Description logics

Description Logics

- A family of KR formalisms, based on FOPL
  decidable, supported by automatic reasoning systems
- Used for modelling of application domains
- Classification of concepts and individuals
  concepts (unary predicates), subconcept (subsumption), roles (binary predicates), individuals (constants), constructors for building concepts, equality …

[Baader et al. 2002]

Applications

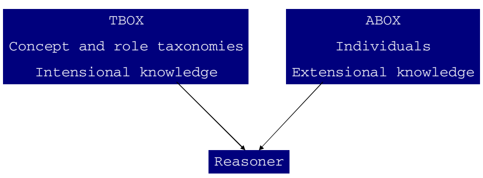
- software management
- configuration management
- natural language processing
- clinical information systems
- information retrieval
- …

- Ontologies and the Web

Outline

- DL languages
  - syntax and semantics
- DL reasoning services
  - algorithms, complexity
- DL systems
- DLs for the web

Tbox and Abox



Syntax - $\mathcal{AL}$

R atomic role, A atomic concept

$C,D \rightarrow A$ | (atomic concept)

T | (universal concept, top)

$\bot$ | (bottom concept)

$\neg A$ | (atomic negation)

$C \cap D$ | (conjunction)

$\forall R.C$ | (value restriction)

$\exists R.T$ (limited existential quantification)

$\mathcal{AL}[\mathcal{X}]$

$\mathcal{C}$ ¬C (concept negation)

$\mathcal{U}$ C U D (disjunction)

$\mathcal{E}$ ∃R.C (existential quantification)

$\mathcal{N}$ ≥ n R, ≤ n R (number restriction)

$\mathcal{Q}$ ≥ n R.C, ≤ n R.C (qualified number restriction)

Example

Team

Team ∩ ≥ 10 hasMember

Team ∩ ≥ 11 hasMember
∩ ∀ hasMember.Soccer-player

$\mathcal{AL}[\mathcal{X}]$

$\mathcal{R}$ R ∩ S (role conjunction)

$\mathcal{I}$ R- (inverse roles)

$\mathcal{H}$ (role hierarchies)

$\mathcal{F}$ $u_1 = u_2$, $u_1 \neq u_2$ (feature (dis)agreements)

$\mathcal{S}[\mathcal{X}]$

$\mathcal{S}$ $\mathcal{ALC}$ + transitive roles

$\mathcal{SHIQ}$ $\mathcal{ALC}$ + transitive roles
+ role hierarchies
+ inverse roles
+ number restrictions

Tbox

- Terminological axioms:
  - C = D (R = S)
  - C ⊆ D (R ⊆ S)
  - (disjoint C D)
- An equality whose left-hand side is an atomic concept is a definition.
- A finite set of definitions T is a Tbox (or terminology) if no symbolic name is defined more than once.

Example Tbox

Soccer-player ⊆ T

Team ⊆ ≥ 2 hasMember

Large-Team = Team ∩ ≥ 10 hasMember

S-Team = Team ∩ ≥ 11 hasMember
∩ ∀ hasMember.Soccer-player

DL as sublanguage of FOPL

Team(this)
∧
(∃ $x_1,\ldots,x_{11}$:
hasMember(this,x1) ∧ … ∧ hasMember(this,x11)
∧ $x_1 \neq x_2$ ∧ … ∧ $x_{10} \neq x_{11}$)

∧
(∀ x: hasMember(this,x)   Soccer-player(x))

Abox

- Assertions about individuals:
  - C(a)
  - R(a,b)

Example

Ida-member(Sture)

Individuals in the description language

- O   {i1, …, ik}   (one-of)
- R:a (fills)

Example

(S-Team ∩ hasMember:Sture)(IDA-FF)

Knowledge base

A knowledge base is a tuple < *T*, *A* > where *T* is a Tbox and *A* is an Abox.

