October 10, 2018

CSCI – Advance AI

DESCRIPTION LOGICS

Reference: Handbook of Knowledge Representation

Submitted by: Anoop Mishra

My job for presentation -

- Introduction of Description Logics
- Concepts of Description Logics
- Application of Description Logics with Demo

Overview

- Introduction
  - *History*
  - *Importance in Artificial Intelligence*
- Domain Languages
  - *Attribute languages including concepts*
  - *Approaches*
- Web Ontology language & DLs
- Application and Demo

Importance of Description Logics in AI/C.S.

- DLs are used in **artificial intelligence** to describe and reason about the relevant concepts of an application domain (known as *terminological knowledge*).

- There are general, spatial, temporal, spatiotemporal, and fuzzy descriptions logics, and each description logic features a different balance between DL expressivity and reasoning complexity by supporting different sets of mathematical constructors.

- Example: DLs and OWL is in **biomedical informatics** where DL assists in the codification of biomedical knowledge.

Description Logics

- Description logics (DLs): family of knowledge representation languages, used to represent the knowledge of an application domain.

- in terms of –

  concepts (classes),

  roles (properties and relations) and

  individuals (instances of classes).

Description Logics

- The important notions of the domain: descriptions, i.e., expressions built from **atomic concepts (unary predicates)** and **atomic roles (binary predicates)** using the concept and role constructors provided by the particular DL.

- DLs differ from their predecessors, such as semantic networks and frames, as they are equipped with a formal, logic-based semantics.

GENERAL VIEW

- In computer applications, to have access to symbolic models of an application world described in terms of individuals and related to other individuals and classes of individuals by relationships -

e.g., CSCI 8110 that are grouped into classes

COURSE, TEACHER, STUDENT

CSCI 8110 taught by TEACHER (conceptual structuring of domain knowledge)

Specifically used to structure and allow reasoning

Description Logics employ a similar structuring

- Description Logics subsets of *first order logic*, and so can be viewed as providing a semantics for many network-based object-orientated formalisms.
- They serve to disambiguate imprecise representations that these formalisms permit.

For example, what precisely is the meaning of the description?

FROG HasColour **GREEN**?

Every *frog* is just **green**?

Every frog is also **green**?

There is a frog which is just ***green***?

Reasoning

?

Difference in First order Logic and DL

First-Order Logic	Description Logic
$A(x)$	A
$C(a)$	$C(a)$, alternatively $a : C$
$A \approx B$	$A \equiv \mathrm{B}$
$\neg C(x)$	$\neg C$
$C(x) \wedge D(x)$	$C \sqcap D$
$C(x) \vee D(x)$	$C \sqcup D$
$\forall \mathrm{x}(C(x) \rightarrow D(x))$	$C \sqsubseteq D$
$R(a,b)$	$R(a,b)$, alternatively $(a,b) : R$
$\forall x \forall y(R(x,y) \rightarrow S(x,y))$	$R \sqsubseteq S$
$\exists \mathrm{y}(R(\mathrm{x,y}) \wedge C(\mathrm{y}))$	$\exists R.C$
$\forall x \forall y \forall z(R(x,y) \rightarrow R(y,z) \rightarrow R(x,z))$	$R \circ R \sqsubseteq R$

Difference in First order Logic and DL

- Description logics have more *features* than first order logics.
- Description logic is less *expressive* than first order logic.
- Despite of the feasibility of direct translation between FOL and DL, guaranteeing complete and terminating reasoning requires a different transformation, such as the *structural transformation*.
- The *structural transformation* is based on a conjunction normal form, which replaces FOL sub-formulae with new predicates, for which it also provides definitions.

Formal Definitions By Example

- "A man that is married to a doctor, and all of whose children are either doctors or professors."

Let's define the concept –

$$Human \sqcap \neg Female \sqcap (\exists married.Doctor) \sqcap (\forall hasChild.(Doctor \sqcup Professor))$$

Details:

Employs the Boolean constructors – *conjunction* as set intersection, *disjunction* as set Union, *negation* as set complement, *existential restriction constructor* ($\exists r.C$), and the *value restriction constructor* ($\forall r.C$).

