Adding Semantic to Web Data and Services
Part 2 – DL Knowledge base Reasoning

Doctoral School, St Etienne January 2009
Alain Léger FT R&D Orange Labs Research
DR Knowledge Processing (KRR)
Manager Industry Area IST NoEs OntoWeb et Knowledgeweb (2000 -2007)
Associated DR CNRS Lyon I - LIRIS

Plan Cours 1 (5 janv 09 13:30 - 17:15 / 6 janv 09 8:00 - 11:45)

• Why adding semantics to the Web ? (1h30)
  - Introduction
  - Take Away and References

• Foundations of Semantic Web (2h15)
  - Introduction to Description Logics
  - Standards Inferences and Tableau

• From XML, RDF to OWL (2h45)
  - XML, RDF, RDF-S
  - OWL

• Applications and Roadmap (1h)
  - Application Scenarios
  - Visions prospectives et verrous technologiques
Ontologies

Ontology the key ingredient: Origins and History

a philosophical discipline—a branch of philosophy that
deals with the nature and the organisation of reality

• Science of Being (Aristotle, Metaphysics, IV, 1)

• Tries to answer the questions:
  What characterizes being?
  Eventually, what is being?

• How should things be classified?
Classification: An Old Problem

Les représentations du Système figuré : novembre 1750 et juin 1751,
publié dans l’Encyclopédie ou Dictionnaire raisonné des sciences, des arts et métiers,
Par une Société de gens de Lettres ... Tome I, 1751

Machine Intelligence and Turing Test

The Phaistos Disc (1700 BC) – undeciphered -
can perhaps be thought of as the earliest typewritten work
discovered on the 3rd of July 1908 by L. Pernier,
during an excavation he supervised
at the Minoan palace of Phaistos in Southern Crete
Ontology in Computer Science

- An ontology is an engineering artefact consisting of:
  - A vocabulary used to describe (a particular view of) some domain
  - An explicit specification of the intended meaning of the vocabulary.
    - almost always includes how concepts should be classified
  - Constraints capturing additional knowledge about the domain

- Ideally, an ontology should:
  - Capture a shared understanding of a domain of interest
  - Provide a formal and machine manipulable model of the domain

What is a concept?

- Concepts or “classes”:
  - Are in general language independent (the words ‘university’ and ‘ollscoil’ denote the same concept)
  - Are mental or logical representations of reality
  - Are related to other concepts
  - Do not need symbols but hold them for means of communication

- A concept has:
  - Intension, i.e. meaning
  - Extension, i.e. a set of objects that the concept refers to

- Ontology is mainly concerned with intension
Components of an ontology

- **Concepts**
  - Cat
  - Dog
- **Properties**
  - Length
  - Age
- **Constraints**
  - Cardinality is at least 1
  - Maximum value is 300
- **Axioms**
  - Cows are larger than dogs
  - Cats cannot eat only vegetation
- **Relationships**
  - Is a
  - Part of

Example Ontology (1)

- **Vocabulary and meaning ("definitions")**
  - *Elephant* is a concept whose members are a kind of animal
  - *Herbivore* is a concept whose members are exactly those animals who eat only plants or parts of plants
  - *Adult_Elephant* is a concept whose members are exactly those elephants whose age is greater than 20 years

- **Background knowledge/constraints on the domain ("general axioms")**
  - *Adult_Elephants* weigh at least 2,000 kg
  - All *Elephants* are either *African_Elephants* or *Indian_Elephants*
  - No individual can be both a *Herbivore* and a *Carnivore*
Implementing or creating ontologies

- Implementation consists in defining all the ontology components through an ontology definition language

- Generally in two stages:
  - Informal stage:
    - Ontology is sketched out using either natural language descriptions or some diagram technique
  - Formal stage:
    - Ontology is encoded in a formal knowledge representation language, that is machine computable

- Different tools (e.g., Protégé) may help in the implementation
  - [http://protege.stanford.edu/overview/protege-owl.html](http://protege.stanford.edu/overview/protege-owl.html)
Example Ontology (Editor Protégé)

Example Ontology (Editor OilEd)
Where are ontologies used?

