Artificial Intelligence

Lecture 7 - Predicate Calculus

Outline

- Limitations of propositional calculus
- Predicate calculus
- Syntax
- Semantics
- Entailment
- Proof
- Resolution

Limitations of Propositional Calculus

- Propositional calculus allows us to talk about propositions (simple facts about the world) and combinations of facts
- We can't talk about things in terms of their properties or relationships to other things except in concrete terms
- We can't express rules or generalisations:
  - if the train is late and there are no taxis at the station, John is late for the meeting
  - if the train is late and there are no taxis at the station, *anyone* travelling by train will be late for the meeting

(First order) Predicate Calculus

- Predicate calculus provides a richer representation language
- Objects and properties
- Relationships between objects (including functions)
- Quantification - ability to refer to *all objects*, or to say that there is *some object* which has a given property

Syntax

- **Logical symbols** (have a fixed meaning)
  - punctuation: “(”, “)”, “.”
  - connectives: ¬ (not), ∧ (and), ∨ (or), → (conditional), ∀ (universal quantifier), ∃ (existential quantifier), = (equals)
  - variables: x, y, z, x_1, y_1, z_1, . . .

Syntax

- **Non-logical** symbols
  - function symbols: *father*, *mother*, *sum*, … (sometimes just *f*, *g*, *h*, …)
  - predicate symbols: *Man*, *Woman*, *Brother*, *Married*, … (sometimes just *P*, *Q*, *R* …)
- Function and predicate symbols have fixed *arity* (i.e., number of argument places, a non-negative integer), e.g., *father* has arity 1, *sum* has arity 2, etc.
- *Constants* (or names) are function symbols of arity 0, e.g., *John*, *EiffelTower* and *TipperaryInstitute* are function symbols of arity 0

Atomic Formulas

- **Terms**
  - every variable is a term
  - if f is a function symbol of arity n and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term
- **Formulas**
  - if P is a predicate symbol of arity n and $t_1, \ldots, t_n$ are terms, then $P(t_1, \ldots, t_n)$ is an atomic formula
  - if t_1 and t_2 are terms then $t_1 = t_2$ is an atomic formula

Complex Formulas

- If α and β are formulas and x is a variable then

  $\neg\alpha$: not α

  $\alpha \wedge \beta$: α and β

  $\alpha \vee \beta$: α or β

  $\alpha \rightarrow \beta$: if α then β

  $\forall x.\alpha$: for all x, α is true

  $\exists x.\alpha$: there exists some x such that α is true

  are formulas

Examples

- Basic facts about a domain are usually represented by atomic sentences or negations of atomic sentences, e.g., *Man*(*john*), *Woman*(*jane*)
- We can also express relations between people and objects in the domain, e.g., *Married*(*john*, *jane*), *Brother*(*john*, *william*), *Owns* (*william*, *ford*) etc
- We can also express general properties of all objects, e.g., $\forall x.$*Happy*(x) says that "*everyone is happy*" (or more precisely that every *thing* is happy)
- Or of unspecified objects, .e.g., $\exists x.\neg$*Happy*(x) says that someone (some thing) is not happy

More Examples

- More complex facts about a domain can be expressed using simple formulas composed with logical connectives
- Disjunctions, e.g., $Loves(jane, john) \vee Loves(jane, william)$ says that “Jane loves John or Jane loves William (or both)”
- Disjointness of predicates, e.g., $\forall x.\neg(Man(x) \wedge Woman(x))$ says the categories *Man* and *Woman* are disjoint
- Subtypes, e.g., $\forall x.(Dog(x) \rightarrow Animal(x))$ says that “all dogs are animals”
- Exhaustiveness, e.g., $\forall x.(Adult(x) \rightarrow Man(x) \vee Woman(x))$ says that “an adult is either a man or a woman (or both unless we specify disjointness)”
- Inverses, e.g., $\forall x.(ChildOf(x, y) \rightarrow ParentOf(y, x))$ says that “if x is a child of y, then y is a parent of x”

Exercise

- Given the predicates *Student*, *Undergraduate*, *Masters*, *Happy*, (all of arity 1) the function *mother* (with arity 1) and the constant *john*
- Express the following in predicate calculus
  - John is a student
  - if John is a student then John is happy
  - if John is a student then John's mother is happy
  - some student is happy
  - all students are happy
  - all undergraduates are students
  - all students are either undergraduates or masters but not both

Models

- A model $M = \langle D, I \rangle$ where D is the *domain*, a non-empty set of individuals
- I is an *interpretation* which
  - associates constants (0 arity function symbols) with individuals, e.g., the constants "*william*", "*bill*" and "*PresidentClinton*" may all map to the same individual in D
  - associates a function symbol (arity > 0) with a total function of the same arity from $D^n \rightarrow D$, e.g., "*sum*" with arity 2 may map to "+"
  - associates predicates of arity 1 with sets of individuals with the corresponding property, e.g., the interpretation of the predicate "*Red*" is the set of things in D which are coloured red
  - associates predicates of arity > 1 with sets of tuples (pairs, triples etc.) which specify which tuples of individuals are in the corresponding relation, e.g., the interpretation of the relation "*Married*" is the set of pairs of individuals in D who are married

