COSC410 Logic for AI

Description Logics and the Semantic Web

Richard A. O'Keefe

2 June 2016

Key Topics

- Hiding the arguments
- The calculus of binary relations
- Individuals, Concepts, Rôles
- The Terminology Box and the Assertion Box
- What a DL is good for
- AL
- RDF
- SPARQL

Hiding the arguments

- Computer programs are often littered with variables whose main purpose is to get information from one place to another.
- A craft technique in advanced programming languages is to develop macros (Lisp), source transformers (Prolog), or combinators (Haskell, Clean) to "hide the plumbing".
- Example: $p \longrightarrow q, r, s$ is Prolog for $\forall S_0 \forall S_1 \forall S_2 \forall S (p(S_0, S) \leftarrow q(S_0, S_1) \wedge r(S_1, S_2) \wedge s(S_2, S))$.

Hiding and unary predicate calculus

- Instead of writing $P(x)$ write P.
- All the predicate symbols in a formula will have the same variable as argument.
- A sentence is $(\forall x)\phi$ so we don't need to write x.
- We can treat a UPC formula *as if* it were propositional.

Predicates and sets, 1

- What is the difference between a predicate and a set?

Predicates and sets, 1

- What is the difference between a predicate and a set?
- One's weaselly told, the other's stoatally different.

Predicates and sets, 1

- What is the difference between a predicate and a set?
- One's weaselly told, the other's stoatally different.
- That is, not much.

Predicates and sets, 2

Logic	Set	Assertion
$P(x)$		$x \in P$
$\bot$	$\varnothing$	
$\top$	$\mathcal{U}$	
P	P	
$P \wedge Q$	$P \cap Q$	
$P \vee Q$	$P \cup Q$	
$\neg P$	$\mathcal{U} \setminus P$	$P \subseteq \varnothing$
$P \to Q$	$Q \cup (\mathcal{U} \setminus P)$	$P \subseteq Q$
$P \leftrightarrow Q$	$(P \setminus (Q \setminus P)) \cup (Q \setminus (P \setminus Q))$	$P = Q$

Binary Relations and Regular Expressions

- A binary relation xRy can be thought of as a predicate of two arguments $R(x, y)$ or as its graph, the set $\{(x, y)|xRy\}$.
- A language $\mathcal{L}$ can be thought of as a set $\mathcal{L}_1$ of strings, or as a binary relation $\alpha\mathcal{L}_2\omega \leftrightarrow \exists\mu(\alpha = \mu \frown \omega \wedge \mu \in \mathcal{L}_1)$.
- There's a rich set of operations on binary relations.

Binary Relations

- $\mathcal{I} = \{(x, x) | x \in \mathcal{U}\}$
- $P^{=} = \{(x, x) | P(x)\}$ when P is a unary predicate
- $P^{r} = \{(y, x) | (x, y) \in P\}$ — converse/opposite
- $P.Q = \{(x, y) | \exists z((x, z) \in P \wedge (z, x) \in Q)\}$
- $P^{?} = P \cup \mathcal{I}$
- $P^{*} = \mathcal{I} \cup P^{+}$ reflexive transitive closure
- $P^{+} = P.P^{*}$ transitive closure

Regular expressions

- Let $c \in \Sigma$ be a character in a character set.
- Define $\hat{c} = \{(c\omega, \omega) | \omega \in \Sigma^*\}$
- That is, $s\hat{c}r$ iff s begins with c and the rest of s is r.
- The regular expression /(b|br)ea*ds?/ is the binary relation $(\hat{b} \cup \hat{b}.\hat{r}).\hat{e}.\hat{a}^*.\hat{d}.\hat{s}^?$
- The algebra of binary relations generalises regular expressions.

Individuals

- An *individual* is (a name for) a single thing.
- OWL also calls them individuals.
- Logic calls them constant(symbol)s.
- Description logics do **not** make the Unique Name Assumption. john_key and andrew_little could well name the same thing, as far as a DL is concerned, unless there is evidence against it.

