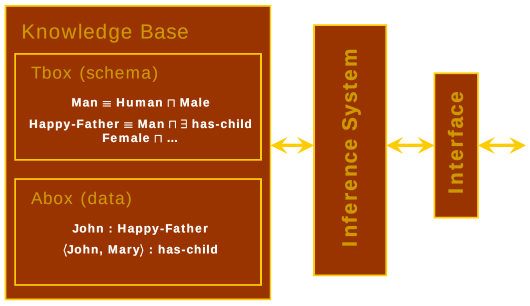
An Introduction to Description Logics

What Are Description Logics?

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

DL Architecture



Short History of Description Logics

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

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

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

Latest Developments

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

Description Logic Family

- **DLs are a family of logic based KR formalisms**
- **Particular languages mainly characterised by:**
  - **Set of constructors for building complex concepts and roles from simpler ones**
  - **Set of axioms for asserting facts about concepts, roles and individuals**

- $\mathcal{ALC}$ **is the smallest DL that is propositionally closed**
  - **Constructors include booleans (and, or, not), and**
  - **Restrictions on role successors**
  - **E.g., concept describing “happy fathers” could be written:**

Man ⊓ ∃hasChild.Female ⊓ ∃hasChild.Male
⊓ ∀hasChild.(Rich ⊔ Happy)

DL Concept and Role Constructors

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

DL Knowledge Base

- **DL Knowledge Base (KB) normally separated into 2 parts:**
  - **TBox is a set of axioms describing structure of domain (i.e., a conceptual schema), e.g.:**
    - **HappyFather ≡ Man ⊓ ∃hasChild.Female ⊓ …**
    - **Elephant ⊏ Animal ⊓ Large ⊓ Grey**
    - **transitive(ancestor)**
  - **ABox is a set of axioms describing a concrete situation (data), e.g.:**
    - **John:HappyFather**
    - **<John,Mary>:hasChild**
- **Separation has no logical significance**
  - **But may be conceptually and implementationally convenient**

OWL as DL: Class Constructors

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾2hasChild	$\exists^{\geqslant n} y.P(x,y)$

- **XMLS datatypes as well as classes in ∀P.C and ∃P.C**
  - **E.g., ∃hasAge.nonNegativeInteger**
- **Arbitrarily complex nesting of constructors**
  - **E.g., Person ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)**

