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

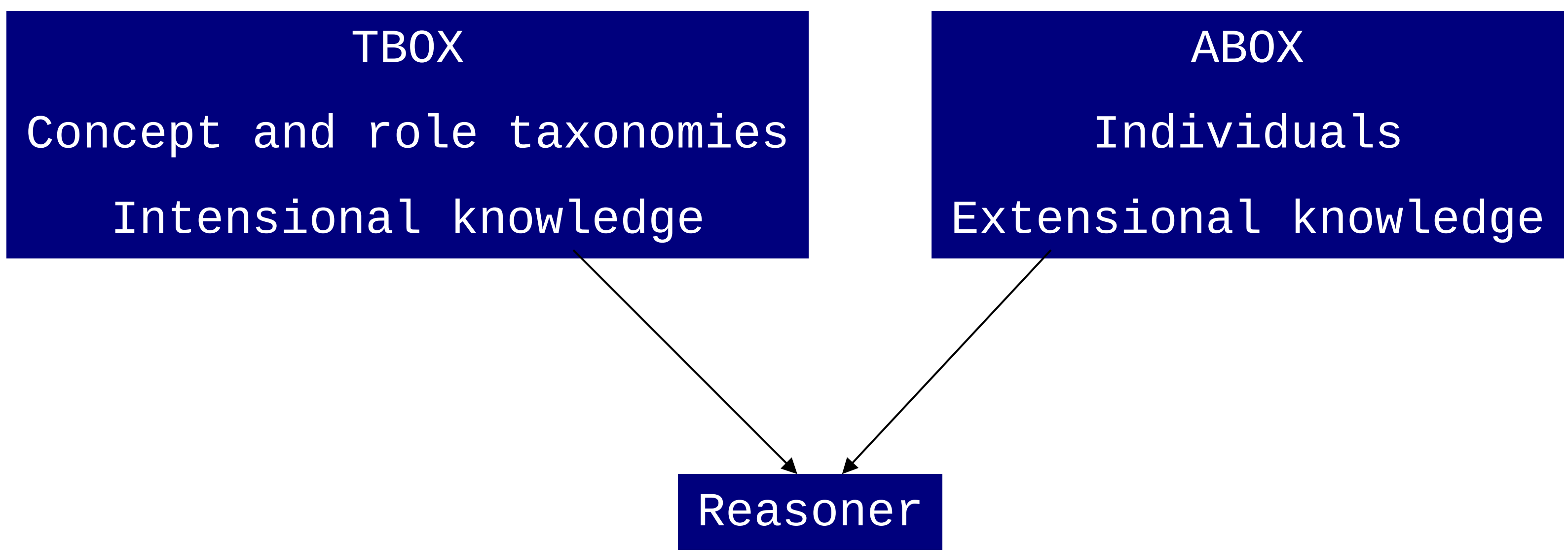
Ontologies, Description Logics and OWL terminology

Ontologies	DL	OWL
concept	concept	class
relation	role (binary)	property
axiom	axiom	axiom
instance	individual	individual

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)

$\perp$ | (bottom concept)

$\neg A$ | (atomic negation)

$C \cap D$ | (conjunction)

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

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

$\mathcal{AL}[X]$

$\mathcal{C}$ ¬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 $\cap \geq 10$ hasMember

Team $\cap \geq 11$ hasMember
$\cap\ \forall$ 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:
  - $C = D$ ($R = S$)
    owl:equivalentClass / owl:equivalentProperty
  - $C \subseteq D$ ($R \subseteq S$)
    rdfs:subClassOf / rdfs:subPropertyOf
  - (disjoint C D)
    owl:disjointWith

Tbox

- 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)

Example

Ida-member(Sture)

Individuals in the description language

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

Example

(S-Team $\cap$ 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 $\subseteq$ 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(Sture)

(S-Team $\cap$ hasMember:Sture)(IDA-FF)

