+

Semantic Web (ECS735P/D) 4 – Theorem Proving and Tableaux

Julian Hough
EECS, Queen Mary University of London

Outline

- Recap of Description Logic
  - English translation > DL
  - Predicate Logic > DL
- Theorem Proving
  - Inference in Knowledge Bases
  - General Tableaux framework (prop logic)
  - DL Tableaux algorithm

Outline

- Recap of Description Logic
  - English translation > DL
  - Predicate Logic > DL
- Theorem Proving
  - Inference in Knowledge Bases
  - General Tableaux framework (prop logic)
  - DL Tableaux algorithm

RECAP: English into DL

- Definitions and class hierarchies:

  (definitions using 1-place predicates (atomic classes) with intersection):

  animal ≡ alive ⊓ ¬plant

  (*animals are things that are alive and are not plants, i.e. the animal class is the subclass of the alive class that is not intersected by the plant class)*

  woman ≡ human ⊓ female

  (women are things that are both human and female, i.e. the woman class is the intersection of the human class and the female class)

RECAP: English into DL

- Definitions and class hierarchies:

  (2-place predicates/relations/quantification):

  carnivore ≡ ∀eats.animal

  (carnivores are things that eat only animals; i.e. the carnivore class is equivalent to the class defined by the "eats" relation value restricted for things within the animal class)

  herbivore ≡ ∀eats.plant

  (herbivores eat only plants; i.e. the herbivore class is equivalent to the class defined by the "eats" relation value restricted for things within the plant class)

RECAP: English into DL

- Definitions and class hierarchies:

  (2-place predicates/relations/quantification):

  ∃eats.⊤ ≡ ⊤

  (everything eats something, i.e. the class defined by the "eats" relation which takes has something within the whole world as its range (unrestricted) is equivalent to the whole world)

RECAP: English into DL

- Definitions and class hierarchies:

  (subsumption and subclasses):

  kebabLover $\sqsubseteq$ $\exists$*eats.meat*

  (kebab lovers are one of the things that eat a form of meat as part of their diet i.e. the kebab lover class is a subclass of the class defined by the eats relation that takes some thing(s) in the meat class as its range; there might be other meat-eaters in the world, not just kebabLovers, so kebabLover is a subclass)

RECAP: English into DL

- Numbers/cardinality
  - Subsumption- someone can only be married to one person:

$$\textbf{person} \sqsubseteq \leq \textbf{1married}$$

(A person is a subclass of the class of things in domain of things picked out by the "married" relation whose range is number restricted to being at most one)

- Definition- a happy man is one who has between 2 and 4 children:

$$\textbf{happyMan} \equiv \textbf{male} \sqcap \geq \textbf{2hasChild} \sqcap \leq \textbf{4hasChild}$$

- Statement- at least three students visited every club

$$\textbf{club} \sqsubseteq \ \geq \textbf{3visitedBy.student}$$

RECAP: DL > Predicate Calculus

- $\text{animal} \equiv \text{alive} \sqcap \neg\text{plant}$
  - $\forall x\ (\text{animal}(x) \leftrightarrow (\text{alive}(x) \wedge \neg\text{plant}(x)))$
- $\text{carnivore} \equiv \forall\text{eats.animal}$
  - $\forall x\ \forall y(\text{carnivore}(x) \leftrightarrow (\text{animal}(y) \wedge \text{eats}(x,y)))$
- $\exists\text{eats}.\top \equiv \top$
  - $\forall x\ \exists y\ (\text{eats}(x, y))$
- $\text{kebabLover} \sqsubseteq \exists \mathit{eats.meat}$
  - $\forall x\ \exists y\ (\text{kebabLover}(x) \rightarrow \text{meat}(y) \wedge \text{eats}(x,y))$

Outline

- Recap of Description Logic
  - English translation > DL
  - Predicate Logic > DL
- Theorem Proving
  - Inference in Knowledge Bases
  - General Tableaux framework (prop logic)
  - DL Tableaux algorithm

Semantic Web Architecture

Knowledge Base (KB)

Tbox (schema)

$animal \equiv alive \sqcap \neg plant$

$carnivore \equiv \forall eats.animal$

$herbivore \equiv \forall eats.plant$

$\exists eats.\top \equiv \top$

Abox (data)

carnivore(John)

herbivore(Jane)

Inference system

Semantic Web Architecture

Knowledge Base (KB)

<u>Tbox (schema)</u>

animal ≡ alive ⊓ ¬plant

carnivore ≡ ∀eats.animal

herbivore ≡ ∀eats.plant

∃eats.T ≡ T

<u>Abox (data)</u>

carnivore(John)

herbivore(Jane)

Inference system

Does some statement:

John ⊑ ∀eats.plant

(‘John only eats plants’)

hold in our KB? i.e.

Is it TRUE according to our KB? (Theorem Proving)

Making inferences: Querying the KB

- For knowledge bases (or ontologies) it is important to be able to make inferences:
- Does a statement P follow from the knowledge base KB:
  - Is P entailed by KB, i.e. $\mathbf{KB} \vdash \mathbf{P}$?
  - Is the knowledge base KB consistent ?
- Can be done by *model checking*- filling out truth tables (propositional logic) but it is computationally expensive

Propositional Logic

Arguments: assert truth of the premises (like the KB), see if the conclusion (query) holds.

1. Convert to statement in prop logic
2. Enumerate models of all *atomic* statements (rows in truth table)
3. Fill in the truth table

(Proving the validity through enumerating models)

Propositional Logic

"If Tim Berners Lee works, there is progress.

There is no progress.