Example KB

Soccer-player ⊆ T
Team ⊆ ≥ 2 hasMember
Large-Team = Team ∩ ≥ 10 hasMember
S-Team = Team ∩ ≥ 11 hasMember
∩ ∀ hasMember.Soccer-player

Ida-member(Sture)

(S-Team ∩ hasMember:Sture)(IDA-FF)

$\mathcal{AL}$ (Semantics)

An interpretation $\mathcal{I}$ consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function $.^{\mathcal{I}}$ which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

The interpretation function is extended to concept definitions using inductive definitions.

$\mathcal{AL}$ (Semantics)

C,D → A |	(atomic concept)	
T |	(universal concept)	$T^{\mathcal{I}} = \Delta^{\mathcal{I}}$
⊥ |	(bottom concept)	$\bot^{\mathcal{I}} = \emptyset$
¬A |	(atomic negation)	$(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
C ∩ D |	(conjunction)	$(C \cap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
∀R.C |	(value restriction)	$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
∃R.T |	(limited existential quantification)	$(\exists R.T)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\}$

$\mathcal{ALC}$ (Semantics)

$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$

$(C \cup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$

$(\geq n\ R)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \# \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\} \geq n\}$

$(\leq n\ R)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \# \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\} \leq n\}$

$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}} : (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$

Semantics

Individual i

$i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

Unique Name Assumption:
if $i_1 \neq i_2$ then $i_1^{\mathcal{I}} \neq i_2^{\mathcal{I}}$

Semantics

An interpretation $.^{\mathcal{I}}$ is a model for a terminology *T* iff

$C^{\mathcal{I}} = D^{\mathcal{I}}$ for all C = D in *T*

$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all a C ⊆ D in *T*

$C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for all (disjoint C D) in *T*

Semantics

An interpretation $.^{\mathcal{I}}$ is a model for a knowledge base $<T, A>$ iff

$.^{\mathcal{I}}$ is a model for T

$a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all C(a) in A

$<a^{\mathcal{I}},b^{\mathcal{I}}> \in R^{\mathcal{I}}$ for all R(a,b) in A

Semantics - acyclic Tbox

Bird = Animal $\cap$ $\forall$ Skin.Feather

$\Delta^{\mathcal{I}}$ = {tweety, goofy, fea1, fur1}

Animal$^{\mathcal{I}}$ = {tweety, goofy}

Feather$^{\mathcal{I}}$ = {fea1}

Skin$^{\mathcal{I}}$ = {<tweety,fea1>, <goofy,fur1>}

Bird$^{\mathcal{I}}$ = {tweety}

Semantics - cyclic Tbox

QuietPerson = Person $\cap$ $\forall$ Friend.QuietPerson

(A = F(A))

$\Delta^{\mathcal{I}}$ = {john, sue, andrea, bill}

Person$^{\mathcal{I}}$ = {john, sue, andrea, bill}

Friend$^{\mathcal{I}}$ = {<john,sue>, <andrea,bill>, <bill,bill>}

QuietPerson$^{\mathcal{I}}$ ={john, sue}

QuietPerson$^{\mathcal{I}}$ ={john, sue, andrea, bill}

Semantics - cyclic Tbox

Descriptive semantics: A = F(A) is a constraint stating that A has to be some solution for the equation.

- Not appropriate for defining concepts
- Necessary and sufficient conditions for concepts

Human = Mammal $\cap$ $\exists$ Parent $\cap$ $\forall$ Parent.Human

Semantics - cyclic Tbox

Least fixpoint semantics: A = F(A) specifies that A is to be interpreted as the smallest solution (if it exists) for the equation.

- Appropriate for inductively defining concepts

DAG = EmptyDAG U (Node $\cap$ $\forall$ Arc.DAG)

Human = Mammal $\cap$ $\exists$ Parent $\cap$ $\forall$ Parent.Human

Human = $\perp$

Semantics - cyclic Tbox

Greatest fixpoint semantics: A = F(A) specifies that A is to be interpreted as the greatest solution (if it exists) for the equation.