Applications on the concept

$$\text{Human} \sqcap \neg\text{Female} \sqcap (\exists\text{married.Doctor}) \sqcap (\forall\text{hasChild.}(\text{Doctor} \sqcup \text{Professor})).$$

- Let's say an individual,

Bob, belongs to ∃married.

Doctor, belongs to the concept Doctor.

Doctor married to *Bob* (is related to Bob via the *married role*).

Bob belongs to *∀hasChild.*

Bob's children are either *doctors or professors* (Doctor V Professor) .

Structure of Concept Descriptions

- *Concept descriptions* used to *build* statements in a *DL knowledge base.*

- It has two parts:

terminological

and

assertional

Terminological (TBox)

- *Terminological part (Tbox)* – describe the *relevant notions* of an *application domain* by stating properties of *concepts and roles*, and *relationships* between them.
- *Simplest form*, a TBox statement introduce a ***name (abbreviation)*** for a complex description.
- **Example**: Let's introduce the name *HappyMan* as an abbreviation using previous concept –

$$\mathrm{HappyMan} \equiv \mathrm{Human} \sqcap \neg\mathrm{Female} \sqcap (\exists \mathrm{married}.\mathrm{Doctor}) \sqcap (\forall \mathrm{hasChild}.(\mathrm{Doctor} \sqcup \mathrm{Professor}))$$

- **Conclusion:** all the *knowledge* in example could easily be represented by *formulae* of *first-order predicate logic*.

Assertional (ABox)

- *Assertional part (Abox)* – describe a *concrete situation* by stating properties of *individuals*—it corresponds to the *data* in a *database setting*.

- **Example**: *Bob* belongs to the concept *HappyMan*, that *Mary* is one of his children, and that *Mary* is not a *doctor.*

$$\mathrm{HappyMan}(\mathrm{BOB}), \quad \mathrm{hasChild}(\mathrm{BOB}, \mathrm{MARY}), \quad \neg\mathrm{Doctor}(\mathrm{MARY})$$

- **Conclusion:** Modern DL systems of restricted ABox formalism, which basically used to *state ground facts.*

What We understand by this………………….

- *Modern description logic systems* provide their users with *reasoning services* that can automatically deduce *implicit knowledge* from the *explicitly represented knowledge*, and always yield a *correct answer in finite time*.

- *Inference capabilities* have both the *terminological statements (schema)* and the *assertional statements (data).*

- For building the *TBox*, making use of the reasoning services provided to ensure that all concepts in it are *satisfiable*. An ABox, one would first check for its *consistency* with the TBox, for example, compute the *most specific concept(s)* that each individual is an instance of (this is often called realizing the ABox).

Work & Research in DLs

- *Investigating* this trade-off between the *expressivity of DLs* and the *complexity* of their *inference problems.*

- This investigation both *theoretical research*, e.g., determining the worst case complexities for various DLs and reasoning problems, and *practical research*, e.g., developing systems and optimization techniques, and empirically evaluating their behavior when applied to benchmarks and used in various applications.

- The *goal* of the research was still to *design decidable extensions* i.e. extensions leave the realm of classical *first-order predicate logic*, such as DLs with modal and temporal operators, fuzzy DLs, and probabilistic DLs.

History of Description Logics – Phase 0

- **Phase 0 (1965–1980) :** pre-DL phase, *semantic networks* and *frames* were introduced as specialized approaches for *representing knowledge* in a *structured way.*

- *Failed :* lack of a formal semantics.

- *Solution:* Brachman's structured inheritance networks.

History of Description Logics – Phase 1

- **Phase 1 (1980–1990):** implementation of systems, such as KL-ONE, K-REP, KRYPTON, BACK, and LOOM. These have *Structural subsumption algorithms*, which first *normalize* the *concept descriptions*, and then *recursively* compare the *syntactic structure* of the normalized descriptions.
- *Failed :* complete only for very *inexpressive DLs*, i.e., for more expressive DLs they cannot detect all subsumption/instance relationships.
- *Solution: First logic-based accounts* of the semantics of the underlying representation formalisms.