Concepts

- A *concept* is (a name for) a property of things.
- OWL calls them classes.
- Logic calls them unary predicate(symbol)s.
- Description logics do **not** make the Closed World Assumption. If we cannot show that andrew_little $\in$ prime_ministers, a DL won't conclude that he isn't, unless there is evidence against it.

Rôles

- A *rôle* is (a name for) a relation between pairs of things (like father-of) or a relation between things and values (like has-name).
- OWL calls them properties.
- Logic calls them binary predicate(symbol)s.
- Description logics do **not** make the Closed World Assumption. If we cannot show that (richard,40)$\in$ age, a DL won't conclude that he isn't. Maybe someone can have two ages?

Notations and semantics

- DLs use different notation from standard logic.
- Web notations are different again.
- The *semantics* isn't really different.
- We are using *restrictions* of logic in order to get efficient reasoning.
- We are giving up expressiveness to get decidability.

TBox and ABox

- The *terminology box* (TBox) holds *general* knowledge about concept hierarchies.
- The *assertion box* (ABox) holds facts about individuals.
- The split has to do with what kinds of reasoning are done and what the algorithms are, and also with when information becomes available.
- (has-appendicitis $\rightarrow$ has-abdominal-condition) is a general rule you might know ahead of time and "compile"
- (anthony : has-appendicitis) is a specific fact you might learn at run time and wish to derive consequences of.

Original setting

- There is a general reasoning program.
- It delegates some tasks to a specialised module which always terminates, hopefully fast.
- Some knowledge is used over and over again, so we'd like to pre-check and "compile" it.
- Some facts are added during a run, possibly by the general program itself.
- I'm reminded of "Satisfiability Modulo Theories" solvers, same delegation to specialised modules idea.

What can a DL do for us?

- **Concept Subsumption**
  Is $C \sqsubseteq D$ definitely true, definitely false, unknown?
- **Satisfiability**
  Is $C \sqsubseteq \bot$ true, false, or unknown?
- **Concept Equivalence**
  $C \equiv D$ iff $C \sqsubseteq D \wedge D \sqsubseteq C$
- **Disjointness**
  C and D are disjoint iff $C \sqcap D \sqsubseteq \bot$

What can a DL do for us (2)?

- **Concept classification**
  Form an explicit hierarchy from all mentioned concepts.
- **Consistency checking**
  Is $x \in C \wedge x \notin C$ possible?
  Is $(x, y) \in r \wedge (x, y) \notin r$ possible?
  Could a given TBox ever have a non-empty ABox?

What can a DL do for us (3)?

- **Classify individuals**
  Is $x \in C$ definitely true, definitely false, unknown?
  Take all assertions $x \in A_i$ and test whether $A_1 \sqcap \cdots \sqcap An \sqsubseteq C$.
- **Find individuals**
  Given C, find all individuals x for which $x \in C$.
- **Description (realisation)**
  Given an individual and a set of concepts, find the most specific concept the individual belongs to.

AL: a description logic

Concepts can be

- $\top$
- $\bot$
- A — atomic concepts (names)
- $\neg A$ — negated atomic concepts
- $C \sqcap D$ — intersection
- $\exists r.\top$ – limited existential
- $\forall r.C$ — value restriction

Not in AL

- No union ($\sqcup$)
- Negation only applies to atomic concepts
- There is no way to construct rôles

Extensions of AL

Systematic naming: AL[U][E][N][C].

- U : union $C \sqcup D$ is allowed
- E : full existential $\exists r.C$ is allowed
- N : number restrictions $\geq nR$ and $\leq nR$ are allowed
- C : general complements $\neg C$ are allowed

An example

TBox:

$$\mathsf{Person} \sqsubseteq \forall\, \mathsf{tends}.\mathsf{Pet}$$
$$\exists\, \mathsf{tends}.\top \sqsubseteq \mathsf{Person}$$
$$\mathsf{Person} \sqcap \mathsf{Pet} \sqsubseteq \bot$$
$$\mathsf{Dog} \sqsubseteq \mathsf{Pet}$$
$$\mathsf{Bird} \sqsubseteq \mathsf{Pet}$$
$$\mathsf{Dog} \sqcap \mathsf{Bird} \sqsubseteq \bot$$
$$\mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot$$