Truth

- We can now specify which formulas are true in a model *M* - only defined for *sentences*, i.e., formulas without free variables
- If *α* is a variable free atomic formula of predicate calculus *α* is *true* if the appropriate relationships hold in the model
- e.g,. *Dog*(*bestFriend*(*john*)) is true in *M* if the function corresponding to *bestFriend* when applied to the individual corresponding to the constant *john* returns an individual which is in the set of dogs
- The truth of complex formulas is defined using the truth functions for the logical connectives, e.g, *α* ∧ *β* is *true* iff *α* is *true* and *β* is *true*
- ∀*x*.*α* is *true* if *α* is true for every element of *D*, and ∃*x*.*α* is *true* if *α* is *true* for at least one element of *D*

Entailment

- We can use our definition of truth in a model to define entailment for predicate calculus
- As with propositional calculus, a set of sentences α_1, α_2, . . . , α_n *entails* a sentence β, α_1, α_2, . . . , α_n |= β, if in all models where α_1, α_2, . . . , α_n are *true*, β is also *true*
- However predicate calculus models are much more complex objects than the truth assignments that make up each row in a truth table
- Only feasible way of establishing entailment is to use rules of inference to prove that β follows from α_1, α_2, . . . , α_n syntactically

Resolution

- We will consider a simplified version of resolution for predicate calculus sentences without existentially quantified variables
- All clauses are assumed to be of the form $\forall x_1, \ldots, x_n.(l_1 \vee \ldots \vee l_m)$ where $l_1 \ldots l_m$ are literals and $x_1, \ldots, x_n$ are the free variables in $l_1, \ldots, l_m$
- Reduce to CNF as for propositional calculus
- Apply the resolution rule as for propositional calculus if the clauses can be *unified*

Substitution

- A σ substitution is a finite set of pairs $\{x_1 = t_1, \ldots, x_n = t_n\}$ where the x_i are distinct variables and t_i are arbitrary terms
- If l_i is a literal, then $l_i[\sigma]$ is a literal which results from substituting each x_i in l_i by t_i
- σ unifies two literals l_i and m_j if $l_i[\sigma] = m_j[\sigma]$ e.g., $P(x, f(x))$ and $P(y, f(a))$ are unified by $\sigma = \{x = a, y = a\}$
- If c is a clause, then $c[\sigma]$ is the result of applying the substitution σ to all literals in c

General Resolution

$$\frac{(l_1 \vee \ldots \vee l_i \vee \ldots \vee l_k),\ (m_1 \vee \ldots \vee m_j \vee \ldots \vee m_n)}{(l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)[\sigma]}$$

- where l_i and m_j are *complementary literals*, i.e., one is the negation of the other
- and σ unifies l_i and m_j, i.e., $l_i[\sigma] = m_j[\sigma]$

General Resolution Example

- From the clauses $Man(socrates)$ and $\forall x.(Man(x) \rightarrow Mortal(x))$ we can derive $Mortal(socrates)$ by resolution

$$\frac{Man(socrates),\ \forall x.(\neg Man(x) \vee Mortal(x))}{Mortal(socrates)}$$

- using the unifier $\sigma = \{x = socrates\}$, so

$$Mortal(x)[\sigma] = Mortal(socrates)$$

Example: Murder Mystery

- *d* has been murdered
- *a*, *b*, and *c* are suspects (i.e., at most one of *a*, *b*, and *c* are guilty)
- *b* claims that he did not know the victim *d* (i.e, if *b* did know *d*, then *b* is lying)
- *a* and *c* claim that *b* did know *d* (i.e., if *b* did not know *d*, then *a* and *c* are lying)
- Anyone who lies is guilty
- Prove that *b* committed the murder (is guilty)

Example: Predicate Calculus

- Express the key facts and relationships in predicate calculus

  $Guilty(a) \lor Guilty(b) \lor Guilty(c)$

  $\neg(Guilty(a) \land Guilty(b))$

  $\neg(Guilty(a) \land Guilty(c))$

  $\neg(Guilty(b) \land Guilty(c))$

  $Knows(b, d) \rightarrow Lies(b)$

  $\neg Knows(b, d) \rightarrow Lies(a)$

  $\neg Knows(b, d) \rightarrow \text{Lies}(c)$

  $\forall x.(Lies(x) \rightarrow Guilty(x))$

- Prove that $Guilty(b)$

Example: Clauses

- Convert to clausal form

1. $Guilty(a) \vee Guilty(b) \vee Guilty(c)$
2. $\neg Guilty(a) \vee \neg Guilty(b)$
3. $\neg Guilty(a) \vee \neg Guilty(c)$
4. $\neg Guilty(b) \vee \neg Guilty(c)$
5. $\neg Knows(b, d) \vee Lies(b)$
6. $Knows(b, d) \vee Lies(a)$
7. $Knows(b, d) \vee Lies(c)$
8. $\forall x.(\neg Lies(x) \vee Guilty(x))$

and add the negation of the formula we want to prove:

9. $\neg Guilty(b)$

Example: Proof

1	∀x.(¬Lies(x) ∨ Guilty(x))	(8)
2	¬Guilty(b)	(9)
3	¬Lies(b)	1, 2 by resolution with x = b
4	¬Knows(b, d) ∨ Lies(b)	(5)
5	¬Knows(b, d)	3, 4 by resolution
6	Knows(b, d) ∨ Lies(a)	(6)
7	Lies(a)	5, 6 by resolution
8	Knows(b, d) ∨ Lies(c)	(7)
9	Lies(c)	5, 8 by resolution
10	Guilty(a)	1, 7 by resolution with x = a
11	Guilty(c)	1, 9 by resolution with x = c
12	¬Guilty(a) ∨ ¬Guilty(c)	(3)
13	¬Guilty(c)	10, 12 by resolution
14	∅	11, 13 by resolution