History of Description Logics – Phase 2

- **Phase 2 (1990–1995):** introduction of a *new* algorithmic paradigm into DLs, so-called *tableau based algorithms*. It has DLs with all Boolean operators. Also, *complete* also for expressive DLs.

- *Failed :* all attempts to build a model failed with contradictions.

- *Solution: R*elated to *modal logics,* thorough analysis of the complexity of reasoning in various DLs.

History of Description Logics – Phase 3

- **Phase 3 (1995–2000) :** development of inference procedures for very expressive DLs, either based on the tableau approach or on a translation into modal logics.

- *Work Methodology:* the relationship to *modal logics* and to *decidable fragments* of first order logic was studied in more detail, and applications in databases (like schema reasoning, query optimization, and integration of databases) were investigated.

Current Phase - Phase 4

- ***Working Methodolgy:***

Results from the previous phases are being used to *develop industrial strength DL systems* employing very expressive DLs.

Applications like the *Semantic Web* or *knowledge representation* and *integration in medical and bio-informatics* in mind.

Academic side, the interest in less expressive DLs has been revived, with the goal of developing tools that can deal with very large terminological and/or assertional knowledge bases

Attributive concept Language with Complements, ALC

- *ALC:* A basic DL, first naming scheme for DLs.
- Obtained from *AL* by adding the *complement operator (¬)*.
- *ALE*, obtained from AL by adding *existential restrictions (∃r.C)*.

- **Definition:** The DL that includes set of constructors like, *conjunction, disjunction, negation, existential restriction and value restriction* is called **ALC.**

Syntax and Semantics of ALC

- ***Syntex: NC*** be a set of concept names and ***NR*** be a set of role names then, if C and D are ALC-concept descriptions and r ∈ NR, then **C v D, C Λ D, ¬C, ∀r.C**, and **∃r.C** are **ALC-concept descriptions**.
- ***Semantics:*** An interpretation consists of a nonempty set **ΔI**, called the domain of **I**, and a function that maps every ALC-concept to a subset of **ΔI.**

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}, \qquad \bot^{\mathcal{I}} = \emptyset,$$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, \qquad \neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},$$

$$(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \textit{There is some } y \in \Delta^{\mathcal{I}} \textit{ with } \langle x, y\rangle \in r^{\mathcal{I}} \textit{ and } y \in C^{\mathcal{I}}\},$$

$$(\forall r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \textit{For all } y \in \Delta^{\mathcal{I}}, \textit{if } \langle x, y\rangle \in r^{\mathcal{I}}, \textit{then } y \in C^{\mathcal{I}}\}.$$

Extensions to ALC

- DL based ontology languages: OIL, DAML + OIL, and OWL.
- ALC has been extended with several features that are important in an ontology language, including ***number restrictions, inverse roles, transitive roles, sub-roles, concrete domains***, and ***nominals***.
- Number restrictions: it is possible to describe the number of relationships of a particular type that individuals can participate in. Example: a person can be married to at most one other individual

$$\text{Person} \sqsubseteq \leqslant 1\ \text{married}$$

Extensions to ALC

- **Qualified number restrictions**: we can additionally describe the type of individuals that are counted by a given number restriction. Example: HappyMan to include the fact that instances of HappyMan have at least two children who are doctors:

$$\begin{aligned} \mathrm{HappyMan} \equiv\ & \mathrm{Human} \sqcap \neg \mathrm{Female} \sqcap (\exists \mathrm{married.Doctor}) \\ & \sqcap (\forall \mathrm{hasChild.}(\mathrm{Doctor} \sqcup \mathrm{Professor})) \\ & \sqcap \geqslant 2\ \mathrm{hasChild.Doctor} \sqcap \leqslant 4\ \mathrm{hasChild.} \end{aligned}$$

Extensions to ALC

- ***Inverse roles, transitive roles,*** and ***subroles*** we can, in addition to *hasChild*, also use its inverse *hasParent*, specify that *hasAncestor* is transitive, and specify that *hasParent* is a subrole of *has Ancestor*.