- Appropriate for defining concepts whose individuals have circularly repeating structure

FoB = Blond $\cap$ $\exists$ Child.FoB

Human = Mammal $\cap$ $\exists$ Parent $\cap$ $\forall$ Parent.Human

Horse = Mammal $\cap$ $\exists$ Parent $\cap$ $\forall$ Parent.Horse

Human = Horse

Open world vs closed world semantics

Databases: closed world reasoning
- **database instance represents one interpretation**
  - **absence of information interpreted as negative information**
  - **"complete information"**
- **query evaluation is finite model checking**

DL: open world reasoning
- **Abox represents many interpretations (its models)**
  - **absence of information is lack of information**
  - **"incomplete information"**
- **query evaluation is logical reasoning**

Open world vs closed world semantics

hasChild(Jocasta, Oedipus)
hasChild(Jocasta, Polyneikes)
hasChild(Oedipus, Polyneikes)
hasChild(Polyneikes, Thersandros)
patricide(Oedipus)
¬ patricide(Thersandros)

Does it follow from the Abox that
$\exists$hasChild.(patricide $\sqcap$ $\exists$hasChild. ¬ patricide)(Jocasta) ?

Reasoning services

- Satisfiability of concept
- Subsumption between concepts
- Equivalence between concepts
- Disjointness of concepts

- Classification

- Instance checking
- Realization
- Retrieval
- Knowledge base consistency

Reasoning services

- Satisfiability of concept
  - C is satisfiable w.r.t. $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is not empty.
- Subsumption between concepts
  - C is subsumed by D w.r.t. $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$.
- Equivalence between concepts
  - C is equivalent to D w.r.t. $\mathcal{T}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$.
- Disjointness of concepts
  - C and D are disjoint w.r.t. $\mathcal{T}$ if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{T}$.

Reasoning services

- Reduction to subsumption
  - C is unsatisfiable iff C is subsumed by $\bot$
  - C and D are equivalent iff C is subsumed by D and D is subsumed by C
  - C and D are disjoint iff C $\sqcap$ D is subsumed by $\bot$
- The statements also hold w.r.t. a Tbox.

Reasoning services

- Reduction to unsatisfiability
  - C is subsumed by D iff C $\sqcap$ ¬D is unsatisfiable
  - C and D are equivalent iff both (C $\sqcap$ ¬D) and (D $\sqcap$ ¬C) are unsatisfiable
  - C and D are disjoint iff C $\sqcap$ D is unsatisfiable
- The statements also hold w.r.t. a Tbox.

Tableau algorithms

- To prove that C subsumes D:
  - If C subsumes D, then it is impossible for an individual to belong to D but not to C.
  - Idea: Create an individual that belongs to D and not to C and see if it causes a contradiction.
  - If **always** a contradiction (clash) then subsumption is proven. Otherwise, we have found a model that contradicts the subsumption.

Tableau algorithms

- Based on constraint systems.
  - S = { x: ¬C ∩ D }
  - Add constraints according to a set of propagation rules
  - Until clash or no constraint is applicable

Tableau algorithms – de Morgan rules

¬ ¬ C   C

¬ (A ∩ B)   ¬ A U ¬ B

¬ (A U B)   ¬ A ∩ ¬ B

¬ (∀ R.C)   ∃ R.(¬ C)

¬ (∃ R.C)   ∀ R.(¬ C)

Tableau algorithms – constraint propagation rules

- S $_{\cap}$ {x:C_1, x:C_2} U S

  if x: C_1 ∩ C_2 in S
  and either x:C_1 or x:C_2 is not in S

- S $_{U}$ {x:D} U S

  if x: C_1 U C_2 in S and neither x:C_1 or x:C_2 is in S, and D = C_1 or D = C_2

Tableau algorithms – constraint propagation rules

- S $_{\forall}$ {y:C} U S

  if x: ∀ R.C in S and xRy in S and y:C is not in S

- S $_{\exists}$ {xRy, y:C} U S

  if x: ∃ R.C in S and y is a new variable and there is no z such that both xRz and z:C are in S

