Logics for the Semantic Web

Description logics and the Web

RP - 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]

From introduction lecture

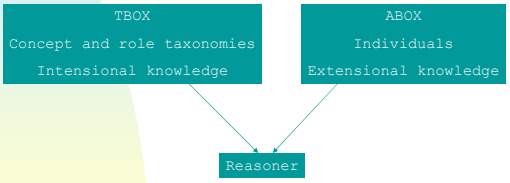
Applications

- 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 versus other formalisms
- DL systems
- DL extensions
- DLs for the web

Tbox and Abox



Syntax - $\mathcal{AL}$

R atomic role, A atomic concept

C,D → A | (atomic concept)
T | (universal concept, top)
⊥ | (bottom concept)
¬A | (atomic negation)
C ∩ D | (conjunction)
∀R.C | (value restriction)
∃R.T (limited existential quantification)

$\mathcal{AL}[X]$

$\mathcal{C}$ $\neg$C (concept negation)

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

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

$\mathcal{N}$ $\geq$ n R, $\leq$ n R (number restriction)

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

Example

Team

Team $\cap \geq$ 10 hasMember

Team $\cap \geq$ 11 hasMember
$\cap \forall$ hasMember.Soccer-player

$\mathcal{AL}[X]$

$\mathcal{R}$ R $\cap$ 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)

$\mathcal{S}[X]$

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

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

$\mathcal{DLR}[X]$

$\mathcal{DLR}$ P atomic relation, A atomic concept

C,D $\rightarrow$ A | (atomic concept)
T_1 | (top)
$\neg$C | (concept negation)
C $\cap$ D | (concept conjunction)
$\leq$ k R [$i] | (component number restriction)
$\exists$[$i]R (existential component quantification)

R,S $\rightarrow$ P | (atomic relation)
T_n |
$\neg$R | (relation "negation")
R $\cap$ S | (relation conjunction)
($i/n:C) (selection)

Tbox

- Terminological axioms:
  - C = D (R = S)
  - C $\sqsubseteq$ D (R $\sqsubseteq$ 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 $\subseteq$ T

Team $\subseteq \geq 2$ hasMember

Large-Team = Team $\cap \geq 10$ hasMember

S-Team = Team $\cap \geq 11$ hasMember
$\cap \forall$ hasMember.Soccer-player

DL as sublanguage of FOPL

Team(this)
^
($\exists$ $x_1,...,x_{11}$:
hasMember(this,x1) ^ … ^ hasMember(this,x11)
^ $x_1 \neq x_2$ ^ … ^ $x_{10} \neq x_{11}$)
^
($\forall$ x: hasMember(this,x) → Soccer-player(x))

Abox

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

Individuals in the description language

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

Knowledge base

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

Example KB

Soccer-player $\subseteq$ T
Team $\subseteq \geq 2$ hasMember
Large-Team = Team $\cap \geq 10$ hasMember
S-Team = Team $\cap \geq 11$ hasMember
$\cap \forall$ hasMember.Soccer-player

Ida-member(Anders)

(S-Team $\cap$ hasMember:Anders)(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 ex. 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 $C \subseteq D$ 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 - cyclic Tbox

Bird = Animal ∩ ∀ 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 ∩ ∀ 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 ∩ ∃ Parent ∩ ∀ 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 ∩ ∃ Arc ∩ ∀ Arc.DAG)

Human = Mammal ∩ ∃ Parent ∩ ∀ Parent.Human
→ Human = ⊥

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 ∩ ∃ Child.FoB

Human = Mammal ∩ ∃ Parent ∩ ∀ Parent.Human
Horse = Mammal ∩ ∃ Parent ∩ ∀ Parent.Horse
→ Human = Horse

Rules vs axioms

C => D
"if an individual is proved to be an instance of C then derive that is also an instance of D"

Operational semantics: series of knowledge bases based on application of rules

Declarative semantics:

Is C => D equivalent to C ⊆ D?

Rules vs axioms

C,D → **K**C (epistemic concept)

KC denotes the objects for which the knowledge base knows that they are instances of C.

C=>D replaced by **K**C ⊆ D

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 ⊥
  - C and D are equivalent iff C is subsumed by D and D is subsumed by C
  - C and D are disjoint iff C ∩ D is subsumed by ⊥
- The statements also hold w.r.t. a Tbox.

Reasoning services

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

Reasoning services

- reasoning with Tbox

For acyclic Tboxes reasoning with Tboxes can be reduced to reasoning without Tbox by expanding concepts w.r.t. the TBox.

