \mu \) - least fixpoint operator: \(\mu X.C\)
- \( R \preceq S \) - role-value-maps
- \( f = g \) - agreement of functional role chains ("same-as")

Role constructors:
- \( \mathcal{I} \) - role inverses: \(\mathcal{I}^{-\top}\)
- \( \land \) - role intersection: \(\mathcal{R} \land \mathcal{S}\)
- \( \cup \) - role union: \(\mathcal{R} \cup \mathcal{S}\)
- \( \neg \) - role complement: \(\mathcal{R}^\complement\)
- \( \circ \) - role chain (composition): \(\mathcal{R} \circ \mathcal{S}\)
- \( * \) - reflexive-transitive closure: \(\mathcal{R}^*\)
- \( id \) - concept identity: \(id(C)\)
- \( \forall \) - complex roles: \(\forall \mathcal{R} \) in number restrictions

TBox is internalized in extensions of ALC\(\text{TIO}\), see [75, Lemma 4.12], [54, p.3]
- Empty TBox
- Acyclic TBox (\(\text{A} \equiv C, A \) is a concept name; no cycles)
- General TBox (\(\forall \mathcal{R} \) for arbitrary concepts \(C\) and \(D\))

Role axioms (RBox):
- \( S \) - Role transitivity: Trans(\(S\))
- \( R \) - Role hierarchy: \(R \subseteq S\)
- \( R \) - Complex role inclusions: \(\mathcal{R} \subseteq R, \mathcal{R} \subseteq S\)
- \( s \) - some additional features

Complexity of reasoning problems

<table>
<thead>
<tr>
<th>Reasoning problem</th>
<th>Complexity</th>
<th>Comments and references</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept satisfiability</td>
<td>NExpTime-complete</td>
<td>Hardness of even ALC(\text{TIO}) is proved in [75, Corollary 4.13]. In that paper, the result is formulated for ALC(\text{TIO}), but only number restrictions of the form ((\leq 1 \mathcal{R})) are used in the proof. A different proof of the NExpTime-hardness for ALC(\text{TIO}) is given in [54] (even with 1 nominal, and role inverses not used in number restrictions). Upper bound for SHIQ is proved in [77, Corollary 6.31] with numbers coded in unary (for binary coding, the upper bound remains an open problem for all logics in between ALC(\text{TIO}) and SHIQ). Important: in number restrictions, only simple roles (i.e. which are neither transitive nor have a transitive subrole) are allowed; otherwise we gain undecidability even in SHIQ, see [46]. Remark: recently [47] it was observed that, in many cases, one can use transitive roles in number restrictions - and still have a decidable logic! So the above notion of a simple role could be substantially extended.</td>
</tr>
<tr>
<td>ABox consistency</td>
<td>NExpTime-complete By reduction to concept satisfiability problem in presence of nominals shown in [69, Theorem 3.7].</td>
<td></td>
</tr>
</tbody>
</table>
Complexity Measures

- **Taxonomic** complexity
  Measured w.r.t. total size of “schema” axioms

- **Data** complexity
  Measured w.r.t. total size of “data” facts

- **Query** complexity
  Measured w.r.t. size of query

- **Combined** complexity
  Measured w.r.t. total size of KB (plus query if appropriate)
Complexity Classes

- **LogSpace, PTime, NP, PSpace, ExpTime, etc**
  - worst case for a given problem w.r.t. a given parameter
  - X-hard means at-least this hard (could be harder); in X means no harder than this (could be easier); X-complete means both hard and in, i.e., exactly this hard
    - e.g., *SROIQ* KB satisfiability is 2NExpTime-complete w.r.t. combined complexity and NP-hard w.r.t. data complexity

- **Note** that:
  - this is for the **worst case**, not a **typical case**
  - complexity of **problem** means we can never devise a more efficient (in the worst case) algorithm
  - complexity of **algorithm** may, however, be even higher (in the worst case)
DLs and Ontology Languages
DLs and Ontology Languages