- e-Science, e.g., Bioinformatics
  - The Gene Ontology
  - The Protein Ontology (MGED)
  - "in silico" investigations relating theory and data
- Medicine
  - Terminologies
- Databases
  - Integration
  - Query answering
- User interfaces
- Linguistics
- The Semantic Web

Ontology in a nutshell

- "Ontology is an explicit conceptualisation, formal and shared" [Gruber 95] [Borst 97]
- Thus, an ontology describes a formal specification of a certain domain:
  - Shared understanding of a domain of interest
  - A formal and machine manipulable model of a domain of interest

Aristotle ten categories

- Substance. E.g., individual man.
- Quantity. E.g., two cubits.
- Quality. E.g., white.
- Relation. E.g., double.
- Location. E.g., in the market.
- Time. E.g., today.
- Position. E.g., sitting.
- Possession. E.g., wearing shoes.
- Doing. E.g., cutting.
- Undergoing. E.g., being cut.

METAPHYSIC: ONTOLOGY and EPISTEMOLOGY

Ontology

"discourse on the being" fundamental questions: "what does exist?"; "what is the content of the reality?"; "how does the reality work?"; "what are the origins of the reality?"; "what is the future of the reality?"

Epistemology

"discourse on the knowledge". Central question is: "how do we know?"

LOGIC

It's goal is to infer truths from sound reasoning
Ontology Technology and Process in a nutshell

To make the Semantic Web working we need:

- **Ontology Languages:**
  - expressivity
  - reasoning support
  - web compliance

- **Ontology Reasoning:**
  - large scale knowledge handling
  - fault-tolerant
  - stable & scalable inference machines

- **Ontology Management Techniques:**
  - editing and browsing
  - storage and retrieval
  - versioning and evolution Support

- **Ontology Integration Techniques:**
  - ontology mapping, alignment, merging
  - semantic interoperability determination

- and ... Applications

---

Introduction to Description Logics
What Are Description Logics?

- A family of logic based Knowledge Representation formalisms
  - Descendants of semantic networks (Brachman 78) and KL-ONE (Brachman & Schmolze 85)
  - Describe domain in terms of concepts (classes), roles (properties, relationships) and individuals

- Characterized by:
  - Formal semantics (typically model theoretic)
    - Decidable fragments of FOL (First Order Logics)
    - Closely related to Propositional Modal (nécessaire, contingent, possible)
    - Dynamic Logics (comportement dynamique)
    - Closely related to Guarded Fragment (Decidable Fragment of FOL)
  - Provision of inference services
    - Decision procedures for key problems (satisfiability, subsumption, etc)
    - Implemented systems (highly optimised !)

DL Basics

- **Concepts** (unary predicates/formulae with one free variable)
  - e.g., Person, Father, Mother
- **Roles** (binary predicates/formulae with two free variables)
  - e.g., hasChild, hasHusband
- **Individual** names (constants)
  - e.g., Alice, Bob, Cindy
- **Axioms** (axiomatic relations between concepts or roles)
  - e.g., Female ⊆ Person
  - e.g. HappyFather ⊆ Father Π ≥1 hasChild.Woman Π ≥1 hasChild.Man
- **Operators** (for forming concepts and roles)
  - And(Π), Or(U), Not (¬)
  - Universal qualifier (∀), Existent qualifier(∃)
  - Number restriction : ≤, ≥, =
  - Inverse role (¬) : hasParent = hasChild
  - transitive role (∗) : hasBrother(Bob, David), hasBrother(David, Mack) => hasBrother(Bob, Mack)
  - Role hierarchy : hasMother ⊆ hasParent
The DL Family (Notation)

- Given DL defined by set of concept and role forming operators
- Smallest propositionally closed DL is $\mathcal{ALC}$ (equiv modal $K(m)$)
  - Concepts constructed using $\cap, \cup, \neg, \exists$ and $\forall$
- $S$ often used for $\mathcal{ALC}$ with transitive roles ($R_+$)
- Additional letters indicate other extension, e.g.:
  - $\mathcal{H}$ for role inclusion axioms (role hierarchy)
  - $\mathcal{O}$ for nominals (singleton classes, written $\{x\}$)
  - $\mathcal{I}$ for inverse roles
  - $\mathcal{N}$ for number restrictions (of form $\leq nR, \geq nR$)
  - $\mathcal{Q}$ for qualified number restrictions (of form $\leq nR.C, \geq nR.C$)
- E.g., $\mathcal{ALC} + R_+ + \text{role hierarchy} + \text{inverse roles} + \text{QNR} = \text{SHIQ}$
- Have been extended in many directions
  - Concrete domains, epistemic, $n$-ary, fuzzy, ...