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

Q: S-Team ∩ ∀ hasMember.IDA-employee

Expand → Team ∩ ≥ 11 hasMember
∩ ∀ hasMember.Soccer-player
∩ ∀ hasMember.IDA-employee

Reasoning services

- reasoning with Tbox

The expansion may be computationally costly.

$C_1 = \forall R_1.C_0 \cap \forall R_2.C_0 \cap \ldots \cap \forall R_m.C_0$
$C_2 = \forall R_1.C_1 \cap \forall R_2.C_1 \cap \ldots \cap \forall R_m.C_1$
…
$C_n = \forall R_1.C_{n-1} \cap \forall R_2.C_{n-1} \cap \ldots \cap \forall R_m.C_{n-1}$

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
∃hasChild.(patricide ∩ ∃hasChild. ¬ patricide)(Jocasta) ?

Algorithms

- Structural subsumption algorithms
- Tableau algorithms

Structural subsumption

- Cannot treat disjunction, full negation, full existential restrictions
- Two steps
  - normalize
  - compare

Structural subsumption

- Normalize
  - Convert primitives
  - Expand terms
  - Use normalization rules

Structural subsumption – normalization rules

- $\forall$ r.T = T
- $\forall$ r.$C_1 \cap \forall$ r.$C_2 = \forall$ r.($C_1 \cap C_2$)
- m < n → $\geq$ m R $\cap \geq$ n R = $\geq$ n R
- m < n → $\leq$ m R $\cap \leq$ n R = $\leq$ m R
- m < n → $\leq$ m R $\cap \geq$ n R = $\perp$

Structural subsumption

- Compare

C subsumes D if

"requirements for D are stronger than requirements for C"

subsumption rules

Structural subsumption – subsumption rules

- $C_1 \sqsubseteq C_2 \rightarrow \forall$ r.$C_1 \sqsubseteq \forall$ r.C_2
- m < n → $\geq$ n R $\sqsubseteq \geq$ m R
- m < n → $\leq$ m R $\sqsubseteq \leq$ n R

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 contradiction (clash) then subsumption is proven. Otherwise, we have found a model that contradicts the subsumption.

Tableau algorithms

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

Tableau algorithms – de Morgan rules

$\neg \neg$ C → C

$\neg$ (A $\cap$ B) → $\neg$ A U $\neg$ B

$\neg$ (A U B) → $\neg$ A $\cap \neg$ B

$\neg$ ($\forall$ R.C) → $\exists$ R.($\neg$ C)

$\neg$ ($\exists$ R.C) → $\forall$ R.($\neg$ C)

Tableau algorithms – constraint propagation rules

- S → $_{\cap}$ {x:C_1, x:C_2} U S

  if x: $C_1 \cap C_2$ 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

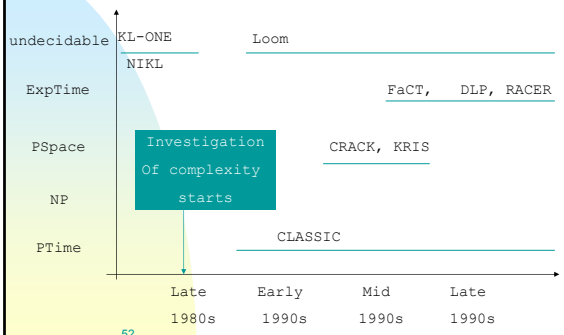
- S → $_{\exists}$ {xRy, y:C} U S

  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

Complexity

- Tractability
- Exponential
- Undecidable

Complexity - systems



Complexity

- tractable language 1

Primitive concepts
Negation of primitive concepts
Top
Bottom
Concept conjunction
Value restriction
Number restriction

Complexity

- tractable language 2

Primitive concepts
Concept conjunction
Value restriction
Limited existential quantification
Primitive roles
Role negation
Role conjunction
Role composition

Complexity

- Tractability preserved adding

Functional roles
Role-value maps for functional roles
Disjunction for primitive concepts

Complexity

- Reasons for intractability
  - OR-branching: exponential number of candidate models, alternatives originating from alternative choices
  - example: disjunctions
    C U D

Complexity

- Reasons for intractability
  - AND-branching: exponential size of a candidate model,
  - Example: interplay qualified existential and universal quantifiers

$$\exists P_1.C_{11} \cap \exists P_1.C_{12} \cap \forall P_1.(\exists P_2.C_{21} \cap \exists P_2.C_{22} \cap \forall P_2.(\exists P_3.C_{31} \cap \exists P_3.C_{32} \cap \forall P_3.(\ldots (\exists P_n.C_{n1} \cap \exists P_n.C_{n2} \cap \forall P_n.D))))$$

