Description Logics

Dave Dubin

March, 2019

# Description Logics

* Description logics (plural) are a family of artificial languages used in information modeling and knowledge representation.
* They are the logical foundations of ontology languages (such as OWL) that are used in deductive information management applications.
* An ontology can be understood as a set of declarations and constraints in a specific description logic such as or .
* Any description logic can be understood as a *decidable subset* of first order logic.

# Logic, computation, and decidability

* Decidability is a complex topic, but it’s basically about our ability to use general-purpose algorithms to answer questions.
* In deductive applications we care about more than just retrieving answers we’ve explicitly stored. Typically we store logical rules or definitions that act on individual facts and records. A typical query would be asking for a list of individuals that are instances of a specific class.
* The more things you can express in an artificial language, the more computational resources (time and memory) it takes to answer questions. Many interesting questions have no general way to answer them.
* Small expressive differences between languages make a huge difference in what kinds of questions are either too expensive or impossible to answer.

# Decidability, continued

* In logic, a typical question is whether some statement follows logically from a set of other statements, or whether its truth is at least consistent with those statements (i.e., that it could be true).
* Propositional logic is decidable, but general-purpose procedures for deduction are too expensive.
* Predicate logic is only semi-decidable. If a statement follows logically from a set of premises, we have algorithms that will verify that. But that can take a very long time, and if the statement does *not* follow, then our procedures may run forever without ever telling us that the answer is no.
* You can think of description logics as crippled versions of predicate logic. The expressive power has been restricted in order to guarantee that our questions are decidable, and that algorithms for answering our questions are usually efficient.

# Key differences: predicate logic vs. description logics

1. Description logics only have unary and binary predicates.
   * Just like Bach’s fragment. Nothing like is allowed.
   * This is one reason why description logics are decidable.
   * We’ll also see a convenient adaptation to the RDF model.
   * Unary predicates are called *concepts* (like RDF classes).
   * Binary predicates are called *roles* (like RDF properties).
2. Variables aren’t explicit in syntax. Concepts have one and roles have two.
3. Reasoning about anonymous individuals is limited with description logics.
4. Description logic statements are understood to be divided into three categories:
   * Statements assigning individuals to concepts and roles are in the *ABox*.
   * Relationships among concepts are in the *TBox*.
   * Relationships among roles are in the *RBox*.

# Syntax used in description logics

Correspondences between description logics and first order logic.

|  |  |
| --- | --- |
| Description Logic | Predicate Logic |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# More syntax examples

Not all description logics let you use all these features.

|  |  |  |
| --- | --- | --- |
| Description Logic | Predicate Logic | English |
|  |  | Fathers or mothers |
|  |  | Unmarried females |
|  |  | Parents of only female children |
|  |  | Everything is either male or female |
|  |  | Nothing is both male and female |
|  |  | Parents are parents of something |
|  |  | Sons of anything are male |
|  |  | You can only be son of a parent |

# Existential quantification

* The existential quantifier in DLs has the same meaning as in predicate logic, but our ability to reason about anonymous individuals is limited.
* Suppose we define a native as someone who lives in the same country they were born in.
* In predicate logic we could express this definition as:
* There’s no way to capture this definition in a description logic. We can come close with classes connected via named individuals.
* For example:

# Grammar for DLs

The grammar for a DL will provide most (but not all) of its expressive constraints. , for example, has this grammar for role and concept expressions and axioms, where is the universal role that links all pairs of individuals, is the set of integers, and are sets of individual, concept, and role names, respectively:

# DL Semantics

A DL interpretation where:

* is the domain set (Bach’s ).
* is the denotation function (Bach’s ). This function maps:
  + a concept
  + a role
  + a name
* An interpretation *satisfies* a statement if it is true in that interpretation: .
* An interpretation satisfying every statement in a set , is a *model* of .
* If **at least one** model exists for a set of DL statements then those statements are *consistent.*

# Models and open world reasoning

* We’re used to concluding that a proposition is false if we cannot demonstrate that it is true.
* That’s because our systems are understood to contain complete information, and so any question for which we cannot demonstrate a yes answer must be answered no.
* But our decentralized, networked, and cooperative information systems increasingly rely on an *open world assumption.*
* The open world assumption is that not all relevant information is necessarily available. We ask whether any model of our statements exists (i.e., whether they’re consistent).

# Open world reasoning example

* Suppose we assert that the individual Dave (class Person) stands in the relationship to the individual instructor:
* Suppose that we also assert that the domain of is the concept :
* In a conventional database application we’d expect an error unless Dave matches the domain constraint.
* But unless persons and appointments are specifically modeled as disjoint, there exist interpretations in which Dave belongs to both classes.
* In knowledge-based applications, domain rules offer sanity checks on our data.