---

The DL Family (a very few part)

<table>
<thead>
<tr>
<th>Concept constructors</th>
<th>$\mathcal{AL}$</th>
<th>$\mathcal{ALN}$</th>
<th>$\mathcal{ALC}$</th>
<th>$\mathcal{ALEN}$</th>
<th>$\mathcal{ALCQ}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\neg A$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\neg C$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$C \cap D$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$C \cup D$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\forall r.C$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\exists r.C$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\geq n r$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\leq n r$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
</tbody>
</table>
Concept Description

- Représenter des concepts
  - Concepts atomiques et rôles
  - Constructeurs de concept
  - Exemple : classe des mères, i.e. des personnes de sexe féminin ayant au moins un enfant qui est lui-même une personne.

  \[
  \text{Mère} \equiv \text{Personne} \cap \text{Féminin} \cap \exists \text{Enfant}.\text{Personne}
  \]

- Une terminologie (ou Tbox) = \{définitions de concepts\}
- Une logique de description = \{constructeurs\}
- Sémantique :
  - Notion d’interprétation issue de la théorie des modèles
  - Concept = ensemble d’\{individus\} du dom. Interprétation

DL Semantics

- Semantics defined by interpretations
- An interpretation \( I = (\Delta^I, .^I) \), where
  - \( \Delta^I \) is the domain (a non-empty set)
  - \( .^I \) is an interpretation function that maps:
    - Concept (class) name \( A \rightarrow \text{subset} \ A^I \) of \( \Delta^I \)
    - Role (property) name \( R \rightarrow \text{binary relation} \ R^I \) over \( \Delta^I \)
    - Individual name \( i \rightarrow i^I \) element of \( \Delta^I \)
DL Semantics (2)

- Interpretation function \( \mathcal{I} \) extends to concept (and role) expressions in the obvious way, e.g.:

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

Interpretation Example (homework)

\[\Delta = \{v, w, x, y, z\}\]
\[A^I = \{v, w, x\}\]
\[B^I = \{x, y\}\]
\[R^I = \{(v, w), (v, x), (y, x), (x, z)\}\]

- \(\neg B = \)
- \(A \cap B = \)
- \(\neg A \cup B = \)
- \(\exists R.B = \)
- \(\forall R.B = \)
- \(\exists R.(\exists R.A) = \)
- \(\exists R.\neg (A \cup B) = \)
- \(\leq 1 R.A = \)
- \(\geq 1 R.A = \)
Base de Connaissance (KB) : Architecture, syntaxe, sémantique

- Un langage $\mathcal{L}$-KR étant donné, une Base de connaissance $\mathcal{K}$ dans $\mathcal{L}$ est définie par $\mathcal{K} = (\mathcal{T}, \mathcal{A})$
  - $\mathcal{T}(\text{Tbox})$ est un ensemble de définitions et d’axiomes (en $\mathcal{L}$) :
    - $C \sqsubseteq D$ (concept inclusion)
    - $C \equiv D$ (concept equivalence)
    - $R \sqsubseteq S$ (role inclusion)
    - ... + autres constructeurs selon $\mathcal{L}$-KR
  - $\mathcal{A}(\text{Abox})$ est un ensemble d’assertions (en $\mathcal{L}$) :
    - $x \in D$ (concept instantiation)
    - $\left(x, y\right) \in R$ (role instantiation)

- La sémantique est donnée par interprétation $I = (\Delta^I, \cdot^I)$
  - $\Delta^I$ est le domaine (non vide)
  - $\cdot^I$ est une fonction d'interprétation qui fait correspondre :
    - Concept (classe) nom $\Lambda \rightarrow$ Sous-ensemble $\Lambda^I$ of $\Delta^I$
    - Role (propriété) nom $R \rightarrow$ Relation binaire $R^I$ sur $\Delta^I$
    - Individu (instance) nom $i \rightarrow$ $i^I$ element of $\Delta^I$