- W3C's OWL 2 (like OWL, DAML+OIL & OIL) based on DL
  - OWL 2 based on SROIQ, i.e., ALC extended with transitive roles, a role box nominals, inverse roles and qualified number restrictions
    - OWL 2 EL based on EL
    - OWL 2 QL based on DL-Lite
    - OWL 2 EL based on DLP
  - OWL was based on SHOEIN
    - only simple role hierarchy, and unqualified NRs
## Class/Concept Constructors

<table>
<thead>
<tr>
<th>OWL Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \sqcup \ldots \sqcup {x_n}$</td>
<td>${\text{john}} \sqcup {\text{mary}}$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Doctor</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lawyer</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq n P$</td>
<td>$\leq 1$ hasChild</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n P$</td>
<td>$\geq 2$ hasChild</td>
</tr>
</tbody>
</table>
### Ontology Axioms

<table>
<thead>
<tr>
<th>OWL Syntax</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>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>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor$^+$ $\sqsubseteq$ ancestor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OWL Syntax</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>$a : C$</td>
<td>John : Happy-Father</td>
</tr>
<tr>
<td>property</td>
<td>$\langle a, b \rangle : R$</td>
<td>$\langle$John, Mary$\rangle :$ has-child</td>
</tr>
</tbody>
</table>

- An **Ontology** is *usually* considered to be a TBox
  - but an **OWL** ontology is a mixed set of TBox and ABox axioms
Other OWL Features

• XSD datatypes and (in OWL 2) facets, e.g.,
  – integer, string and (in OWL 2) real, float, decimal, datetime, …
  – minExclusive, maxExclusive, length, …
  – PropertyAssertion( hasAge Meg "17"^^xsd:integer )
  – DatatypeRestriction( xsd:integer xsd:minInclusive "5"^^xsd:integer xsd:maxExclusive "10"^^xsd:integer )

These are equivalent to (a limited form of) DL concrete domains

• Keys
  – E.g., HasKey(Vehicle Country LicensePlate)
    • Country + License Plate is a unique identifier for vehicles

This is equivalent to (a limited form of) DL safe rules
E.g., Person \( \sqcap \forall \text{hasChild.}(\text{Doctor} \sqcup \exists \text{hasChild.} \text{Doctor}) \):

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType="collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType="collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```
Complexity/Scalability

- From the complexity navigator we can see that:
  - OWL (aka SHOIN) is \textit{NExpTime-complete}
  - OWL Lite (aka SHIF) is \textit{ExpTime-complete} (oops!)
  - OWL 2 (aka SROIQ) is \textit{2NExpTime-complete}
  - OWL 2 EL (aka \textit{\mathcal{EL}}) is \textit{PTIME-complete} (robustly scalable)
  - OWL 2 RL (aka \textit{DLP}) is \textit{PTIME-complete} (robustly scalable)
    - And implementable using rule based technologies
e.g., rule-extended DBs
  - OWL 2 QL (aka DL-Lite) is in \textit{AC}^0 \text{ w.r.t. size of data}
    - same as DB query answering -- nice!
Why (Description) Logic?

- OWL exploits results of 20+ years of DL research
  - Well defined (model theoretic) **semantics**

<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>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
<td>$C_1(x) \wedge \ldots \wedge C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
<td>$C_1(x) \vee \ldots \vee C_n(x)$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
<td>$\neg C(x)$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \sqcup \ldots \sqcup {x_n}$</td>
<td>${john} \sqcup {mary}$</td>
<td>$x = x_1 \vee \ldots \vee x = x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Doctor</td>
<td>$\forall y.P(x, y) \rightarrow C(y)$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lawyer</td>
<td>$\exists y.P(x, y) \wedge C(y)$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq nP$</td>
<td>$\leq 1$ hasChild</td>
<td>$\exists y.P(x, y)$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq nP$</td>
<td>$\geq 2$ hasChild</td>
<td>$\exists y.P(x, y)$</td>
</tr>
</tbody>
</table>
Why (Description) Logic?