Therefore, Tim Berners Lee doesn't work"

1. Convert to statement

$$\begin{array}{l} P \rightarrow Q \\ \neg Q \\ \hline \neg P \end{array} \dashrightarrow \begin{array}{l} = ((P \rightarrow Q) \land \neg Q) \quad \vdash \neg P \\ = ((P \rightarrow Q) \land \neg Q) \rightarrow \neg P \end{array}$$

Propositional Logic

2. Enumerate all models.

		premise 1	premise 2	conclusion	formula
P	Q	$P \rightarrow Q$	$\neg Q$	$\neg P$	$((P \rightarrow Q) \wedge \neg Q) \rightarrow \neg P$
T	T	T	F	F	T
T	F	F	T	F	T
F	T	T	F	T	T
F	F	**T**	**T**	**T**	T

3. Check. Tautology. VALID ARGUMENT ☺

Propositional Logic

- If premises = KB and query = conclusion, we have a simple exhaustive model checking entailment and consistency in the Semantic Web. Decompose to variables and check models.

- But what about *complexity*?

- If n = number of variables, we have 2^n possible models. $2^5 = 32$; $2^6 = 64$ etc.. !

- In Semantic Web applications, we might have a large world to deal with.

- We need to use an inferential calculus that is as reliable, but much faster..

Theorem Proving

- We can construct a proof to verify entailment/lack of entailment from the sentences directly- *theorem proving*
- If number of models is large, but proof short, we have saved time!
- We use *validity* in the same way as before (true in all models), but through syntactic means. We need to check for known *tautologies.*
- *Satisfiability* (true in some model) is the same concept but through syntactic means.

Theorem Proving

- Validity and satisfiability connected in that p is valid iff ¬p is unsatisfiable. There is no model in which the negation is satisfiable- no *counter model.*
- *Proof by contradiction* at the heart of theorem proving:

$$p \vdash q \textit{ iff } p \wedge \neg q \textit{ is unsatisfiable}$$

- In semantic web terms:

$$KB \vdash query \textit{ iff } KB \wedge \neg\, query \textit{ is unsatisfiable}$$

Queen Mary University of London

Theorem Proving

- Inference rules (lecture 2).
- We exhaustively apply syntactic rules until formulae are atomic (not decomposable) or until we derive a contradiction.
- Keep track of our inferences in a tree-like structure called a *tableau* (or proof tree)
- We generally stack up statements, decompose them via the *connectives* of the logic, or other inference rules.
- Try to derive a contradiction of the negation

Tableaux

- We will use Smullyan's tableaux (like Beth tableaux, but simplified)
- Trying to close branches (derive contradiction) by having P and ¬ P on the same branch

Tableaux

E.G. Try to prove $((P \rightarrow Q) \wedge \neg Q) \rightarrow \neg P$ *by proving its negation*

1. $\neg (((P \rightarrow Q) \wedge \neg Q) \rightarrow \neg P)$ (assumption)
2. $((P \rightarrow Q) \wedge \neg Q)$ (cond. sub, comp., AND elim, 1)
3. $\neg \neg P$ (cond. sub, comp., AND elim,1)
4. $P \rightarrow Q$ (AND elim, 2)
5. $\neg Q$ (AND elim, 2)

6. $\neg P$ (cond. sub, OR elim, 4) 7. Q (cond. sub, OR elim, 4)

X X

Description Logic Tableaux

Rules: Negation normal form conversion:

- $\neg(C \sqcap D)$ gives $\neg C \sqcup \neg D$

and $\neg(C \sqcup D)$ gives $\neg C \sqcap \neg D$

- $\neg\exists R.C$ gives $\forall R.\neg C$ and $\neg\forall R.C$ gives $\exists R.\neg C$

And Expansion rules:

- α expansion($\sqcap$-rule)
- β expansion ($\sqcup$-rule)
- $\forall$ expansion (Universal expansion)
- $\exists$ expansion (Existential expansion)

And Branch closing condition! X

Description Logic Tableaux

- α expansion **(⊓-rule)**

  If ($C1 \sqcap C2$) ∈ *Branch* and {$C1$,$C2$} $\not\subseteq$ *Branch* then:

  add $C1$ and $C2$ to L(x)

$$
\begin{array}{c}
C1 \sqcap C2 \\
| \\
\mathrm{C1} \\
| \\
\mathrm{C2}
\end{array}
$$

Description Logic Tableaux

-β expansion **(⊔-rule)**

If $(C1 \sqcup C2) \in BranchX$

and $\{C1, C2\} \cap BranchX = \emptyset$ then:

Add $C1$ to new $BranchY$. If this leads to a clash, go back and add $C2$ to new $BranchZ$.

$$
\begin{tikzcd}
 & {} \arrow[d, no head] & \\
 & C1 \sqcup C2 \arrow[dl, no head] \arrow[dr, no head] & \\
\mathrm{C1} & & \mathrm{C2}
\end{tikzcd}
$$

Description Logic Tableaux

- ($\exists$ expansion)

- If $\exists R.C \in BranchX$ and there is no $BranchY$ s.t. Edge($BranchX$,$BranchY$)=R and $C \in BranchY$ then:

  create new branch $BranchY$ and new edge($BranchX$,$BranchY$) = R

$$\begin{array}{c} \exists R.C \\ \Big| \, \text{R} \\ \text{C} \end{array}$$

Description Logic Tableaux

- (∀ expansion)

- If $\forall R.C \in BranchX$ and there is some $BranchY$ s.t. edge($BranchX$,$BranchY$)=R and $C \notin BranchY$ then add C to $BranchY$.

$\forall R.C$

R

C (add this)

DL Tableau Algorithm for proof

Termination of the algorithm

- The ⊓-, ⊔- and ∃-rules can only be applied once to a concept in L(x).
- The ∀-rule can be applied many times to a given $\forall R.C$ expression in L(x), but only once to a given edge (x,y).
- Applying any rule to a concept C extends the labelling with a concept strictly smaller than C.