- ***Concrete domains*** integrate DLs with concrete sets such as the real numbers, integers, or strings, as well as concrete predicates defined on these sets, such as numerical comparisons, string comparisons (e.g., isPrefixOf), or comparisons with constants.

Extensions to ALC

- ***Nominal constructor***: allows us to use individual names also within concept descriptions: if *"a"* is an individual name, then ***{a}*** is a concept, called a nominal, which is interpreted by a singleton set.

Example: Using the individual **Turing**, we can describe all those computer scientists that have met Turing by **CScientist Λ ∃hasMet.{Turing}.**

Naming Convention

An additional comment on the naming of DLs is in order:

The letter **S** is often used as an abbreviation for the "basic" DL consisting of ALC extended with transitive roles.

The letter **H** represents subroles (role Hierarchies),

O represents nominals (nominals),

I represents inverse roles (Iinverse),

N represent number restrictions (N umber), and

Q represent qualified number restrictions (Qualified).

The **"generic" D** is used to express that some concrete domain/datatype has been integrated.

The DL corresponding to the OWL DL ontology language includes all of these constructors and is therefore called -

SHOIN (D).

More about ALC

- ***Relationships with ALC :*** Predicate Logic & Modal Logic
- ***Implementation & optimization techniques:*** Absorption, Dependency directed backtracking.
- ***Complexity***: PSpace and ExpTime
- ***Reasoning Techniques:*** Automata Based approach
- Automata based approach contains tree modal property, looping tree automata, the emptiness test, & the reduction.

More Naming conventions

- Frame based description language ***(FL)***
- Existential Language ***(EL)***

DLs in Language Applications

- Telecommunications equipment
- software information
- documentation systems
- area of databases, support schema design, schema & data integration and query answering
- basis for ontology languages, OIL, DAML + OIL and OWL

Semantic web ontology Language (OWL)

- *OWL*, a semantic web ontology language, developed by the *W3C Web-Ontology* working group.
- *Semantics* defined via a *translation* into an *expressive DL.*
- *Mapping* allows *OWL* to exploit results from *DL research* (e.g., regarding the decidability and complexity of key inference problems).

Semantic web ontology Language (OWL)

- *Mapping* allows to use implemented *DL reasoners* (e.g., *FaCT*) in order to provide reasoning services for OWL applications.

- *A role hierarchy*, describing: *domain* in terms of *classes* (corresponding to *concepts*) and *properties* (corresponding to *roles*).

- *ontology* consists of a set of *axioms* that assert, e.g., subsumption relationships between *classes or properties.*

Facets of OWL

- 3 *"species"* of OWL: *OWL Lite, OWL DL* and *OWL full*.
- Only the first two, have DL based semantics.
- The semantics of OWL full is given by an extension of the *RDF model theory.*
- **Resource Description Framework (RDF) :** a family of *W3C* for *meta data modelling.* Used in knowledge management applications

More About OWL

- *OWL classes* - *names* or *expressions* built up from simpler classes and properties using a variety of constructors.
- **Example:** Human ∧ Male would be written as - *The full XML serialization*

```
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Human"/>
    <owl:Class rdf:about="#Male"/>
  </owl:intersectionOf>
</owl:Class>
```