Complexity

- Reasons for undecidability
  - role-value maps
    R = S
    $R \subseteq S$

DL vs other formalisms

- Variable fragments
- Graded fragments
- Modal logics
- ER, OO
- XML

DL vs variable fragments

- $\mathcal{L}^k$: FOPL over unary and binary predicates with at most k variables
  $\mathcal{C}^k$: FOPL over unary and binary predicates with at most k variables with counting quantifiers
  - $\mathcal{ALC}$ can be translated into $\mathcal{L}^2$
    also: inverse roles, role conjunction, role disjunction, role negation, one-of
  - number restrictions → $\mathcal{C}^2$
  - transitive closure → second order logic

DL vs variable fragments

- $\mathcal{ALCR}$ is less expressive than $\mathcal{L}^2$
- $\mathcal{ALCNR}$ is less expressive than $\mathcal{C}^2$

- $\mathcal{ALCO}$ + role negation, role conjunction, inverse roles, cross-product of concepts, identity role
  is as expressive as $\mathcal{L}^2$
- $\mathcal{ALCO}$ + role negation, role conjunction, inverse roles, cross-product of concepts, identity role + all kinds of role-value maps
  is as expressive as $\mathcal{L}^3$

DL vs guarded fragments

- FOPL where quantification is restricted to variables that are 'guarded' by appropriate atoms

  First guarded fragment
  $\exists$ y (P(x,y) ^ Φ(y))
  $\forall$ y (P(x,y) $\rightarrow$ Φ(y))

  Guarded fragment
  $\exists$ y (P(x,y) ^ Φ(x,y))
  $\forall$ y (P(x,y) $\rightarrow$ Φ(x,y))

- Guarded fragments are decidable.

DL vs modal logics

- $\mathcal{ALC}$ - $\mathbf{K}_m$

$f(A) = A$

$f(\neg C) = \neg f(C)$

$f(C \cap D) = f(C) \wedge f(D)$

$f(C \cup D) = f(C) \vee f(D)$

$f(\forall R_i.C) = \Box_i f(C)$

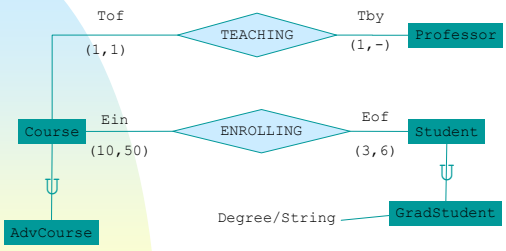
$f(\exists R_i.C) = \Diamond_i f(C)$

DL vs modal logics

- $\mathcal{ALC}_{reg}$ – PDL

Propositional Dynamic Logic
- for dynamic behavior of programs
- Build complex programs from atomic programs using operators (choice, composition, iteration)

DL vs EER



DL vs EER

- Abbreviation:

  = n R for ($\geq$ n R) $\cap$ ($\leq$ n R)

  $\exists$ R for $\exists$ R.T

DL vs EER

- TEACHING ⊆ ∀ Tof.Course ∩ = 1 Tof
  ∩ ∀ Tby.Professor ∩ = 1 Tby
- ENROLLING ⊆ ∀ Ein.Course ∩ = 1 Ein ∩ ∀ Eof.Student
  ∩ = 1 Eof
- Course ⊆ ∀ Tof-.TEACHING ∩ = 1 Tof-
  ∩ ∀ Ein.ENROLLING ∩ ≥ 10 Ein-
  ∩ ≤ 50 Ein-

DL vs EER

- AdvCourse ⊆ Course
- Professor ⊆ ∀ Tby-.TEACHING ∩ ∃ Tby-
- Student ⊆ ∀ Eof-.ENROLLING ∩ ≥ 3 Eof-
  ∩ ≤ 6 Eof-
- GradStudent ⊆ Student ∩ ∀ degree.String
  ∩ = 1 degree

DL vs OO

Class Customer type-is
  union BussinessCustomer, PrivateCustomer

Class PrivateCustomer is-a Customer type-is
  record
   SSN: String

Class Registration type-is
  record
   cust: Customer
   regis: set-of record
            serv: Service
            loc: Location

DL vs OO

AbstractClass ⊆ = 1 value
RecType ⊆ ∀ value. ⊥
SetType ⊆ ∀ value. ⊥ ∩ ¬ RecType

Class C is mapped to a concept M(C)
Union of classes is mapped to a disjunction of concepts
Set-of C is mapped to SetType ∩ ∀ member.M(C)
Attribute A is mapped to an atomic role M(A)
Record A_1:C_1, … is mapped to
  RecType ∩ ∀ $M(A_1).M(C_1)$ ∩ = 1 $M(A_1)$ …