ABox:

$$\mathsf{richard} \in \mathsf{Person} \sqcap \mathsf{Male}$$
$$\mathsf{lily} \in \mathsf{Dog} \sqcap \mathsf{Female}$$
$$\mathsf{jazzie} \in \mathsf{Bird} \sqcap \mathsf{Female}$$
$$\mathsf{perry} \in \mathsf{Bird} \sqcap \mathsf{Male}$$
$$(\mathsf{richard},\mathsf{lily}) \in \mathsf{tends}$$
$$(\mathsf{richard},\mathsf{jazzie}) \in \mathsf{tends}$$
$$(\mathsf{richard},\mathsf{perry}) \in \mathsf{tends}$$

Resource Description Framework

- An RDF document is basically a set of triples.
- An (Attribute $\circ$ Object $\equiv$ Value) store was built into an old AI language called SAIL.
- A rôle fact in a DL, $(x, y) \in r$, is written $x\ r\ y$.
- Think of x and y as nodes in a directed graph with an edge labelled r linking them. (Graph data bases. . .)

Naming in RDF

- subjects, predicates, and objects can be URIs.
- subjects and objects can be "blank nodes".
- objects can be literals (numbers or strings).

URIs

- Mostly, they're namespaced strings.
- Some of them have semantics defined in public documents, notably rdf and foaf.
- They always stand for the same thing (are *rigid designators*).

Blank nodes

Blank nodes are like existentially quantified variables.
_:foobar will refer to the same node throughout an RDF graph, but it doesn't have an absolute identity that can be referred to elsewhere.
_:m mother_of simpsons:bart.
_:m hair_colour "blue".

RDF Schema is a DL

- C rdf:type rdf:class. C is a concept.
- r rdf:type rdf:property. r is a rôle.
- x rdf:type C. $x \in C$.
- C rdfs:subClassOf D. $C \sqsubseteq D$
- p rdfs:subPropertyOf q. $p \sqsubseteq q$
- p rdfs:domain C. $\exists r.\top \sqsubseteq C$
- p rdfs:range C $\exists \top.r \sqsubseteq C$

Example

```
@base <http://example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>
@prefix schema: <http://schema.org/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix wd: <http://www.wikidata.org/entity/> .
@prefix db: <http://dbpedia.org/resource/> .
wd:Q12418
    dcterms:title ”Mona Lisa” ;
    dcterms:creator db:Leonardo_da_Vinci .
```

Example, continued

```
<bob#me>
    a foaf:Person ;
    foaf:knows <alice#me> ;
    schema:birthDate "1990-07-04"^^xsd:date
    foaf:topic_interest wd:Q12418 .

[] foaf:topic_interest [
    dcterms:title "RDF" ;
    dcterms:creator <http://www.w3c.org> ] .
```

Triple Stores

- A triple store is a specialised data base.
- It accepts (s,p,o) and (s,p,v) triples.
- You can enumerate partial matches.
- RDFS is a description logic.
- We'd like queries to succeed if they are *true*, not just if they were *stored*.
- Some triple stores do this, *e.g.*, ClioPatria.

SPARQL

- A rich SQLish query language for RDF.
- SELECT [DISTINCT] *vars* WHERE { *triples* }
- CONSTRUCT { *triples* } WHERE { *triples* }
- In a WHERE part, rôles can be r, ^r (inverse), r1/r2 (composition), r1|r2 (or), r∗, r+, r?, (r), . . .
- :richard (:father|:mother)/:brother ?unc asks for my uncles.
- **This is richer than RDF can express.**

SPARQL, continued

- SELECT may use aggregate expressions: COUNT, SUM, MIN, MAX, SAMPLE
- Groups are defined using GROUP BY *vars*
- and filtered using HAVING (*expression*).
- You can sort with ORDER BY *vars*.
- **It's not a logic, it's a query language**. We never ask about subsumption between queries, we never infer queries from queries, *etc*.