• OWL exploits results of 20+ years of DL research
  – Well defined (model theoretic) **semantics**
  – **Formal properties** well understood (complexity, decidability)

I can’t find an efficient algorithm, but neither can all these famous people.

Why (Description) Logic?

• OWL exploits results of 20+ years of DL research
  – Well defined (model theoretic) **semantics**
  – **Formal properties** well understood (complexity, decidability)
  – Known reasoning algorithms
Why (Description) Logic?

• OWL exploits results of 20+ years of DL research
  – Well defined (model theoretic) **semantics**
  – **Formal properties** well understood (complexity, decidability)
  – Known **reasoning algorithms**
  – **Scalability** demonstrated by **implemented systems**
Tools, Tools, Tools

Major benefit of OWL has been huge increase in range and sophistication of tools and infrastructure:
Tools, Tools, Tools

**Major benefit** of OWL has been huge increase in range and sophistication of tools and infrastructure:

- Editors/development environments
Tools, Tools, Tools

**Major benefit** of OWL has been huge increase in range and sophistication of tools and infrastructure:

- Editors/development environments
- Reasoners
Tools, Tools, Tools

**Major benefit** of OWL has been huge increase in range and sophistication of tools and infrastructure:

- Editors/development environments
- Reasoners
- Explanation, justification and pinpointing
Tools, Tools, Tools

Major benefit of OWL has been huge increase in range and sophistication of tools and infrastructure:

- Editors/development environments
- Reasoners
- Explanation, justification and pinpointing
- Integration and modularisation
Tools, Tools, Tools

**Major benefit** of OWL has been huge increase in range and sophistication of tools and infrastructure:

- Editors/development environments
- Reasoners
- Explanation, justification and pinpointing
- Integration and modularisation
- APIs, in particular the **OWL API**
OWL 2 Profiles and Reasoning

OWL 2 “DL” (full language)

• Standard technique is refutation via model construction:
  \[ \mathcal{O} \models Q(x) \text{ iff } \mathcal{O} \cup \{\neg Q(x)\} \models \bot \]
  – Try to refute by constructing model of \( \mathcal{O} \cup \{\neg Q(x)\} \)
  – Model construction very similar to DB CHASE techniques

• E.g., HermiT, FaCT++, Pellet, ...

• Scalability issues for query answering (number and size of models)
  – but many optimisations are possible
OWL 2 Profiles and Reasoning

**OWL 2 EL**

- A (near maximal) fragment of OWL 2 such that
  - Satisfiability checking is in PTime (**PTime-Complete**)
  - Data complexity of query answering also PTime-Complete

- Based on \( \mathcal{EL} \) family of description logics

- Can exploit “saturation” reasoning techniques
  - Deductive inference rules used to materialise all relevant schema axioms (e.g., atomic subsumption axioms)

- E.g., CB, CEL, Snorocket, ...
OWL 2 Profiles and Reasoning

OWL 2 QL

• A (near maximal) fragment of OWL 2 such that
  – Data complexity of conjunctive query answering in $\text{AC}^0$
• Based on DL-Lite family of description logics
• Can exploit query rewriting based reasoning technique
  – Ontology axioms treated as backward chaining rules and used to expand query
  – Data storage and query evaluation can be delegated to standard RDBMS
• E.g., QuOnto, Oracle
OWL 2 Profiles and Reasoning

**OWL 2 RL**

- A (near maximal) fragment of OWL 2 such that
  - Reasoning can be implemented via forward chaining rule engines

- Can exploit *materialisation* based reasoning technique
  - Ontology plus standard set of forward chaining inference rules used to materialise all relevant facts (data)
  - Can be implemented on top of standard RDBMS with rule engine

- E.g., Jena, Sesame, Owlim, **Oracle**
OWL 2 Profiles and Reasoning

Oracle Database Semantic Technologies

- Scalable, secure, and standard-compliant platform for storage, inference, and querying of semantic data
  - RDF/RDFS/OWL/SKOS/SPARQL
  - OWL RL and EL (SNOMED support)
  - semantic document indexing framework that works with 3rd party entity extraction engines
  - set of easy to use Java programming APIs (Jena Adapter/ Sesame Adapter)
Motivating Applications

