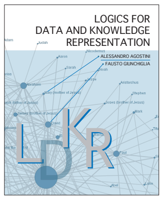
Logics for Data and Knowledge Representation

Alessandro Agostini
agostini@dit.unitn.it

Fausto Giunchiglia
fausto@dit.unitn.it

University of Trento

The Logic of Descriptions



- Introduction
- Language (Syntax)
- Semantics
  - interpretation
  - entailment
- Knowledge Bases
- Reasoning Services

Overview

- Description Logics (DLs) is a family of KR formalisms that represent the knowledge of an application domain ("the world") by
  - defining the relevant concepts of the domain (i.e., its terminology), and then
  - using these concepts to specify the properties of objects in the domain (i.e., the world's description).

Terminology

- Previously used names for DLs are:
  - terminological knowledge representation languages,
  - concept languages,
  - term subsumption languages,
  - Kl-One-based knowledge representation languages.

Some History on DLs

- Descended from the "structured inheritance networks" (Brachman, 1977).
- Introduced to overcome the ambiguities of early semantic networks and frames.
- First realized in the system Kl-One by Brachman and Schmolze (1985).
  - First DL presented in the B & S's paper.

Three Basic Features

1. The basic syntactic building blocks are atomic concepts, atomic roles, individuals.
2. The expressive power of DLs is restricted to a rather small set of constructors for building complex concepts and relations.
3. Implicit knowledge about concepts and individuals can be inferred automatically with the help of specific reasoning services.

Language

Language (Syntax)

- The first step in setting up a formal language (viz. a descriptive language) is to list the symbols, that is, the alphabet of symbols.
- We denote a generic alphabet of a descriptive (or 'description') language: dΣ.
- Similarly to any logical language we can divide symbols in dΣ in 'descriptive' (nonlogical) and 'non-descriptive' (logical).

Language (Syntax)

- Descriptive dΣ consists of concept names (set **C**), which denote sets of individuals, role names (set **R**), which denote binary relations between individuals, and individual names, (set **I**), which denote individuals.
- Example:
  concept names: Room, Person, Fruit
  role names: likeSkiing, hasChild, partOf, isA,..
  individual names: I, you, apple, Fido, ...

DL Language and Previous Languages

- concept names are propositional variables
  - (PL/ClassL)
- role names are binary predicate symbols
  - (FOL)
- individual names are constants
  - (FOL)

Remark FOL versus DLs

- Strictly speaking we do not need DLs to represent concepts and roles,
  - but the variable-free syntax of DLs is much more concise!
    - That's good for automation!!

Language (Syntax)

- Non-descriptive dΣ provides concept constructors to build complex formulas, called concept descriptions and role descriptions, from atomic formulas.
- Example:

  ¬ (negation), ⊓ (conjunction)
  ∀ (for all), ∃ (there exists)

AL-family Languages

- We shall now discuss various descriptive languages from the family of AL-languages.
- An AL-language (= Attributive Languages) is a minimal DL language of practical interest.
- More expressive descriptive languages are usually extensions of some AL-language.
- AL-languages do not deal with individuals.

AL Logical Symbols

- 1. Universal concept symbol: $\top$.
  2. Bottom concept symbol: $\bot$.
- 3. Logical constants (concept constructors):
  $\neg$ (atomic negation), $\sqcap$ (conjunction)
  $\forall$R (for all atomic roles)
  $\exists$R (there exists an atomic role)
- 4. Parentheses (auxiliary symbols): (,)

AL Non-logical Symbols

- 5. Atomic concept names: A, B, ...
- 6. Atomic role names: R (generic)
- 7. Concept names: C, D, ...
- Remark. There is no logical symbol in *AL* for logical implication (as '$\rightarrow$' in PL and in FOL). For, we will use the subsumption symbol '$\sqsubseteq$' instead (as in classL).

Defined Symbols

- Similarly to ClassL, $\top$ and $\bot$ can be *defined*:
  - For all concept names C,
    - $\bot =_{df} C \sqcap \neg C$
  - $\top =_{df} \neg\bot$ or also $\top =_{df} U$

    for U be a special coincept name denoting the Universal Concept.
- We prefer to consider $\top$ and $\bot$ *AL*' symbols.