DL vs OO

Customer ⊆ AbstractClass
  ∩ ∀ value.(BusinessCustomer U PrivateCustomer)

PrivateCustomer ⊆ AbstractClass ∩ Customer
  ∩ ∀ value.(RecType ∩ = 1 SSN ∩ ∀ SSN.String)

Registration ⊆ AbstractClass
  ∩ ∀ value.(RecType
    ∩ = 1 cust ∩ ∀ cust.Customer
    ∩ = 1 regis ∩ ∀ regis.(SetType
      ∩ ∀ member.(RecType ∩ = 1 serv ∩ ∀ serv.Service
      ∩ = 1 loc ∩ ∀ loc.Location)))

DL systems

- First generation
  - KL-ONE
    - Based on semantic networks tradition
    - concept names, constructors, roles, instances, subsumption
    - Informal specification of the language → logical reconstruction in 1983
    - Proven to be undecidable in 1989

DL systems

- First generation
  - KRYPTON
    - First attempt to define a new KL-ONE style language with formal semantics
    - Conjunction, value restriction, role chains → polynomial language
    - User should not know about implementation details → declarative

DL systems

- First generation
  - NIKL, KL-TWO
    - Role hierarchies
    - NIKL + assertional reasoning → KL-TWO
  - KANDOR
    - based on KRYPTON
    - Led to the design of CLASSIC

DL systems

- Second generation
  - CLASSIC
    - $\mathcal{ALNFO}$ + role inclusion + fills
    - incomplete reasoning
    - new semantics: individuals are mapped to disjoint sets of objects → complete reasoning
    - forward chaining rules
    - explanation
    - extension interface for concrete domains

DL systems

- Second generation
  - Loom
    - $\mathcal{ALCQRIFO}$ + constructs for real numbers
    - incomplete algorithm

DL systems

- Second generation
  - BACK, FLEX
    - incomplete algorithms
    - BACK: $\mathcal{ALQR}$- + reasoning with numbers
    - led to complexity results, semantics for cycles
    - incremental addition for Abox statements, also retraction
    - FLEX: $\mathcal{ALCQRIFO}$ + equations and inequalities concerning integers

DL systems

- Second generation
  - KRIS
    - $\mathcal{ALCNFO}$
    - sound and complete algorithm
    - experimental interface for concrete domains
    - tableau based
    - optimization

DL systems

- Second generation
  - CRACK
    - sound and complete algorithms for dealing with individuals in concept terms
    - web interface

DL systems

- Third generation
  - FaCT
    - only Tboxes
    - current version $\mathcal{SHIQ}$
    - sound and complete
    - optimized
    - Oiled

DL systems

- Third generation
  - DLP
    - $\mathcal{ALCNreg}$
    - sound and complete
    - highly optimized
    - comparable performance to traditional approaches in modal logics
    - no classification

DL systems

- Third generation
  - RACER
    - $\mathcal{SHIQ}$, concrete domains
    - Tbox + Abox
    - No incremental addition of Abox statements

Extensions

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

DL and concrete domains

- A concrete domain $\mathcal{D}$ consists of a set $\Delta^{\mathcal{D}}$ (the domain of $\mathcal{D}$) and a set *pred*($\mathcal{D}$), the predicates of $\mathcal{D}$. Each predicate name P $\in$ *pred*($\mathcal{D}$) is associated with an arity n, and an n-ary predicate $P^{\mathcal{D}} \subseteq (\Delta^{\mathcal{D}})^n$

Examples:

- Non-negative integers,
  binary predicates $<$, $>$, $\geq$, $\leq$ and unary predicates $<_n$, $>_n$, $\geq_n$, $\leq_n$
- Real numbers, Formulae built with first-order means (Boolean operators and quantifiers) from equalities and inequalities between integer polynomials in several indeterminates.
- Atomic values in a relational database,
  Relations that can be defined using SQL

DL and concrete domains

- A concrete domain $\mathcal{D}$ is admissible iff

(i) set of predicate names is closed under negation and contains a name $T_{\mathcal{D}}$ for $\Delta^{\mathcal{D}}$

(ii) the satisfiability problem for $\mathcal{D}$ is decidable (i.e. deciding satisfiability of finite conjunctions of predicates)

DL and concrete domains

- C,D → $\exists(u_1,\ldots,u_n).P$

For concrete domain the natural numbers:

Woman ∩

$\exists$(hasFather ○ hasAge, hasHusband ○ hasAge).<

DL and concrete domains

An interpretation $\mathcal{I}$ for $\mathcal{ALC}(\mathcal{D})$ consists of a set $\Delta^{\mathcal{I}}$ (the abstract domain of the interpretation) and an interpretation function $\cdot^{\mathcal{I}}$. The abstract domain and the concrete domain must be disjoint. The interpretation function assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every ordinary role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Functional roles are interpreted by partial functions from $\Delta^{\mathcal{I}}$ into $\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{D}}$.