Example

- ST: Tournament
  ∩ ∃ hasParticipant.Swedish
- SBT: Tournament
  ∩ ∃ hasParticipant.(Swedish ∩ Belgian)

Example 1

- SBT => ST?
- S = { x:
  ¬(Tournament ∩ ∃ hasParticipant.Swedish)
  ∩ (Tournament
  ∩ ∃ hasParticipant.(Swedish ∩ Belgian))
  }

Example 1

- S = { x:
  (¬Tournament
  U ∀ hasParticipant.¬ Swedish)
  ∩ (Tournament
  ∩ ∃ hasParticipant.(Swedish ∩ Belgian))
  }

Example 1

∩-rule:

- S = {
  x: (¬Tournament
  U ∀ hasParticipant.¬ Swedish)
  ∩ (Tournament
  ∩ ∃ hasParticipant.(Swedish ∩ Belgian)),
  x: ¬Tournament
  U ∀ hasParticipant.¬ Swedish,
  x: Tournament,
  x: ∃ hasParticipant.(Swedish ∩ Belgian)
  }

Example 1

∃ -rule:

- S = {
  x: (¬Tournament U ∀ hasParticipant.¬ Swedish)
  ∩ (Tournament
  ∩ ∃ hasParticipant.(Swedish ∩ Belgian)),
  x: ¬Tournament
  U ∀ hasParticipant.¬ Swedish,
  x: Tournament,
  x: ∃ hasParticipant.(Swedish ∩ Belgian),
  x hasParticipant y, y: (Swedish ∩ Belgian)
  }

Example 1

∩-rule:

- S= {x: (¬Tournament U ∀ hasParticipant.¬ Swedish)
  ∩ (Tournament
  ∩ ∃ hasParticipant.(Swedish ∩ Belgian)),
  x: ¬Tournament U ∀ hasParticipant.¬ Swedish,
  x: Tournament,
  x: ∃ hasParticipant.(Swedish ∩ Belgian),
  x hasParticipant y, y: (Swedish ∩ Belgian),
  y: Swedish, y: Belgian }

Example 1

U-rule, choice 1

- S = { x: (¬Tournament U ∀ hasParticipant.¬ Swedish)
  ∩ (Tournament
  ∩ ∃ hasParticipant.(Swedish ∩ Belgian)),
  x: ¬Tournament U ∀ hasParticipant.¬ Swedish,
  x: Tournament,
  x: ∃ hasParticipant.(Swedish ∩ Belgian),
  x hasParticipant y, y: (Swedish ∩ Belgian),
  y: Swedish, y: Belgian,
  x: ¬Tournament
  }

clash

Example 1

U-rule, choice 2
- S = {x: (¬Tournament U ∀ hasParticipant.¬ Swedish)
  ∩ (Tournament
  ∩ ∃ hasParticipant.(Swedish ∩ Belgian)),
  x: ¬Tournament U ∀ hasParticipant.¬ Swedish,
  x: Tournament,
  x: ∃ hasParticipant.(Swedish ∩ Belgian),
  x hasParticipant y, y: (Swedish ∩ Belgian),
  y: Swedish, y: Belgian,
  x: ∀ hasParticipant.¬ Swedish