Formation Rules for *AL*

- Atomic Concepts: 1. A, B,..., $\bot$, $\top$.
- Concepts (concept descriptions):
  2. All the atomic concepts
  3. $\neg$A for A (atomic concept negation)
  4. C$\sqcap$D (intersection)
  5. $\forall$R.C (value restriction)
  6. $\exists$R.$\top$ (limited existential quantification)
- Resulting language: attributive language (*AL*).

Examples

- Atomic concepts: Person, Female, Room, ...
  Atomic roles: hasChild, partOf, isIn, isA,...
- Concepts: Person $\sqcap$ Female,
  Person $\sqcap$ $\exists$hasChild.$\top$ ($\exists$hasChild)
  Person $\sqcap$ $\forall$hasChild.$\bot$ (Not:$\neg\exists$hasChild.$\top$)
  Person $\sqcap$ $\forall$hasChild.$\neg$Female
- Question: What is the *intended* meaning?

Examples (cont')

- Person ⊓ Female "persons that are female"
- Person ⊓ ∃hasChild.⊤

  "(all those) persons that have a child"
- Person ⊓ ∀hasChild.⊥

  "(all those) persons without a child"
- Person ⊓ ∀hasChild.Female

  "persons all of whose children are female"

AL's Extensions - *ALU*

- Extended Alphabet:
  Logical constants (concept constructors):
  ⊔ (disjunction).
- Extended concepts (descriptions):
  C ⊔ D (union)
- The resulting new language (i.e. *AL* plus the new set of concepts) usually denoted *ALU*.

AL's Extensions - *ALE*

- Extended Alphabet:
  Logical constants (concept constructors):
  ∃R (there exists an arbitrary role)
- Extended concepts (descriptions):
  ∃R.C (full existential quantification)
- The resulting new language (i.e. *AL* plus the new set of concepts) usually denoted *ALE*.

AL's Extensions - *ALN*

- Extended Alphabet:
  Logical constants (concept constructors):
  ≥n, ≤n for all n ∈ N (at-least/at-most n)
- Extended concepts (descriptions):
  ≥nR (at-least number restriction)
  ≤nR (at-most number restriction)
- The resulting new language (i.e. *AL* plus the new set of concepts) usually denoted *ALN*.

AL's Extensions - *ALC*

- Extended Alphabet:
  Logical constants (concept constructors):
  ¬ (general negation)
- Extended concepts (descriptions):
  ¬C (full concept negation)
- The resulting new language (i.e. *AL* plus the new set of concepts) usually denoted *ALC*. (C stands for "Complement").

AL's Extensions (Summary)

- Extending *AL* by any subsets of the above constructors yields a particular DL language.
- Each language is denoted by a string of the form *AL*[*U*][*E*][*N*][*C*], where a letter in the name stands for the presence of the corresponding constructor. Notation: AL*.
- *ALC* as the most important in many aspects. (We'll see that *ALU* ⊆ *ALC* and *ALE* ⊆ *ALC*.)

AL's Contractions

- Contracting *AL* by eliminating any subsets of symbols yields a particular DL language.
- The most important language obtained by contraction of a language in the AL family is the language of class logic (see next slide).
- Historically, another important contraction defines the Frame Language *FL0*.

ALC's Contraction: The Language of ClassL

- Contracted Alphabet (w.r.t. *ALUEC!*): Logical constants (concept constructors): ∀R, ∃R (quantifiers on arbitrary roles)
- Contracted concepts (descriptions): ∀R.C, ∃R.C (∀,∃ quantifications)
- The new language is a propositional description language. Such language is exactly our class propositional language.

AL's Contractions: *FL-*

- Contracted Alphabet (w.r.t *AL*'s alphabet): Universal and bottom symbols: ⊤, ⊥.

  Logical constants (concept constructors): ¬ (atomic negation)
- Contracted concepts (descriptions): ⊤, ⊥ ,¬A (atomic negation)
- The resulting new language (i.e. *AL* without the contracted concepts) is denoted: *FL-*.