Knowledge Base Semantics

- An interpretation $I$ satisfies (models) an axiom $A$ ($I \vdash A$):
  - $I \vdash C \sqsubseteq D$ iff $C^I \subseteq D^I$
  - $I \vdash C \equiv D$ iff $C^I = D^I$
  - $I \vdash R \sqsubseteq S$ iff $R^I \subseteq S^I$
  - $I \vdash R \equiv S$ iff $R^I = S^I$
  - $I \vdash R^+ \subseteq R$ iff $(R^I)^+ \subseteq R^I$
  - $I \vdash x \in D$ iff $x^I \in D^I$
  - $I \vdash (x, y)^I \in R$ iff $(x^I, y^I) \in R^I$
- $I$ satisfies a Tbox $\mathcal{T}$ ($I \models \mathcal{T}$) iff $I$ satisfies every axiom $T$ in $\mathcal{T}$
- $I$ satisfies an Abox $\mathcal{A}$ ($I \models \mathcal{A}$) iff $I$ satisfies every axiom $A$ in $\mathcal{A}$
- $I$ satisfies an KB $\mathcal{K}$ ($I \models \mathcal{K}$) iff $I$ satisfies both $\mathcal{T}$ and $\mathcal{A}$
Multiple Models - v - Single Model

- DL KB doesn’t define a single model, it is a set of constraints that define a set of possible models
  - No constraints (empty KB) means any model is possible
  - More constraints means fewer models
  - Too many constraints may mean no possible model (inconsistent KB)
- In contrast, DBs (and frame/rule KR systems) make assumptions such that DB/KB defines a single model
  - Unique name assumption
    • Different names always interpreted as different individuals
  - Closed world assumption
    • Domain consists only of individuals named in the DB/KB
  - Minimal models
    • Extensions are as small as possible

Example of Multiple Models

\[
\begin{align*}
\text{KB} &= \{\} \\
\text{KB} &= \{a:C, b:D, c:C, d:E\} \\
\text{KB} &= \{a:C, b:D, c:C, d:E, b:C\} \\
\text{KB} &= \{a:C, b:D, c:C, d:E, b:C, D \subseteq C\} \\
\end{align*}
\]

\[
\begin{align*}
\Delta_1 &= \{v, w, x, y, z\} \\
\Delta &= \{v, w, x, y, z\} \\
C' &= \{w, y\} \\
E' &= \{z\} \\
a = v &\quad b' = x \\
c = x &\quad d = y \\
\Delta_2 &= \{v, w, x, y, z\} \\
\Delta &= \{v, w, x, y, z\} \\
C' &= \{v, w, y\} \\
E' &= \{z\} \\
a = v &\quad b' = x \\
c = w &\quad d' = z \\
\Delta_3 &= \{v, w, x, y, z\} \\
\Delta &= \{v, w, x, y, z\} \\
C' &= \{v, w, y\} \\
E' &= \{z\} \\
a = v &\quad b' = y \\
c = w &\quad d' = z \\
\Delta_4 &= \{v, w, x, y, z\} \\
\Delta &= \{v, w, x, y, z\} \\
C' &= \{v, w, x, y\} \\
E' &= \{z\} \\
a = x &\quad b' = x \\
c = y &\quad d' = y \\
\end{align*}
\]
Example of Single Model (homework)

\[ \text{KB} = \{\} \]
\[ \Delta = \{\} \]
\[ I: \]
\[ \Delta = \{a, b, c, d\} \]
\[ C' = \{a, c\} \]
\[ D' = \{b\} \quad E' = \{d\} \]
\[ a' = a \quad b' = b \]
\[ c' = c \quad d' = d \]

\[ \text{KB} = \{a:C, b:D, c:C, d:E\} \]
\[ \Delta = \{\} \]
\[ C' = \{a, c\} \]
\[ D' = \{b\} \quad E' = \{d\} \]
\[ a' = a \quad b' = b \]
\[ c' = c \quad d' = d \]

\[ \text{KB} = \{a:C, b:D, c:C, d:E, b:C\} \]
\[ \Delta = \{\} \]
\[ C' = \{a, c\} \]
\[ D' = \{b\} \quad E' = \{d\} \]
\[ a' = a \quad b' = b \]
\[ c' = c \quad d' = d \]

\[ \text{KB} = \{a:C, b:D, c:C, d:E, b:C, E \sqsubseteq C\} \]
\[ \Delta = \{a, b, c, d\} \]
\[ C' = \{a, b, c\} \]
\[ D' = \{b\} \quad E' = \{d\} \]
\[ a' = a \quad b' = b \]
\[ c' = c \quad d' = d \]

\[ \text{KB} = \{a:C, b:D, c:C, d:E, b:C\} \]
\[ \Delta = \{a, b, c\} \]
\[ C' = \{a, b, c\} \]
\[ D' = \{b\} \quad E' = \{d\} \]
\[ a' = a \quad b' = b \]
\[ c' = c \quad d' = d \]