RDFS Syntax

E.g., Person ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor):

```
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:toClass>
        <owl:unionOf rdf:parseType=" collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:hasClass rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:toClass>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

OWL as DL: Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human ⊑ Animal ⊓ Biped
equivalentClass	$C_1 \equiv C_2$	Man ≡ Human ⊓ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male ⊑ ¬Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President Bush} ≡ {G W Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} ⊑ ¬{peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter ⊑ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost ≡ price
inverseOf	$P_1 \equiv P_2^-$	hasChild ≡ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ ⊑ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1P$	⊤ ⊑ ⩽1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1P^-$	⊤ ⊑ ⩽1hasSSN$^-$

- **Axioms (mostly) reducible to inclusion (⊑)**
  - µ **F ≡ G iff both F ⊑ G and G ⊑ F**
- **Obvious FOL equivalences**
  - **E.g., F ≡ G ⇔ ∀{F+{,↔G+{„ F ⊑ G ⇔ ∀{F+{,→G+{,**

XML Schema Datatypes in OWL

- **OWL supports XML Schema primitive datatypes**
  - **E.g., integer, real, string, …**
- **Strict separation between “object” classes and datatypes**
  - **Disjoint interpretation domain Δ_G for datatypes**
    - **For a datavalue g, $g^{\mathcal{I}} \subseteq \Delta_G$**
    - **And $\Delta_G \cap \Delta^{\mathcal{I}} = \emptyset$**
  - **Disjoint “object” and datatype properties**
    - **For a datatype propterty S, $S^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_G$**
    - **For object property V and datatype property S, $V^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$**
- **Equivalent to the “+G$_q$,” in $\mathcal{SHOIN}$+G$_q$,**

Why Separate Classes and Datatypes?

- Philosophical reasons:
  - Datatypes structured by built-in predicates
  - Not appropriate to form new datatypes using ontology language
- Practical reasons:
  - Ontology language remains simple and compact
  - Semantic integrity of ontology language not compromised
  - Implementability not compromised — can use hybrid reasoner
    - Only need sound and complete decision procedure for: $g^{\mathcal{I}}_1 \cap \ldots \cap g^{\mathcal{I}}_q$, where g is a (possibly negated) datatype

OWL DL Semantics

- Mapping OWL to equivalent DL ($\mathcal{SHOIN}$+G$_q$,):
  - Facilitates provision of reasoning services (using DL systems)
  - Provides well defined semantics
- DL semantics defined by interpretations: $\mathcal{I}$ @ +$\Delta^{\mathcal{I}}$/$\cdot^{\mathcal{I}}$,/where
  - $\Delta^{\mathcal{I}}$ is the domain (a non-empty set)
  - $\cdot^{\mathcal{I}}$ is an interpretation function that maps:
    - Concept (class) name $\mathrm{A} \to$ subset $\mathrm{A}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$
    - Role (property) name $\mathrm{R} \to$ binary relation $\mathrm{R}^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$
    - Individual name $\mathrm{i} \to \mathrm{i}^{\mathcal{I}}$ element of $\Delta^{\mathcal{I}}$

DL Semantics

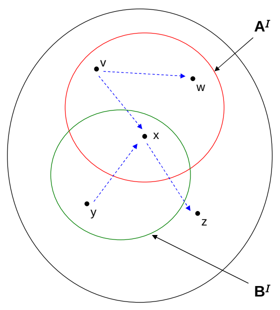
- Interpretation function $\cdot^{\mathcal{I}}$ extends to concept expressions in the obvious way, i.e.:

$$
\begin{aligned}
&(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}\\
&(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}\\
&(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}\\
&\{x\}^{\mathcal{I}} = \{x^{\mathcal{I}}\}\\
&(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}\\
&(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.(x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}\\
&(\leqslant nR)^{\mathcal{I}} = \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \leqslant n\}\\
&(\geqslant nR)^{\mathcal{I}} = \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \geqslant n\}
\end{aligned}
$$

Interpretation Example

Δ = {v, w, x, y, z}
$\mathrm{A}^{\mathcal{I}}$ = {v, w, x}
$\mathrm{B}^{\mathcal{I}}$ = {x, y}
$\mathrm{R}^{\mathcal{I}}$ = {(v, w), (v, x), (y, x), (x, z)}

- ¬ B =
- A ⊓ B =
- ¬ A ⊔ B =
- ∃ R B =
- ∀ R B =
- ∃ R (∃ R A) =
- ∃ R ¬ (A ⊔ B) =
- ⩽ 1 R A =
- ⩾ 1 R A =



DL Knowledge Bases (Ontologies)

- An OWL ontology maps to a DL Knowledge Base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
  - $\mathcal{T}$ (Tbox) is a set of axioms of the form:
    - $\mathrm{C} \sqsubseteq \mathrm{D}$ (concept inclusion)
    - $\mathrm{C} \equiv \mathrm{D}$ (concept equivalence)