while ($\geqslant$ 2 hasChild.Thing) would be written as

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasChild"/>
  <owl:minCardinality
    rdf:datatype="&xsd;NonNegativeInteger">2
  </owl:minCardinality>
</owl:Restriction>
```

More About OWL…..

- There are a few additional constructors provided as "syntactic sugar", but all are trivially reducible to -

Constructor	DL syntax	Example
`intersectionOf`	$C_1 \sqcap \cdots \sqcap C_n$	Human $\sqcap$ Male
`unionOf`	$C_1 \sqcup \cdots \sqcup C_n$	Doctor $\sqcup$ Lawyer
`complementOf`	$\neg C$	$\neg$Male
`oneOf`	$\{x_1 \ldots x_n\}$	{john, mary}
`allValuesFrom`	$\forall P.C$	$\forall$hasChild.Doctor
`someValuesFrom`	$\exists r.C$	$\exists$hasChild.Lawyer
`hasValue`	$\exists r.\{x\}$	$\exists$citizenOf.{USA}
`minCardinality`	$(\geqslant nr)$	$(\geqslant$ 2 hasChild$)$
`maxCardinality`	$(\leqslant nr)$	$(\leqslant$ 1 hasChild$)$
`inverseOf`	r^{-}	hasChild^{-}

Features of OWL

- besides *"abstract"* classes, use of *XML Schema datatypes-* string, decimal and float & has *someValuesFrom, allValuesFrom*, and *hasValue* restrictions

- Allowing for concepts such as ∃age.xsd : *nonNegativeInteger*; e.g., be used in an axiom Person " ∃age.xsd : nonNegativeInteger to assert that all persons have an age that is a nonnegative integer.

Features of OWL

- *Axioms* allows to assert *subsumption* or *equivalence* with respect to *classes or properties*, the *disjointness* of classes, and the *equivalence* or *non-equivalence* of individuals (resources).

- OWL also allows *properties of properties (i.e., DL roles)* to be asserted. Possible to assert a property i.e. transitive, functional, inverse functional or symmetric.

OWL Axioms

Axiom	DL syntax	Example
`subClassOf`	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
`equivalentClass`	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
`subPropertyOf`	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
`equivalentProperty`	$P_1 \equiv P_2$	cost $\equiv$ price
`disjointWith`	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq$ $\neg$Female
`sameAs`	$\{x_1\} \equiv \{x_2\}$	{Pres_Bush} $\equiv$ {G_W_Bush}
`differentFrom`	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq$ $\neg${peter}
`TransitiveProperty`	P transitive role	hasAncestor is a transitive role
`FunctionalProperty`	$\top \sqsubseteq (\leqslant 1\ P)$	$\top \sqsubseteq (\leqslant 1$ hasMother$)$
`InverseFunctionalProperty`	$\top \sqsubseteq (\leqslant 1\ P^-)$	$\top \sqsubseteq (\leqslant 1$ isMotherOf$^-)$
`SymmetricProperty`	$P \equiv P^-$	isSiblingOf $\equiv$ isSiblingOf$^-$

OWL tools and applications

- Ontology design tools, both "*academic*" and *"commercial"*.
- *Objective:* highlighting inconsistent classes and implicit subsumption relationships.
- **Example** - Protégé, Swoop, OilEd and TopBraid Composer.
- Reasoning support for such tools provided by a DL reasoner, ex - *FaCT++, RACER* or *Pellet*.

OWL tools and applications

- Ontology development in fields as diverse as -

  biology, medicine,

  geography, geology,

  astronomy, agriculture and

  defence.

- **Example:** Biological Pathways Exchange (BioPAX) ontology, the GALEN ontology, the Foundational Model of Anatomy (FMA), and the National Cancer Institute thesaurus,Medical Entities Dictionary (MED).

Other researches in DLs.

- Great deal of interest in the idea of *combining DLs* with other KR formalisms such as *rules* and *Answer Set Programming (ASP).*

- *Via* with the ability to describe more complex relationships between named individuals, or by adding support for non-monotonic features such as negation as failure.

Other researches in DLs.

- ***Important contributions:*** rule support in the Classic system, the integration of Datalog with DLs in AL-log and CARIN, the integration of answer set programming with DLs, and the extension of DLs with so-called DL-safe rules.

- Investigating the implementation and optimization of DL systems.

- A number of tools available that use the reasoners to support, e.g., ontology design or schema integration.

Let’s see a demo on software named…..

Protégé

https://protege.stanford.edu/

References

Handbook of knowledge representation

https://dai.fmph.uniba.sk/~sefranek/kri/handbook/handbook_of_kr.pdf

Thank You!

Questions ?