AL's Contractions: *FL0*

- Contracted Alphabet (w.r.t *FL-*'s alphabet): Logical constants (concept constructors): ∃R (there exists an atomic role)
- Contracted concepts (descriptions): ∃R.⊤ (limited existential quantification)
- The resulting new language (i.e. *FL-* without the contracted concepts) is denoted: *FL0*.
- FL = Frame Language (for historical reasons)

Summary

- DLs are a family of logic-based KR formalisms to describe a domain in terms of - concepts - roles - individuals ("grounding")
- Strictly speaking we do not need DLs to represent concepts and roles, but the variable-free syntax of DLs is much more concise! (That's good for automation!)
- Class language is *ALC* without quantification.

Semantics

Semantics

- The elements of the description languages in AL-family (AL*) are plain strings of symbols

without a <u>formal</u> meaning

- The meaning which is intended to be attached to concept, role, and individual names form an *informal* interpretation of the given AL* language's expressions.

DL Interpretations AL* Interpretation (Δ,I)

- DL languages (AL* for us now) have a formal semantics defined in terms of DL interpretations over a domain of "objects".
- Definition. An <u>interpretation</u> of an AL* language L is a pair *I* = (Δ,I), where:
  - Δ (<u>domain</u>) is a non-empty set of objects
  - I (<u>interpretation function</u>) is a mapping from L to Δ defined as follows.

(see the next slide)

AL* Interpretation (Δ,I) (Definition, cont')

- I(⊥) = ∅ and I(⊤) = Δ (domain, "Universe")
- For every individual name *a* of L, I(*a*) ∈ Δ.
- For every concept name A of L, I(A) ⊆ Δ.
- For every role name R of L, I(R) ⊆ Δ × Δ.
- I(¬C) = Δ \ I(C).
- I(C⊓D) = I(C) ∩ I(D); I(C ⊔D) = I(C) ∪ I(D).

AL* Interpretation (Δ,I) (Definition, cont')

- I(∀R.C) =

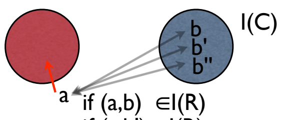
  = {a ∈ Δ | for all b, if (a,b)∈I(R) then b∈I(C)}.
- I(∃R.⊤) = {a ∈ Δ | exists b s.t. (a,b) ∈ I(R)}.
- I(∃R.C) =

  = {a ∈ Δ | exists b s.t. (a,b) ∈ I(R), b ∈ I(C)}.
- I(≥nR) = {a ∈ Δ | |{b | (a, b) ∈ I(R)}| ≥ n}.
- I(≤nR) = {a ∈ Δ | |{b | (a, b) ∈ I(R)}| ≤ n}.

AL* Interpretation (Δ,I)

- I(∀R.C) =

  = {a ∈ Δ | for all b, if (a,b)∈I(R) then b∈I(C)}



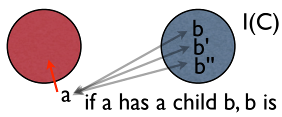
if (a,b) ∈I(R)
if (a,b') ∈I(R)
if (a,b'') ∈I(R)

- Remark: a ∈ Δ

Example

- I(∀R.C) = I(∀hasChild.Female) =

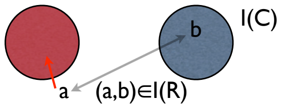
  = {a ∈ Δ | for all b, if (a,b)∈I(R) then b∈I(C)}



if a has a child b, b is a female
if a has a child b', b' is a female
if a has a child b'', b'' is a female
if ...