Short History of Description Logics

**Phase 1:** (early 80's) mostly system development
- Incomplete systems (KL-ONE, Back, Classic, Loom, . . . )
- Based on structural algorithms

**Phase 2:** (mid-80's) first formal investigation
- Development of tableau algorithms and complexity results
- Tableau-based systems for Pspace logics (e.g., Kris, Crack)
- Investigation of optimisation techniques

**Phase 3:** (90's) tableau algorithms and thorough complexity analysis
- Tableau algorithms for very expressive DLs
- Highly optimised tableau systems for ExpTime logics (e.g., FaCT, DLP, Racer)
- Relationship to modal logic and decidable fragments of FOL
**Recent Developments**

**Phase 4: (2010's)**
- **Mainstream applications and tools**
  - **Databases**
    - Consistency of conceptual schemata (EER, UML etc.)
    - Schema integration
    - Query subsumption (w.r.t. a conceptual schema)
  - **Ontologies, e-Science and Semantic Web/Grid**
    - Ontology engineering (schema design, maintenance, integration)
    - Reasoning with ontology-based annotations (data)
- **Mature implementations**
  - Research implementations
    - FaCT, FaCT++, Racer, Pellet, ...
  - Commercial implementations
    - Cerebra system from Network Inference (and now Racer)

---

**Description Logic Reasoning**
Practical Reasons

- Given key role of ontologies in e-Science and Semantic Web, it is essential to provide tools and services to help users:
  - Design and maintain high quality ontologies, e.g.:
    - Meaningful — all named classes can have instances
    - Correct — captured intuitions of domain experts
    - Minimally redundant — no unintended synonyms
    - Richly axiomatised — (sufficiently) detailed descriptions
  - Store (large numbers) of instances of ontology classes, e.g.:
    - Annotations from web pages (or gene product data)
  - Answer queries over ontology classes and instances, e.g.:
    - Find more general/specific classes
    - Retrieve annotations/pages matching a given description
  - Integrate and align multiple ontologies

Why Decidable Reasoning?

- OWL constructors/axioms restricted so reasoning is decidable
- Consistent with Semantic Web's layered architecture
  - XML provides syntax transport layer
  - RDF(S) provides basic relational language and simple ontological primitives
  - OWL provides powerful but still decidable ontology language
  - Further layers (e.g. SWRL) will extend OWL
    - Will almost certainly be undecidable
- Facilitates provision of reasoning services
  - "Practical" algorithms for sound and complete reasoning
  - Several implemented systems
  - Evidence of empirical tractability
Why Sound & Complete Reasoning?

- Important for ontology design
  - Ontologists need to have complete confidence in reasoner
  - Otherwise they will cease to trust results
  - Doubting unexpected results makes reasoner useless
- Important for ontology deployment
  - Many realistic web applications will be agent ↔ agent
  - No human intervention to spot glitches in reasoning
- Incomplete reasoning might be OK in 3-valued system
  - But "don't know" typically treated as "no"

DL Reasoning: Highly Optimised Implementations

- DL reasoning based on tableaux algorithms
- Naive implementation → effective non-termination
- Modern systems include MANY optimisations
- Optimised classification (compute partial ordering)
  - Enhanced traversal (exploits information from previous tests)
  - Use structural information to select classification order
- Optimised subsumption testing (search for models)
  - Normalisation and simplification of concepts
  - Absorption (simplification) of axioms
  - Dependency directed backtracking
  - Caching of satisfiability results and (partial) models
  - Heuristic ordering of propositional and modal expansion
**KB Inférences**

- **Subsomption** :
  - \( C \subseteq T D \) ssi \( \forall I \vdash I C' \subseteq D' \)
  - Structure la connaissance, calcule le graphe terminologique

- **Consistance ou Satisfiable** :
  - concept \( C \) ssi \( \exists I \vdash I C' \neq \emptyset \)
  - ABox ssi \( \exists I \vdash \) Toutes les assertions de ABox
  - KB \( \mathcal{K} = \langle T, A \rangle \) ssi \( \exists I \vdash \) Toutes les assertions de TBox et ABox

- **Aussi Equivalence** \( C \equiv T D \), Instance a : \( c(a) \)

- **Les problèmes d’inférence sont liés** :
  - \( C \subseteq T D \) ssi \( \forall I \not\vdash C' \cap -D' \)
  - \( C \) est consistant ssi \( \forall I \not\vdash C' \subseteq A' \cap -A' \)
  - Inférences standard se réduisent à un test de satisfiabilité

- **Inférences non standard** (étudiées systématiquement depuis \( ~2000 \))
  - Approximations, LCS, MSC, Matching, Unification, Différence, Ré-écriture ...