}

Example 1

choice 2 – continued
∀-rule
- S = {
  x: (¬Tournament U ∀ hasParticipant.¬ Swedish)
  ∩ (Tournament ∩ ∃ hasParticipant.(Swedish ∩ Belgian)),
  x: ¬Tournament U ∀ hasParticipant.¬ Swedish,
  x: Tournament,
  x: ∃ hasParticipant.(Swedish ∩ Belgian),
  x hasParticipant y, y: (Swedish ∩ Belgian),
  y: Swedish, y: Belgian,
  x: ∀ hasParticipant.¬ Swedish,
  y: ¬ Swedish
  }

clash

Example 2

- ST => SBT?
- S = { x:
  ¬ (Tournament
  ∩ ∃ hasParticipant.(Swedish ∩ Belgian))
  ∩ (Tournament ∩ ∃ hasParticipant.Swedish)
  }

Example 2

- S = { x:
  (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian))
  ∩ (Tournament ∩ ∃ hasParticipant.Swedish)
  }

Example 2

∩-rule
- S = {
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian))
  ∩ (Tournament ∩ ∃ hasParticipant.Swedish),
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian)),
  x: Tournament,
  x: ∃ hasParticipant.Swedish
  }

Example 2

∃ -rule
- S = {
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian))
  ∩ (Tournament ∩ ∃ hasParticipant.Swedish),
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian)),
  x: Tournament,
  x: ∃ hasParticipant.Swedish,
  x hasParticipant y, y: Swedish
  }

Example 2

U –rule, choice 1

- S = {
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian))
  ∩ (Tournament ∩ ∃ hasParticipant.Swedish),
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian)),
  x: Tournament,
  x: ∃ hasParticipant.Swedish,
  x hasParticipant y, y: Swedish,
  x: ¬Tournament
  }
  clash

Example 2

U –rule, choice 2

- S = {
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian))
  ∩ (Tournament ∩ ∃ hasParticipant.Swedish),
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian)),
  x: Tournament,
  x: ∃ hasParticipant.Swedish,
  x hasParticipant y, y: Swedish,
  x: ∀ hasParticipant.(¬ Swedish U ¬ Belgian)
  }

Example 2

choice 2 continued
∀–rule

- S = {
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian))
  ∩ (Tournament ∩ ∃ hasParticipant.Swedish),
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian)),
  x: Tournament,
  x: ∃ hasParticipant.Swedish,
  x hasParticipant y, y: Swedish,
  x: ∀ hasParticipant.(¬ Swedish U ¬ Belgian),
  y: (¬ Swedish U ¬ Belgian)
  }

Example 2

choice 2 continued
U–rule, choice 2.1

- S = {
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian))
  ∩ (Tournament ∩ ∃ hasParticipant.Swedish),
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian)),
  x: Tournament,
  x: ∃ hasParticipant.Swedish,
  x hasParticipant y, y: Swedish,
  x: ∀ hasParticipant.(¬ Swedish U ¬ Belgian),
  y: (¬ Swedish U ¬ Belgian),
  y: ¬ Swedish
  } clash

Example 2

choice 2 continued
U–rule, choice 2.2

- S = {
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian))
  ∩ (Tournament ∩ ∃ hasParticipant.Swedish),
  x: (¬Tournament
  U ∀ hasParticipant.(¬ Swedish U ¬ Belgian)),
  x: Tournament,
  x: ∃ hasParticipant.Swedish,
  x hasParticipant y, y: Swedish,
  x: ∀ hasParticipant.(¬ Swedish U ¬ Belgian),
  y: (¬ Swedish U ¬ Belgian),
  y: ¬ Belgian
  } ok, model

Complexity - languages

- Overview available via the DL home page at http://dl.kr.org

Example tractable language:

A, T,⊥ , ¬A, C ∩ D, ∀R.C, ≥ n R, ≤ n R

Reasons for intractability:

choices, e.g. C U D

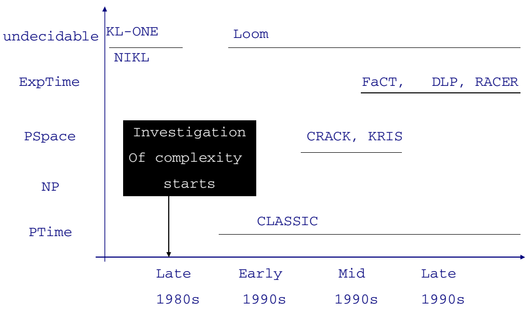
exponential size models,

e.g interplay universal and existential quantification

Reasons for undecidability:

e.g. role-value maps R=S

Systems



Systems

- Overview available via the DL home page at http://dl.kr.org
- Current systems include: CEL, Cerebra Enginer, FaCT++, fuzzyDL, HermiT, KAON2, MSPASS, Pellet, QuOnto, RacerPro, SHER

Extensions

- Time
- Defaults
- Part-of
- Knowledge and belief
- Uncertainty (fuzzy, probabilistic)

DAML+OIL Class Constructors

Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer
complementOf	$\neg C$	¬Male
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}
toClass	$\forall P.C$	∀hasChild.Doctor
hasClass	$\exists P.C$	∃hasChild.Lawyer
hasValue	$\exists P.\{x\}$	∃citizenOf.{USA}
minCardinalityQ	$\geqslant n P.C$	⩾2hasChild.Lawyer
maxCardinalityQ	$\leqslant n P.C$	⩽1hasChild.Male
cardinalityQ	$= n\ P.C$	=1 hasParent.Female

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

DAML+OIL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human ⊑ Animal ⊓ Biped
sameClassAs	$C_1 \equiv C_2$	Man ≡ Human ⊓ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter ⊑ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost ≡ price
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} ≡ {G_W_Bush}
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male ⊑ ¬Female
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} ⊑ ¬{peter}
inverseOf	$P_1 \equiv P_2^-$	hasChild ≡ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ ⊑ ancestor
uniqueProperty	$\top \sqsubseteq \leqslant 1 P$	⊤ ⊑ ⩽1hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1 P^-$	⊤ ⊑ ⩽1isMotherOf$^-$

- Axioms (mostly) **reducible to subClass/PropertyOf**

OWL

- OWL-Lite, OWL-DL, OWL-Full: increasing expressivity
- A legal OWL-Lite ontology is a legal OWL-DL ontology is a legal OWL-Full ontology
- OWL-DL: expressive description logic, decidable
- XML-based
- RDF-based (OWL-Full is extension of RDF, OWL-Lite and OWL-DL are extensions of a restriction of RDF)

OWL-Lite

- **Class**, subClassOf, equivalentClass
- intersectionOf (only named classes and restrictions)
- **Property**, subPropertyOf, equivalentProperty
- domain, range (global restrictions)
- inverseOf, TransitiveProperty (*), SymmetricProperty, FunctionalProperty, InverseFunctionalProperty
- allValuesFrom, someValuesFrom (local restrictions)
- minCardinality, maxCardinality (only 0/1)
- **Individual**, sameAs, differentFrom, AllDifferent

(*) restricted

OWL-DL

- **Type separation** (class cannot also be individual or property, property cannot be also class or individual), Separation between DatatypeProperties and ObjectProperties
- **Class –complex classes**, subClassOf, equivalentClass, *disjointWith*
- intersectionOf, *unionOf, complementOf*
- **Property**, subPropertyOf, equivalentProperty
- domain, range (global restrictions)
- inverseOf, TransitiveProperty (*), SymmetricProperty, FunctionalProperty, InverseFunctionalProperty
- allValuesFrom, someValuesFrom (local restrictions), *oneOf, hasValue*
- minCardinality, maxCardinality
- **Individual**, sameAs, differentFrom, AllDifferent

(*) restricted

References

- Baader, Calvanese, McGuinness, Nardi, Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2003.
- Donini, Lenzerini, Nardi, Schaerf, Reasoning in description logics. *Principles of knowledge representation*. CSLI publications. pp 191-236. 1996.
- dl.kr.org
- www.daml.org
- www.w3.org (owl)