AL* Interpretation (Δ,I)

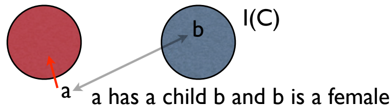
- I(∃R.C) =
  = {a ∈ Δ | exists b s.t. (a,b) ∈ I(R), b ∈ I(C)}



- Remark: a ∈ Δ

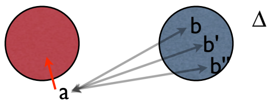
Example

- I(∃R.C) = I(∃hasChild.Female) =
  = {a ∈ Δ | exists b s.t. (a,b) ∈ I(R), b ∈ I(C)}



AL* Interpretation (Δ,I)

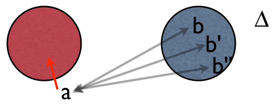
- I(≥nR) =
  = {a ∈ Δ | ||{b : (a, b) ∈ I(R)}|| ≥ n }



||{b, b', b''... : (a,b) ∈ I(R)}|| ≥ n
(a,b') ∈ I(R)
(a,b'') ∈ I(R)

Example

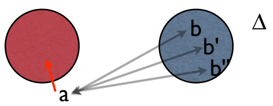
- I(≥nR) = I(≥2hasChild) =
  = {a ∈ Δ | ||{b : (a, b) ∈ I(R)}|| ≥ n }



a's children {b, b', b''...} are ≥ 2

AL* Interpretation (Δ,I)

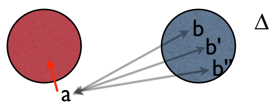
- I(≤nR) =
  = {a ∈ Δ | ||{b : (a, b) ∈ I(R)}|| ≤n }



||{b, b', b''... : (a,b) ∈ I(R)}|| ≤n
(a,b') ∈ I(R)
(a,b'') ∈ I(R)

Example

- I(≤nR) = I(≤2hasChild) =
  = {a ∈ Δ | ||{b : (a, b) ∈ I(R)}|| ≥ n }



a's children {b, b', b''...} are ≤ 2

Simple Exercise

- Verify the following equivalences hold for all interpretations (Δ,I):
  - I(¬(C⊓D)) = I(¬C ⊔¬D)
  - I(¬(C⊔D)) = I(¬C⊓¬D)
  - I(¬∀R.C) = I(∃R.¬C)
  - I(¬∃R.C) = I(∀R.¬C)

Concept Equivalence

- Definition. Concepts C, D are equivalent (C≡D), if I(C) = I(D) for all interpretations I.
- Example:
  1. ∀hasChild.Female ⊓∀hasChild.Student
  2. ∀hasChild.Female ⊓ Student
- Exercise: Prove that 1 and 2 are equivalent.
- Notation: For I = (Δ,I), C^I in place of I(C).

Exercise (Solution)

- I(∀hasChild.F ⊓∀hasChild.S) =
  = I(∀hasChild.F) ∩ I(∀hasChild.S)
  = {a∈Δ | for all b, if (a,b)∈I(hasChild) then
  b∈I(F)} ∩ {a∈Δ| for all b, if (a,b)∈I(hasChild)
  then b∈I(S)}
  = {a ∈ Δ | for all b, if (a,b)∈I(hasChild) then
  b∈I(F ⊓ S)}

Correspondence Theorems

DL vs. FOL Concepts as Predicates

- The semantics of concepts identifies DL (AL*) languages as fragments of FOL:
  - a DL interpretation (Δ,I) assigns to every atomic concept or role a unary or binary relation over Δ, respectively,
  - so one can think of atomic concepts and roles as unary and binary predicates.

DL vs. FOL (cont')

- Strictly speaking we do not need DLs to represent concepts and roles, but the variable-free syntax of DLs is much more concise! (That's good for automation!)
- Any concept description C can be translated effectively into a predicate logic formula C(x), which has one free variable, such that for all (Δ,I), the set of elements of Δ satisfying C(x) is exactly I(C).

Correspondence to FOL (I)

We define an effective mapping τ from AL-concepts to FO-formulas (wffs) as follows:

i. $\tau(\bot) = \bot$, ii. $\tau(\top) = \top$

iii. $\tau(A) = A(x)$ (A atomic, x free variable in A)

iv. $\tau(\neg C) = \neg\tau(C)$

v. $\tau(C \sqcap D) = \tau(C) \wedge \tau(D)$

vi. $\tau(C \sqcup D) = \tau(C) \vee \tau(D)$

Correspondence to FOL (2)

vii. Let C(x) a wff (x the only free variable).

$\tau(\forall R.C) = \forall x(R(y, x) \rightarrow C(x))$ (y new variable)

viii. Let C(x) a wff (x the only free variable).

