Session Objectives

- Use logical equivalences in predicate logic.
- Define the Natural Deduction System for predicate calculus.
- Use resolution as a means of inference for propositional and predicate calculus.
- Check if a clause is a Horn clause.

Commutative Rules for Quantifiers

The order in which the two quantifiers $\forall x$ and $\forall y$ are applied to a binary predicate $P(x, y)$ has no effect:

$$\forall x(\forall y P(x, y)) \Longleftrightarrow \forall y(\forall x P(x, y))$$

As an example,

$$\forall x \in \mathbb{R}(\forall y \in \mathbb{R}(x > y \rightarrow x^2 > y^2))$$

asserts that for any pair of real numbers such that $x > y$ it follows that $x^2 > y^2$. In fact, $\forall x \forall y$ is often abbreviated to $\forall x, y$.

Similarly

$$\exists x(\exists y P(x, y)) \Longleftrightarrow \exists y(\exists x P(x, y)).$$

Mixed Quantifiers

In formulae with mixed quantifiers, the order does matter. The following two formulae are not, in general, equivalent:

$$\exists x(\forall y P(x,y)) \qquad \text{and} \qquad \forall y(\exists x P(x,y)).$$

An example will show why this is the case.

- The formula $\exists x \in \mathbb{R}(\forall y \in \mathbb{R} x > y)$ asserts that there is a real number x that is greater than every real number y. This statement is false.
- On the other hand $\forall y \in \mathbb{R}(\exists x \in \mathbb{R} x > y)$ asserts that for each real number y we can find a real number x such that $x > y$. This proposition is true since we can choose $x = y + 1$.

Distributive Rules for Quantifiers

The following are equivalent:

$$\forall x(P(x) \wedge Q(x)) \iff (\forall x P(x)) \wedge (\forall x Q(x))$$

$$\exists x(P(x) \vee Q(x)) \iff (\exists x P(x)) \vee (\exists x Q(x))$$

However, the following are not equivalent:

$$(\forall x P(x)) \vee (\forall x Q(x)) \qquad \text{and} \qquad \forall x(P(x) \vee Q(x))$$

$$(\exists x P(x)) \wedge (\exists x Q(x)) \qquad \text{and} \qquad \exists x(P(x) \wedge Q(x))$$

As a counter-example, let the domain be the set of integers $\mathbb{Z}$ and let $P(x)$ be the predicate x is even and $Q(x)$ the predicate x is odd. Clearly

$$\forall x \in \mathbb{Z}(P(x) \vee Q(x))$$

because this just states that every integer is even or odd. However, the proposition

$$(\forall x P(x)) \vee (\forall x Q(x))$$

is clearly false since it states that either all integers are even or all integers are odd.

Predicate Calculus

The purpose of predicate calculus is to support the derivation of formal proofs in a similar way to propositional calculus.

To the rules of the Natural Deduction system, we can add two pairs of new rules for the quantifiers.

Existential Quantifier

Introduction

$$\frac{P(t)}{(\exists x)P(x)} \tag{1}$$

where t is a term substitutable for x in P. If for some particular term we have proved $P(t)$ then we are allowed to deduce that $(\exists x)(P(x))$ provided that x is a variable not clashing with t.

Elimination

$$\frac{(\exists x)P(x) \quad P(x) \vdash Q}{Q} \tag{2}$$

x may not occur as a free variable in Q or in any undischarged assumptions.

Universal Quantifier

Introduction

$$\frac{P(x)}{(\forall x)P(x)} \tag{3}$$

x may not occur as a free variable in hypotheses on which $P(x)$ depends. This is equivalent to saying that we have proved $P(x)$ for arbitrary x.

Elimination

$$\frac{\forall x\, P(x)}{P(t)}$$

where t is a term substitutable for x in P.

Properties of Natural Deduction for Predicate Calculus

- It can be shown that the predicate calculus (like propositional calculus) is complete and consistent. That is, for any set $S \cup \{p\}$ of sentences of a language in predicate logic $S \vdash p$ if and only if $S \models p$. This means that our restricted inference scheme is powerful enough to capture all valid proofs.
- In natural deduction the task of finding proofs of 'real theorems' (i.e. non-trivial results) is very difficult. In propositional logic one can check whether or not a formula is provable by using truth tables, but in predicate logic, no such method is available.
- In 1936, Alonso Church proved that validity in predicate calculus is undecidable: that is, there is no algorithm that can be used to check if a statement is true.

Incompleteness of Predicate Calculus

- Gödel's Incompleteness Theorem states that in any consistent formal system that is large enough to encapsulate the natural numbers and arithmetic, there are theorems which are true but whose truth cannot be proved.
- Roughly speaking, he found a way to encode statements as numbers and showed that there was a statement with Gödel number N which read

  The statement with Gödel number N is not provable.

- If the statement were false, then it would be provable. However, that would mean that in the formal system we could prove false statements, contrary to the initial assumption. This contradiction means that the statement cannot be false.
- However, if we assume that the statement is true, there is no contradiction; the statement is true but unprovable.