Example - OWL

<Declaration> <ObjectProperty IRI="#hasmember"/> </Declaration>

<Declaration> <Class IRI="#soccer-player"/> </Declaration>
<Declaration> <Class IRI="#ida-member"/> </Declaration>
<Declaration> <Class IRI="#team"/> </Declaration>
<Declaration> <Class IRI="#large-team"/> </Declaration>
<Declaration> <Class IRI="#s-team"/> </Declaration>

<Declaration> <NamedIndividual IRI="#IDA-FF"/> </Declaration>
<Declaration> <NamedIndividual IRI="#Sture"/> </Declaration>

Example - OWL

```
<EquivalentClasses>
    <Class IRI="#large-team"/>
    <ObjectIntersectionOf>
        <Class IRI="#team"/>
        <ObjectMinCardinality cardinality="10">
            <ObjectProperty IRI="#hasmember"/>
        </ObjectMinCardinality>
    </ObjectIntersectionOf>
</EquivalentClasses>
```

Example - OWL

```
<EquivalentClasses>
    <Class IRI="#s-team"/>
    <ObjectIntersectionOf>
        <Class IRI="#team"/>
        <ObjectAllValuesFrom>
            <ObjectProperty IRI="#hasmember"/>
            <Class IRI="#soccer-player"/>
        </ObjectAllValuesFrom>
        <ObjectMinCardinality cardinality="11">
            <ObjectProperty IRI="#hasmember"/>
        </ObjectMinCardinality>
    </ObjectIntersectionOf>
</EquivalentClasses>
```

Example - OWL

```
<ClassAssertion>
    <ObjectIntersectionOf>
        <Class IRI="#s-team"/>
        <ObjectHasValue>
            <ObjectProperty IRI="#hasmember"/>
            <NamedIndividual IRI="#Sture"/>
        </ObjectHasValue>
    </ObjectIntersectionOf>
    <NamedIndividual IRI="#IDA-FF"/>
  </ClassAssertion>

<ClassAssertion>
    <Class IRI="#ida-member"/>
    <NamedIndividual IRI="#Sture"/>
  </ClassAssertion>
```

$\mathcal{AL}$ (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 $C \subseteq D$ 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*

$\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ 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 ∩ ∀ 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

DG = EmptyDG U Non-Empty-DG

Non-Empty-DG = Node ∩ ∀ Arc.Non-Empty-DG

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

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

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

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

$\rightarrow$ 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)

$\neg$ patricide(Thersandros) ***(not represented in DB)***

Does it follow from the Abox that

$\exists$hasChild.(patricide $\cap$ $\exists$hasChild. $\neg$ 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 $\cap$ 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 $C \cap \neg D$ is unsatisfiable
  - C and D are equivalent iff

    both $(C \cap \neg D)$ and $(D \cap \neg C)$ are unsatisfiable
  - C and D are disjoint iff $C \cap D$ 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.
  - $S = \{ x: \neg C \cap D \}$
  - Add constraints according to a set of propagation rules
  - Until clash or no constraint is applicable

Tableau algorithms – de Morgan rules

$\neg \neg C \rightarrow C$

$\neg (A \cap B) \rightarrow \neg A \cup \neg B$

$\neg (A \cup B) \rightarrow \neg A \cap \neg B$

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

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

Tableau algorithms – constraint propagation rules

- $S \rightarrow_{\cap} \{x:C_1, x:C_2\} \cup S$

  if $x: C_1 \cap C_2$ in S

  and either $x:C_1$ or $x:C_2$ is not in S

- $S \rightarrow_{U} \{x:D\} \cup S$

  if $x: C_1 \cup C_2$ 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

- $S \rightarrow_{\forall} \{y{:}C\} \cup S$

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

- $S \rightarrow_{\exists} \{xRy, y{:}C\} \cup S$

  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

  $\cap$ $\exists$ hasParticipant.Swedish
- SBT: Tournament

  $\cap$ $\exists$ hasParticipant.(Swedish $\cap$ Belgian)

Example 1

- SBT => ST?
- S = { x:

$\neg$(Tournament $\cap$ $\exists$ hasParticipant.Swedish)