$\tau(\exists R.C) = \exists x(R(y, x) \wedge C(x))$ (y new variable)

ix. ... [see the next slide]

Correspondence to FOL (3)

ix. $\tau(\geq nR)$ =

$\exists y_1 \ldots \exists y_n R(x, y_1) \wedge \ldots \wedge R(x, y_n) \bigwedge_{i<j} \neg(y_i \doteq y_j)$.

Note: "$\doteq$" is needed

x. $\tau(\leq nR)$ = ...

- Exercise: define $\tau(\leq nR)$.

Correspondence to FOL (4)

- Definition. An AL-concept C is coherent if there is an interpretation (Δ, I) s.t. I(C) is nonempty. (Δ, I) is called a model of C.
- Theorem. For every AL-concept C, C is coherent iff FO-formula $\tau(C)$ is satisfiable (i.e. $\tau(C)$ has a FO-model).

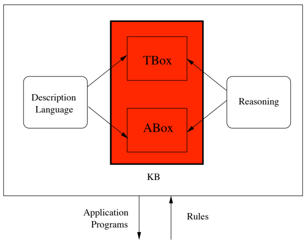
  Proof: Immediate from the definition of τ and the semantics of AL*.

Knowledge Bases TBox + ABox

Knowledge and DL-KB Systems

- In PL, ClassL, and FOL a **KB** is a theory, i.e. a set of propositions / closed predicates
- The first question to answer for a DL-based KB system is: what is a DL-**KB**?
  - if sentence = *concept* then a DL-**KB** is a "DL-theory" i.e. a set of concepts. True?
- Strictly speaking: No. Conceptually: Yes.

DL-based KB system (Architecture)



(Baader & Nutt, 2002)

DL Knowledge Base (TBox)

- Definition. A DL knowledge base is a pair **KB** = (TBox, ABox), where:

  i. TBox, called terminological box, is a finite set of 'expressions' *describing* concepts and roles hierarchies, i.e. relations between concepts and relations between roles).

  ii. see the next slide

DL Knowledge Base (Abox)

- Definition (cont'). A DL knowledge base is a pair **KB** = (TBox, ABox), where:

  i. see the previous slide

  ii. ABox, called assertional box, is a finite set of '*ground* expressions' *asserting* the relations between individuals and concepts or roles.

DL Knowledge Base (Three Remarks)

- **Remark 1.** A TBox expresses intensional knowledge about concepts and relations.
- **Remark 2.** An ABox expresses extensional knowledge about individual objects.
- **Remark 3.** Because of an ABox refers to individual objects (of the domain Δ), the expressions of an ABox are *grounded* (on Δ).

Terminological Box (TBox)

TBox Definition

- Definition. A TBox is a set of expressions, called terminological axioms, of this forms:

  ***General* inclusion axioms**:
  $C \sqsubseteq D$ | concept inclusion
  Role axioms:
  $R \sqsubseteq S$ | role inclusion
  Equivalence axioms:
  $C \equiv D$ ($R \equiv S$) | concept/role equivalence

TBox Example

- ***General* inclusion axioms**:
  C⊑D: Arm⊑∃isPartOf.Body,
  Body⊑∃isDirectPartOf.Human⊓≤2hasArm
- **Role axioms**:
  R⊑S: isDirectPartOf ⊑ isPartOf
- **Equivalence axioms**:
  C≡D: Men ≡ Person⊓Male

A TBox for UniTn (Example)

- TBox = { DISI ⊑ ∃isPartOf.UniTn,
  UniTn ⊑ ∃isPartOf.ItalianEdu,
  UniTn ⊑ ≤4hasLocation,
  DISI ≡ exDIT
  FacScience ≡ {DISI} ⊔ {MAT} ⊔ {PH} ⊔ ...
  DISI ≡ Research ⊓ Education ⊓ ¬Profit }

TBox (Descriptive) Semantics

- Definition. (1) A DL interpretation (Δ,I) satisfies
  i. C ⊑ D (R ⊑ S) if I(C) ⊆ I(D) (I(R) ⊆ I(S));
  ii. C ≡ D (R ≡ S) if I(C) = I(D) (I(R) = I(S)).
  (2) (Δ,I) satisfies a Tbox T if (Δ,I) satisfies all the axioms in T.
- **Remark.** Thus, semantically, we have that C ⊑ D and D ⊑ C iff C ≡ D for all C,D.