DL and concrete domains

- C,D → $\exists(u_1,\ldots,u_n).P$

$(\exists(u_1,\ldots,u_n).P)^{\mathcal{I}}$
$= \{a \in \Delta^{\mathcal{I}} \mid \exists\, d_1,\ldots, d_n \in \Delta^{\mathcal{D}}:$
$u_1^{\mathcal{I}}(a) = d_1 \wedge \ldots \wedge u_n^{\mathcal{I}}(a) = d_n$
$\wedge (d_1,\ldots, d_n) \in P^{\mathcal{D}}\}$

DAML

- DARPA Mark-up Language
- Initial DAML web ontology language (DAML-ONT) released October 2000
- Builds on XML, RDF, RDFS
- Modeling primitives from OO and Frame systems
- (formal foundation)

OIL

- Core-OIL
  - inclusion axioms
  - range and domain constraints
- Standard-OIL
- Instance-OIL
  - Standard-OIL + Abox axioms
- Heavy-OIL
  - Not yet defined

Standard-OIL

- OIL: $\mathcal{SHIQ}(\mathcal{D})$
- Data types are integers, strings, sub-ranges and boolean combinations of these
- Semantics defined by a mapping from an OIL ontology to an equivalent $\mathcal{SHIQ}(\mathcal{D})$ terminology
- individuals are mapped to disjoint primitive concepts
  one-of is mapped to a disjunction of disjoint primitive concepts
  implicit unique name assumption
- XML and RDFS serializations

DAML+OIL

- DAML+ OIL
- more tightly integrated with RDFS:
  - use of RDFS machinery
  - portability
  - unintuitive semantics
  - undecidable
  - true individuals
  - no unique name assumption for individuals
  - data types (clean separation with object types)

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	$\top \sqsubseteq \leqslant 1P$	⊤ ⊑ ⩽1hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1P^-$	⊤ ⊑ ⩽1isMotherOf$^-$

- Axioms (mostly) reducible to subClass/PropertyOf

RP - OWL

- An expressive description logic
  Under development [Dean et al. (eds) 2002]
- Encoded in XML
- Extending (?) RDFS
  Some problems with that [Patel-Schneider, Fensel 2002]
  Ontology: a conceptual model of the application domain;
  OWL ontology: a set of OWL statements –
  XML encoding of a set of DL axioms

From introduction lecture

OWL

- OWL Full
  - All language constructs
  - Unconstrained use of RDF constructs
  - owl:Class is equivalent to rdfs:Class
  - Classes can be treated as individuals
  - All data values are in the individual domain
  - owl:ObjectProperty is equivalent to rdf:Property
  - Expressivity of OWL and flexibility of RDF
  - Reasoning guarantees?

OWL

- OWL DL
  - Separation between classes, data types, data type properties, object properties, individuals, data values
  - No inverses, symmetry and transitivity for data type properties
  - No cardinality constraints on transitive properties or their inverses or super-properties
  - Most RDF(S) vocabulary cannot be used
  - All axioms must be well-formed and must form a tree-like structure
  - Facts about equality and difference of individuals must be about named individuals

OWL

- OWL Lite
  - Forbidden to use: oneOf, unionOf, complementOf, hasValue, disjointWith, DataRange
  - Other restrictions

OWL-DL

- OWL DL = $\mathcal{SHIQO}$ + data types
- Satisfiability in $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ is ExpTime-complete
- $\mathcal{SHIQO}$ is NExpTime-hard
- No algorithm for $\mathcal{SHIQO}$ yet, but for $\mathcal{SHIQ}$ and for $\mathcal{SHOQ}$
- $\mathcal{SHIQ}(\mathcal{D})$: ExpTime-complete, implementation, well understood
- $\mathcal{SHOQ}(\mathcal{D})$: decidable, exact complexity not known, currently being investigated

→ no complete algorithm for OWL DL yet

OWL vs DAML+OIL

- DAML+OIL closest to OWL DL
- No qualified restrictions in OWL
- Renaming of properties and classes
- owl:symmetricProperty
- Versioning constructs in owl
- owl:allDifferent and owl:distinctMembers for addressing the unique name assumption

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)