---

**KB Basic Inference Tasks (2)**

- **Knowledge is correct** (captures intuitions)
  - Does \( C \) subsume \( D \) w.r.t. ontology \( \mathcal{O} \)? (\( C \sqsubseteq D \) in every model \( I \) of \( \mathcal{O} \))

- **Knowledge is minimally redundant** (no unintended synonyms)
  - Is \( C \) equivalent to \( D \) w.r.t. \( \mathcal{O} \)? (\( C = D \) in every model \( I \) of \( \mathcal{O} \))

- **Knowledge is meaningful** (classes can have instances)
  - Is \( C \) is satisfiable w.r.t. \( \mathcal{O} \)? (\( C \neq \emptyset \) in some model \( I \) of \( \mathcal{O} \))

- **Querying knowledge**
  - Is \( x \) an instance of \( C \) w.r.t. \( \mathcal{O} \)? (\( x \in C \) in every model \( I \) of \( \mathcal{O} \))
  - Is \( (x, y) \) an instance of \( R \) w.r.t. \( \mathcal{O} \)? (\( (x, y) \in R \) in every model \( I \) of \( \mathcal{O} \))

- **Above problems can be solved using highly optimised DL reasoners**
DL Reasoning: Basics

- Two types of algorithms are employed to decide inference problems:
  - Structural subsumption algorithms
  - Tableau-based algorithms

Tableau Algorithm (1)

- Tableau Algorithm is the de facto standard reasoning algorithm used in DL
- Basic intuitions
  - Reduces a reasoning problem to concept satisfiability problem
  - Finds an interpretation that satisfies concepts in question.
  - The interpretation is incrementally constructed as a "Tableau"
- Tableaux algorithms are decision procedures for concept satisfiability (\& subsumption \& w.r.t. an ontology)
  - i.e., algorithms return "SAT" iff input concept is satisfiable
Tableau Algorithm (2) (basic case)

- **given**: Wife ⊆ Woman, Woman ⊆ Person
- **question**: if Wife ⊆ Person
- **Reasoning process**
  - Test if there is an individual that is a Woman but not a Person, i.e. test the satisfiability of concept $C_0 = (\text{Wife} \cap \neg \text{Person})$
  - $C_0(x) \rightarrow \text{Wife}(x), (\neg \text{Person})(x)$
  - $\text{Wife}(x) \rightarrow \text{Woman}(x)$
  - $\text{Woman}(x) \rightarrow \text{Person}(x)$
  - **Conflict!**
  - $C_0$ is unsatisfiable, therefore Wife ⊆ Person is true with the given ontology.

---

Tableau Algorithm (3) (General Process)

- **Transform $C$ into negation normal form (NNF)**, i.e. negation occurs only in front of concept names.
  - $\neg \forall R.C \equiv \exists R.\neg C$
  - $\neg \exists R.C \equiv \forall R.\neg C$
  - $\neg \leq n R.C \equiv \geq (n+1)R.C$
  - $\neg \geq (n+1)R.C \equiv \leq n R.C$
  - $\neg \geq 0 R.C \equiv \leq n R.C$

- **Denote** the transformed expression as $C_0$, the algorithm starts with an ABox $A_0 = \{C_0(x_0)\}$, and apply consistency-preserving transformation rules (tableaux expansion) to the ABox as far as possible.
- If one possible ABox is found, $C_0$ is satisfiable.
- If not ABox is found under all search paths, $C_0$ is unsatisfiable.
Tableau Algorithm (4) (Exemple 2)

- Employed for DLs that allow for negation, the subsumption is reduced to deciding satisfiability of concepts: \( C \sqsubseteq D \iff C \sqcap \neg D \) is unsatisfiable.

\[
C_{ex} \sqcap \neg D_{ex} = \exists r. P \sqcap \forall r. Q \sqcap \forall r. Q' \sqcap \neg (\exists r. (P \sqcap Q) \sqcap \forall r. Q')
\]

\[
\equiv \exists r. P \sqcap \forall r. Q \sqcap \forall r. Q' \sqcap (\forall r. (\neg P \sqcup \neg Q) \sqcup \exists r. \neg Q') =: E_{ex}
\]

Current Research (1)

- Extending Description Logics
  - Existing DL systems implement (at most) \( SHIQ \)
  - OWL extends \( SHIQ \) with datatypes and nominals (\( SHOIN(D_n) \))
  - Complex roles, finite domains, concrete domains, fuzzyness, ...
  - Future OWL extensions (e.g., with "rules") (undecidable such as SWRL ?)