- OWL playing **key role** in increasing number & range of applications
  - eScience

3D Analysis of Patterns of Gene Expression

Ontology of Zebrafish Developmental Anatomy

Integration of Heterogeneous gene expression data
Motivating Applications

• OWL playing **key role** in increasing number & range of applications
  – eScience, geography
Motivating Applications

- OWL playing **key role** in increasing number & range of applications
  - eScience, geography, engineering,
Motivating Applications

- OWL playing **key role** in increasing number & range of applications
  - eScience, geography, engineering, defence, …

Experience of OWL in use has identified restrictions:
- on expressivity
- on scalability

These restrictions are problematic in some applications.

Research has now shown how some restrictions can be overcome.
- W3C OWL WG is updating OWL accordingly
Motivating Applications: HCLS

• **OBO foundry** includes more than 100 biological and biomedical ontologies

• **Siemens** “actively building OWL based clinical solutions”

• OWL tools used to find and repair critical errors in ontology used at **Columbia Presbyterian**

• **SNOMED-CT** (Clinical Terms) ontology
  – used in healthcare systems of more than 15 countries, including Australia, Canada, Denmark, Spain, Sweden and the UK
  – also used by major US providers, e.g., Kaiser Permanente
  – ontology provides common vocabulary for recording clinical data
Motivating Applications: BBC
Motivating Applications: BBC
Motivating Applications: BBC
Ontology -v- Database
Obvious Database Analogy

- Ontology axioms analogous to DB schema
  - Schema describes structure of and constraints on data
- Ontology facts analogous to DB data
  - Instantiates schema
    - Consistent with schema constraints
- But there are also important differences…
Obvious Database Analogy

Database:

- Closed world assumption (CWA)
  - Missing information treated as false
- Unique name assumption (UNA)
  - Each individual has a single, unique name
- Schema behaves as constraints on structure of data
  - Define legal database states

Ontology:

- Open world assumption (OWA)
  - Missing information treated as unknown
- No UNA
  - Individuals may have more than one name
- Ontology axioms behave like implications (inference rules)
  - Entail implicit information
Database -v- Ontology

E.g., given the following ontology/schema:

- HogwartsStudent ≡ Student □ ∃ attendsSchool.Hogwarts
- HogwartsStudent ⊑ ∀ hasPet.(Owl or Cat or Toad)
- hasPet ≡ isPetOf^−1 (i.e., hasPet inverse of isPetOf)
- ∃ hasPet. ⊑ Human (i.e., domain of hasPet is Human)
- Phoenix ⊑ ∀ isPetOf.Wizard (i.e., only Wizards have Phoenix pets)
- Muggle ⊑ ¬ Wizard (i.e., Muggles and Wizards are disjoint)
Database -v- Ontology

And the following facts/data:

- HarryPotter: Wizard
- DracoMalfoy: Wizard
- HarryPotter hasFriend RonWeasley
- HarryPotter hasFriend HermioneGranger
- HarryPotter hasPet Hedwig

Query: Is Draco Malfoy a friend of HarryPotter?

- DB: No
- Ontology: Don’t Know

OWA (didn’t say Draco was not Harry’s friend)
Database -v- Ontology

And the following facts/data:

- HarryPotter: Wizard
- DracoMalfoy: Wizard
- HarryPotter hasFriend RonWeasley
- HarryPotter hasFriend HermioneGranger
- HarryPotter hasPet Hedwig

Query: How many friends does Harry Potter have?

- DB: 2
- Ontology: at least 1

No UNA (Ron and Hermione may be 2 names for same person)
Database -v- Ontology

And the following facts/data:

- HarryPotter: Wizard
- DracoMalfoy: Wizard
- HarryPotter hasFriend RonWeasley
- HarryPotter hasFriend HermioneGranger
- HarryPotter hasPet Hedwig

\[ \text{RonWeasley } \neq \text{ HermioneGranger} \]

Query: How many friends does Harry Potter have?