TBox - Definitions

- Definition. Equivalence axioms of the form A≡C with A *atomic* are called definitions.
- **Remark 1.** Definitions are used to introduce symbolic names to shorten complex descriptions (abbreviations).
- **Remark 2.** Definitions are typical of frame systems from which DLs originate.
  [Think of *FL*- and *FL0* languages (1984).]

Taxonomy (Definition)

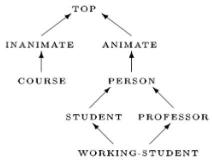
- From the Greek: τάξις, *taxis*, i.e. 'order'
- A taxonomy (of concepts) T is a *partially ordered* set (of concepts) such that:
  (a) there is no more than one *definition* for a concept in T,
  (b) each *definition* is acyclic, i.e., concepts in T are neither defined in terms of themselves nor in terms of other concepts that refer to them via a chain of definitions.

DL Taxonomy

- A taxonomy is the minimal relation in the space of concepts s.t. its reflexive-transitive closure is the subsumption relation.

DL Taxonomy

- A DL taxonomy is a taxonomy of concepts ordered by a subsumption relation.
- Example (Franconi):



Note the arrows' direction - there are no cycles!

DL Taxonomy (Example)

- A taxonomy of docs policy at DIT is:
  { ICT $\sqsubseteq$ UniTn, Student $\sqsubseteq$ ICT, Faculty $\sqsubseteq$ ICT, Student $\equiv$ PhD $\sqcup \neg$College,

  Public $\sqsubseteq$ DIT, Internal $\sqsubseteq$ DIT, Internal $\equiv \neg$Public;

  ICT $\sqsubseteq \exists$readDocs.DIT;

  Student $\sqsubseteq \forall$readDocs.Public}

Assertional Box (ABox)

ABox Definition

- Definition. An ABox is a set of expressions, called individual axioms or assertions:

  C(*a*) | concept assertion
  R(*b*, *c*) | role assertion

  where *a*, *b*, *c* are individual names.
- Alternative notation:
  '*a* : C' for C(*a*), '(*b*, *c*) : R' for R(*b*, *c*).

ABox Example

- **Concept assertions**:

  C(*a*): Men(John)
- **Role assertions**:

  R(*b*, *c*): isPartOf(head,John)

ABox (Descriptive) Semantics

- Definition. (1) A DL interpretation (Δ,I) satisfies
  i. C(*a*) if I(*a*) $\in$ I(C) (notation: I |= C(*a*))

  ii. R(*b*, *c*) if (I(b),I(c)) $\in$ I(R) (I |= R(*b*,*c*)).

  (2) (Δ,I) satisfies an Abox if (Δ,I) satisfies all its (concept, role) assertions.
- **Unique Name Assumption**: For all individual names *a*, *b*, if *a*$\neq$*b* then I(*a*)$\neq$I(*b*).

DL-KB Semantics

- Definition.
  (1) A DL interpretation (Δ,I) satisfies a DL knowledge base KB if (Δ,I) satisfies all the (terminological, individual) axioms in KB.

  (Δ,I) |= KB

  (2) (Δ,I) is a model of KB if it satisfies KB.

DL-KB Semantics

- Definition (cont').
  (3) A DL knowledge base KB is satisfiable if there is a model of KB (i.e. KB has a model).

  (4) KB is unsatisfiable if it is not satisfiable.

- **Example:**
  KB = {∀toBe.X⊓¬∀toBe.X, X≡Human}

  is unsatisfiable.

DL (AL*) Entailment

- Definition. Let a (concept, role, individual) axiom ψ of a DL (AL*) language be given. A DL (AL*) knowledge base KB entails (also: logically implies) ψ if *every* model of KB satisfies ψ.

  KB |= ψ.

- ***Be aware:***
  '|=' used in both (Δ,I) |= ψ and KB |= ψ!