- Integrating with other logics/systems
  - E.g., Answer Set Programming

- Alternative reasoning techniques
  - Automata based algorithms
  - Translation into datalog
  - Graph based algorithms (for sub \( ALC \) languages)
Current Research (2)

- Improving Scalability
  - Very large ontologies
  - Very large numbers of individuals
- Other reasoning tasks (non-standard inferences)
  - Matching, LCS, explanation, querying, ...
- Implementation of tools and Infrastructure
  - More expressive languages (such as SHOIN)
  - New algorithmic techniques
  - Tools to support for large scale ontological engineering
    - Editing, visualisation, etc.

Complexité DLs

Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{AC}^C$ without $\exists$</td>
<td>$\mathcal{AC}^C$ (NP)</td>
<td>$\mathcal{AC}^C$ (NP)</td>
<td>$\mathcal{AC}^C$ (NP)</td>
<td>$\mathcal{AC}^C$ (NP)</td>
</tr>
<tr>
<td>$\exists R.C$</td>
<td>$\mathcal{AC}^C$ (NP) without $\exists$ and $\forall R.C$, only $\exists$</td>
<td>$\mathcal{AC}^C$ (NP) without $\exists$ and $\forall R.C$, only $\exists$</td>
<td>$\mathcal{AC}^C$ (NP) without $\exists$ and $\forall R.C$, only $\exists$</td>
<td>$\mathcal{AC}^C$ (NP) without $\exists$ and $\forall R.C$, only $\exists$</td>
</tr>
</tbody>
</table>

Symbols and abbreviations:
- $\mathcal{AC}^C$: Description Logic
- $\exists$: Existential quantification
- $\forall$: Universal quantification
- $R.C$: Role concept
- $\mathcal{AC}^C$: Role concept
- $\mathcal{AC}^C$: Role concept
- $\mathcal{AC}^C$: Role concept
- $\mathcal{AC}^C$: Role concept
- $\mathcal{AC}^C$: Role concept
- $\mathcal{AC}^C$: Role concept

Summary (1)

- **DLs** are family of object oriented KR formalisms related to frames and semantic networks.
  - Characterized by formal semantics and inference services.

- **An Ontology** is an engineering artefact consisting of:
  - A vocabulary of terms.
  - An explicit specification their intended meaning.

- **Ontologies** play a key role in many applications:
  - e-Science, Telecommunications, Banking, Medicine, Databases, Semantic Web, etc.

- **OWL** is a DL based ontology language designed for the Web:
  - Exploits existing standards: XML, RDF(S).
  - Adds KR idioms from object oriented and frame systems.
  - W3C recommendation and already widely adopted in e-Science.
  - DL provides formal foundations and reasoning support.
Summary (2)

- **Reasoning is important because**
  - Understanding is closely related to reasoning
  - Essential for design, maintenance and deployment of ontologies

- **Reasoning support based on DL systems**
  - Sound and complete reasoning
  - Highly optimised implementations

- **Challenges remain**
  - Reasoning with full OWL language
  - (Convincing) demonstration(s) of scalability
  - New reasoning tasks
  - Development of (more) high quality tools and infrastructure

---

Remerciements

- Une pensée toute particulière à tous ceux à qui j’ai emprunté, et ils sont nombreux !

- Et à ceux qui m’ont emprunté ... ☺

Merci !
A Faire pour la semaine prochaine

- Lire les deux premiers chapitres du livre DL Handbook (Baader et al.)*
- Faire les petits exercices de compréhension des Interprétations de KB (cf. slides 26-30-31)
- Revoir les exemples de l’algorithme Tableau de résolution de problème SAT pour le test de subsomption C ⊑ D
- * Je peux les envoyer par courriel à l’un d’entre vous qui se chargera des les distribuer.