$\cap$ (Tournament

$\cap$ $\exists$ hasParticipant.(Swedish $\cap$ Belgian))

}

Example 1

- S = { x:

  (¬Tournament

  U ∀ hasParticipant.¬ Swedish)

  ∩ (Tournament

  ∩ ∃ hasParticipant.(Swedish ∩ Belgian))

  }

Example 1

$\cap$-rule:

- S = {

  x: ($\neg$Tournament

  $\sqcup$ $\forall$ hasParticipant.$\neg$ Swedish)

  $\cap$ (Tournament

  $\cap$ $\exists$ hasParticipant.(Swedish $\cap$ Belgian)),

  x: $\neg$Tournament

  $\sqcup$ $\forall$ hasParticipant.$\neg$ Swedish,

  x: Tournament,

  x: $\exists$ hasParticipant.(Swedish $\cap$ Belgian)

  }

Example 1

$\exists$ -rule:

- S = {
  x: ($\neg$Tournament U $\forall$ hasParticipant.$\neg$ Swedish)
  $\cap$ (Tournament
  $\cap$ $\exists$ hasParticipant.(Swedish $\cap$ Belgian)),
  x: $\neg$Tournament
  U $\forall$ hasParticipant.$\neg$ Swedish,
  x: Tournament,
  x: $\exists$ hasParticipant.(Swedish $\cap$ Belgian),
  x hasParticipant y, y: (Swedish $\cap$ Belgian)
  }

Example 1

$\cap$-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:

$\neg$ (Tournament

$\cap \exists$ hasParticipant.(Swedish $\cap$ Belgian))

$\cap$ (Tournament $\cap \exists$ 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