    - $\mathrm{R} \sqsubseteq \mathrm{S}$ (role inclusion)
    - $\mathrm{R} \equiv \mathrm{S}$ (role equivalence)
    - $\mathrm{R}^{+} \sqsubseteq \mathrm{R}$ (role transitivity)
  - $\mathcal{A}$ (Abox) is a set of axioms of the form
    - $\mathrm{x} \in \mathrm{D}$ (concept instantiation)
    - $\langle \mathrm{x},\mathrm{y} \rangle \in \mathrm{R}$ (role instantiation)
- Two sorts of Tbox axioms often distinguished
  - "Definitions"
    - $\mathrm{C} \sqsubseteq \mathrm{D}$ or $\mathrm{C} \equiv \mathrm{D}$ where C is a concept name
  - General Concept Inclusion axioms (GCIs)
    - $\mathrm{C} \sqsubseteq \mathrm{D}$ where C in an arbitrary concept

Knowledge Base Semantics

- An interpretation $\mathcal{I}$ satisfies (models) an axiom A ($\mathcal{I} \vDash \mathrm{A}$):
  - $\mathcal{I} \vDash \mathrm{C} \sqsubseteq \mathrm{D}$ iff $\mathrm{C}^{\mathcal{I}} \subseteq \mathrm{D}^{\mathcal{I}}$
  - $\mathcal{I} \vDash \mathrm{C} \equiv \mathrm{D}$ iff $\mathrm{C}^{\mathcal{I}} = \mathrm{D}^{\mathcal{I}}$
  - $\mathcal{I} \vDash \mathrm{R} \sqsubseteq \mathrm{S}$ iff $\mathrm{R}^{\mathcal{I}} \subseteq \mathrm{S}^{\mathcal{I}}$
  - $\mathcal{I} \vDash \mathrm{R} \equiv \mathrm{S}$ iff $\mathrm{R}^{\mathcal{I}} = \mathrm{S}^{\mathcal{I}}$
  - $\mathcal{I} \vDash \mathrm{R}^{+} \sqsubseteq \mathrm{R}$ iff $(\mathrm{R}^{\mathcal{I}})^{+} \subseteq \mathrm{R}^{\mathcal{I}}$
  - $\mathcal{I} \vDash \mathrm{x} \in \mathrm{D}$ iff $\mathrm{x}^{\mathcal{I}} \in \mathrm{D}^{\mathcal{I}}$
  - $\mathcal{I} \vDash \langle \mathrm{x},\mathrm{y} \rangle \in \mathrm{R}$ iff $(\mathrm{x}^{\mathcal{I}},\mathrm{y}^{\mathcal{I}}) \in \mathrm{R}^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a Tbox $\mathcal{T}$ ($\mathcal{I} \vDash \mathcal{T}$) iff $\mathcal{I}$ satisfies every axiom A in $\mathcal{T}$
- $\mathcal{I}$ satisfies an Abox $\mathcal{A}$ ($\mathcal{I} \vDash \mathcal{A}$) iff $\mathcal{I}$ satisfies every axiom A in $\mathcal{A}$
- $\mathcal{I}$ satisfies an KB $\mathcal{K}$ ($\mathcal{I} \vDash \mathcal{K}$) iff $\mathcal{I}$ 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

KB = {}

KB = {a:C, b:D, c:C, d:E}

KB = {a:C, b:D, c:C, d:E, b:C}

KB = {a:C, b:D, c:C, d:E, b:C
D ⊑ C}

KB = {a:C, b:D, c:C, d:E, b:C
D ⊑ C, E ⊑ C}

KB = {a:C, b:D, c:C, d:E, b:C
D ⊑ C, E ⊑ C, d:¬ C}

$\mathcal{I}_1$:
~~Δ = {v, w, x, y, z}~~
~~C^I = {v, w, y}~~
~~D^I = {x, y} E^I = {z}~~
~~a^I = v b^I = x~~
~~c^I = w d^I = y~~

$\mathcal{I}_2$:
~~Δ = {v, w, x, y, z}~~
~~C^I = {v, w, y}~~
~~D^I = {x, y} E^I = {z}~~
~~a^I = v b^I = x~~
~~c^I = w d^I = z~~

$\mathcal{I}_3$:
~~Δ = {v, w, x, y, z}~~
~~C^I = {v, w, y}~~
~~D^I = {x, y} E^I = {z}~~
~~a^I = v b^I = y~~
~~c^I = w d^I = z~~

$\mathcal{I}_4$:
~~Δ = {v, w, x, y, z}~~
~~C^I = {v, w, x, y}~~
~~D^I = {x, y} E^I = {z}~~
~~a^I = v b^I = x~~
~~c^I = y d^I = y~~

Example of Single Model

KB = {}

$\mathcal{I}$:
Δ = {}

KB = {a:C, b:D, c:C, d:E}

KB = {a:C, b:D, c:C, d:E, b:C}

$\mathcal{I}$:
Δ = {a, b, c, d}
C^I = {a, c}
D^I = {b} E^I = {d}
a^I = a b^I = b
c^I = c d^I = d

KB = {a:C, b:D, c:C, d:E, b:C
E ⊑ C}

$\mathcal{I}$:
Δ = {a, b, c, d}
C^I = {a, b, c}
D^I = {b} E^I = {d}
a^I = a b^I = b
c^I = c d^I = d

$\mathcal{I}$:
Δ = {a, b, c, d}
C^I = {a, b, c, d}
D^I = {b} E^I = {d}
a^I = a b^I = b
c^I = c d^I = d

Inference Tasks

- Knowledge is correct (captures intuitions)
  - C subsumes D w.r.t. $\mathcal{K}$ iff for *every* model $\mathcal{I}$ of $\mathcal{K}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- Knowledge is minimally redundant (no unintended synonyms)
  - C is equivallent to D w.r.t. $\mathcal{K}$ iff for *every* model $\mathcal{I}$ of $\mathcal{K}$, $C^{\mathcal{I}} = D^{\mathcal{I}}$
- Knowledge is meaningful (classes can have instances)
  - C is satisfiable w.r.t. $\mathcal{K}$ iff there exists *some* model $\mathcal{I}$ of $\mathcal{K}$ s.t. $C^{\mathcal{I}} \neq \emptyset$

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

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

Single Model -v- Multiple Model

Multiple models:

- Expressively powerful
  - Boolean connectives, including ¬ and ⊔
- Can capture incomplete information
  - E.g., using ⊔ and ∃
- Monotonic
  - Adding information preserves truth
- Reasoning (e.g., querying) is hard/slow
- Queries may give counter-intuitive results in some cases

Single model:

- Expressively weaker (in most respects)
  - No negation or disjunction
- Can't capture incomplete information
- Nonmonotonic
  - Adding information does not preserve truth
- Reasoning (e.g., querying) is easy/fast
- Queries may give counter-intuitive results in some cases