- DB: 2
- Ontology: at least 2

OWA (Harry may have more friends we didn’t mention yet)
Database -v- Ontology

And the following facts/data:

- HarryPotter: Wizard
- DracoMalfoy: Wizard
- HarryPotter hasFriend RonWeasley
- HarryPotter hasFriend HermioneGranger
- HarryPotter hasPet Hedwig
- RonWeasley ≠ HermioneGranger

⇒ HarryPotter: ∀hasFriend.{RonWeasley} ⊕ {HermioneGranger}

Query: How many friends does Harry Potter have?

- DB: 2
- Ontology: 2!
Database -v- Ontology

Inserting new facts/data:

- Dumbledore: Wizard
- Fawkes: Phoenix
- Fawkes isPetOf Dumbledore

What is the response from DBMS?
- Update rejected: constraint violation

  Domain of hasPet is Human; Dumbledore is not Human (CWA)

What is the response from Ontology reasoner?

- Infer that Dumbledore is Human (domain restriction)
- Also infer that Dumbledore is a Wizard (only a Wizard can have a phoenix as a pet)
DB Query Answering

• Schema plays no role
  – Data must explicitly satisfy schema constraints

• Query answering amounts to model checking
  – I.e., a “look-up” against the data

• Can be very efficiently implemented
  – Worst case complexity is low (logspace) w.r.t. size of data
Ontology Query Answering

• Ontology axioms play a powerful and crucial role
  – Answer may include implicitly derived facts
  – Can answer conceptual as well as extensional queries
    • E.g., Can a Muggle have a Phoenix for a pet?
• Query answering amounts to theorem proving
  – I.e., logical entailment
• May have very high worst case complexity
  – E.g., for OWL, NP-hard w.r.t. size of data
    (upper bound is an open problem)
  – Implementations may still behave well in typical cases
  – Fragments/profiles may have much better complexity
Ontology Based Information Systems

• Analogous to relational database management systems
  – Ontology ≈ schema; instances ≈ data

• Some important (dis)advantages
  + (Relatively) easy to maintain and update schema
    • Schema plus data are integrated in a logical theory
  + Query answers reflect both schema and data
  + Can deal with incomplete information
  + Able to answer both intensional and extensional queries
  – Semantics can seem counter-intuitive, particularly w.r.t. data
    • Open -v- closed world; axioms -v- constraints
  – Query answering (logical entailment) may be much more difficult
    • Can lead to scalability problems with expressive logics
Ontology Based Information Systems

• Analogous to relational database management systems
  – Ontology ≈ schema

• Some important advantages
  + (Relatively) easy to maintain and update schema
  + Schema plus data are integrated in a logical theory
  + Query answers reflect both schema and data
  + Can deal with incomplete information
  + Able to answer both intensional and extensional queries
  – Semantics can seem counter-intuitive, particularly w.r.t. data
  – Open vs. closed world; axioms vs. constraints
  – Query answering (logical entailment) may be much more difficult
  – Can lead to scalability problems with expressive logics
THE END?
Ongoing Research

• Query answering
  – [Kontchakov et al], [Konev et al], [Baader et al]

• Diagnosis and repair
  – [Horridge et al], [Peñaloza et al]

• Extensions
  – [Motik et al], [Artale et al]

• Optimisation/Profiles
  – [Kazakov], [Glimm et al], [Faddoul et al], [Savo et al]

• ...
Ongoing Standardisation Efforts

- Standardised query language
  - SPARQL standard for RDF
  - Currently being extended for OWL, see [http://www.w3.org/TR/sparql11-entailment/](http://www.w3.org/TR/sparql11-entailment/)

- RDF
  - Revision currently being considered, see [http://www.w3.org/2009/12/rdf-ws/](http://www.w3.org/2009/12/rdf-ws/)
Thank you for listening
Thank you for listening

Any questions?