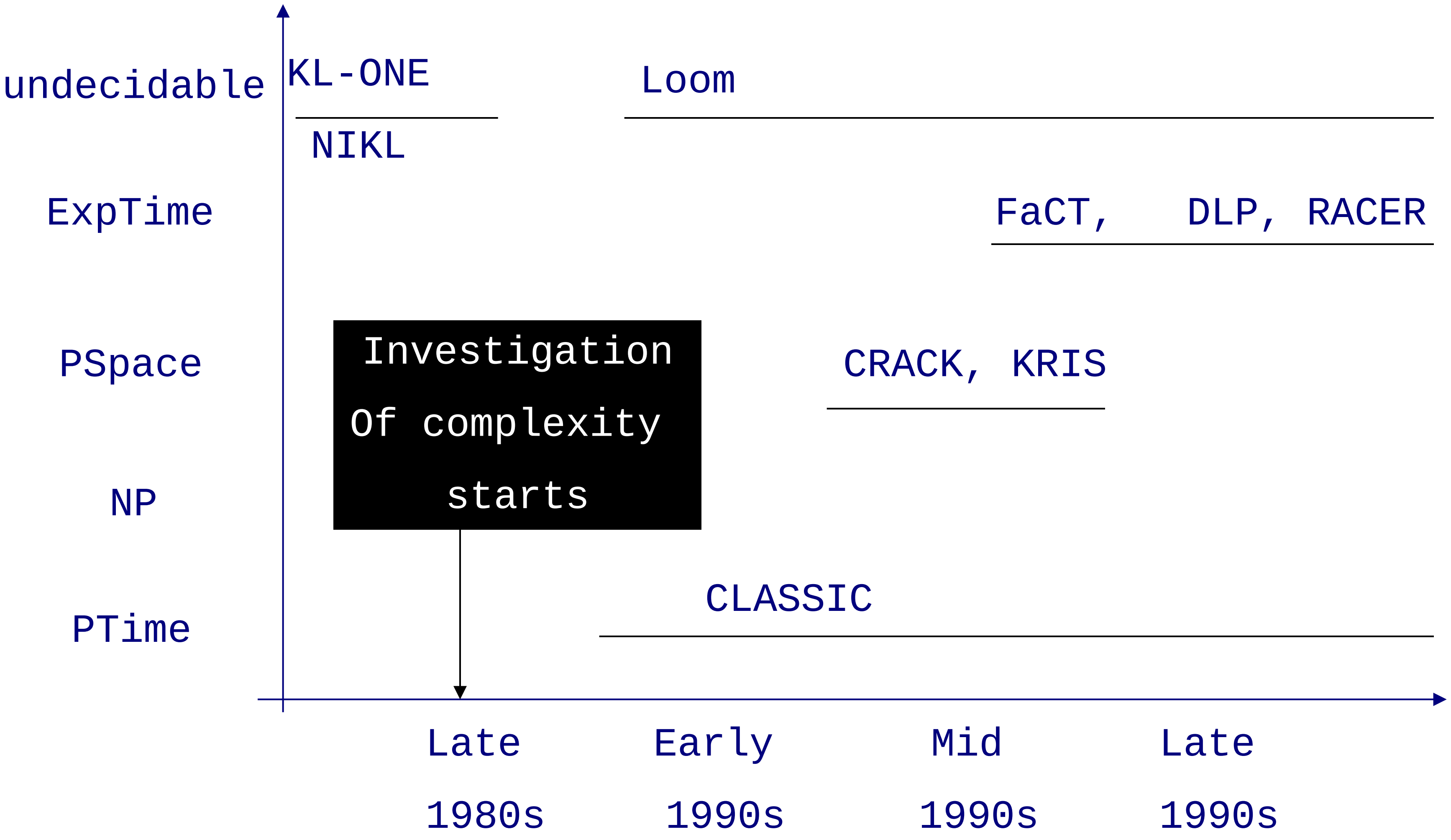
exponential size models,

e.g interplay universal and existential quantification

Reasons for undecidability:

e.g. role-value maps R=S

Systems


undecidable
KL-ONE
NIKL
Loom
ExpTime
FaCT, DLP, RACER
PSpace
Investigation
Of complexity
starts
CRACK, KRIS
NP
PTime
CLASSIC
Late
1980s
Early
1990s
Mid
1990s
Late
1990s

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 $\sqcap$ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	$\neg$Male
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}
toClass	$\forall P.C$	$\forall$hasChild.Doctor
hasClass	$\exists P.C$	$\exists$hasChild.Lawyer
hasValue	$\exists P.\{x\}$	$\exists$citizenOf.{USA}
minCardinalityQ	$\geqslant n P.C$	$\geqslant$2hasChild.Lawyer
maxCardinalityQ	$\leqslant n P.C$	$\leqslant$1hasChild.Male
cardinalityQ	$= n\ P.C$	$=$1 hasParent.Female

- XMLS **datatypes** as well as classes
- Arbitrarily complex **nesting** of constructors
  - E.g., Person $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)

DAML+OIL Axioms

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

☞ 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*
- <u>intersectionOf</u>, *unionOf, complementOf*
- **Property**, subPropertyOf, equivalentProperty
- domain, range (global restrictions)
- inverseOf, TransitiveProperty (*), SymmetricProperty, FunctionalProperty, InverseFunctionalProperty
- allValuesFrom, someValuesFrom (local restrictions), *oneOf*, *hasValue*
- <u>minCardinality, maxCardinality</u>
- **Individual**, sameAs, differentFrom, AllDifferent

(*) restricted

OWL2

- OWL2 Full and OWL2 DL
- OWL2 DL compatible with SROIQ

- Punning
  - IRI may denote both class and individual
  - For reasoning they are considered separate entities

OWL2 profiles

- OWL2 EL (based on EL++)
  - Essentially intersection and existential quantification
  - SNOMED CT, NCI Thesaurus
- OWL2 QL ("query language")
  - Can be realized using relational database technology
  - RDFS + small extensions
- OWL2 RL ("rule language")

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

References

- https://www.w3.org/TR/2004/REC-owl-features-20040210/
- https://www.w3.org/TR/2012/REC-owl2-quick-reference-20121211/
- https://www.w3.org/TR/2012/REC-owl2-primer-20121211/