Resolution for Propositional Logic

- Suppose that we want to test whether a formula ψ is provable from formulae $\{\phi_1, \phi_2, \ldots, \phi_n\}$.
- Most automatic theorem provers attempt to obtain a proof by contradiction: they negate ψ and add it to the set of premises: $\{\neg\psi, \phi_1, \phi_2, \ldots, \phi_n\}$.
- To apply the resolution rule all the formulae have to be in clause form. This means that they must be a disjunction of literals (that is, a propositional variable or its negation).
- We know that this can always be done (it is just the collection of disjunctions from CNF).

Each clause has the form

$$p_1 \vee p_2 \vee \cdots \vee p_k \vee \neg q_1 \vee \neg q_2 \vee \cdots \vee \neg q_l,$$

which is logically equivalent to

$$q_1 \wedge q_2 \wedge \cdots \wedge q_l \rightarrow p_1 \vee p_2 \vee \cdots \vee p_n.$$

In logic programming this is written as

$$p_1, p_2, \ldots, p_k \leftarrow q_1, q_2, \ldots, q_l.$$

Resolution Inference Rule

$$\frac{p_1, p_2, \ldots, p_k \leftarrow q_1, q_2, \ldots, q_l, t \quad \text{and} \quad t, r_1, r_2, \ldots, r_m \leftarrow s_1, s_2, \ldots, s_n}{p_1, p_2, \ldots, p_k r_1, r_2, \ldots, r_m \leftarrow q_1, q_2, \ldots, q_l, s_1, s_2, \ldots, s_n} \tag{4}$$

- The clause on the second line is called the resolvent of the first two clauses.

  The resolvent is formed from all the elements of the two clauses except for any variables present in both positive and negated forms.

- We continue applying this rule until the empty clause $\perp$ is reached (in which case the original formula is true, since we have reached a contradiction) or until we cannot apply resolution any more (in which case the original formula is false).
- It can be shown that in propositional calculus, resolution is sufficient to prove all true formulae.

Example

We shall try to prove that $(p \rightarrow q) \rightarrow r \vdash p \rightarrow (q \rightarrow r)$.

1. Negate the formula we are trying to prove and write both formulae in CNF: $\{\neg(\neg p \neg q) \vee r, p \wedge \neg(q \rightarrow r)\}$ which is equivalent to $\{(p \vee r) \wedge (\neg q \vee r), p \wedge q \wedge \neg r\}$.
2. Separate out the disjunctions in the CNF formulae. $\{p \vee r, \neg q \vee r, p, q, \neg r\}$.
3. Resolve $\neg q \vee r$ and q, to give r: $\{p \vee r, \neg q \vee r, p, q, \neg r, r\}$.
4. Resolve r and $\neg r$ to give the empty clause $\perp$. This contradiction means that the theorem is proved.

Exercise

Now try to prove that $p \rightarrow (q \rightarrow r) \vdash (p \rightarrow q) \rightarrow r$.

1. We can write the set of formulae as $\{\neg p \vee \neg q \vee r, \neg p \vee q, \neg r\}$.
2. Apply the resolution rule as many times as possible. (Hint: you should be able to add three new formulae).
3. Show that the empty clause cannot be derived from this set.
4. We conclude that the theorem is not true.

Solution

1. We can write the set of formulae as $\{\neg p \vee \neg q \vee r, \neg p \vee q, \neg r\}$.
2. Apply the resolution rule as many times as possible. (Hint: you should be able to add three new formulae). These are $\neg p \vee \neg q$, $\neg p \vee r$, and $\neg p$.
3. Show that the empty clause cannot be derived from this set. Every clause that uses p contains $\neg p$ but not p. Hence p can never be resolved from those clauses. Resolution produces no new clauses.
4. We conclude that the theorem is not true.

Resolution for Predicate Logic

The complexity of applying resolution to general formulae involving predicates seems to be far too great. Logic programming languages such as Prolog therefore restrict attention to Horn clauses.

definite Horn clause

$$q \leftarrow p_1, p_2, \ldots, p_n \tag{5}$$

where q and p_i are 'atomic formulae' (i.e. predicates) which are quantifier free. The assumption is that all variables are universally quantified at the front. (Unlike clauses in propositional logic, Horn clauses in predicate logic do not cover all possible formulae).

negative Horn clause

$$\leftarrow p_1, p_2, \ldots, p_n \tag{6}$$

Unification

- Unfortunately, resolution by itself is not strong enough to make all the correct inferences in predicate logic, even in the space of Horn clauses. This is because variables may have different names even though substitution is possible.
- For example, consider the clauses $P(x) \leftarrow Q(x, f(y))$ and $Q(g(z), w) \leftarrow R(w)$ for variables x, y, z and w and functions f and g. We may apply resolution, but only if we substitute $g(z)$ for x in the left-hand clause and $f(y)$ for w in the right-hand clause.
- This yields the clauses

  $$(P(g(z)) \leftarrow Q(g(z), f(y))) \quad \text{and} \quad (Q(g(z), f(y)) \leftarrow R(f(y))),$$

  for which the resolvent is $P(g(z)) \leftarrow R(f(y))$.
- Identifying terms or literals in this way is called unification and a good unification algorithm is a necessary component of a logic programming language.

Example

Consider the following problem: Alice and Belinda are pupils and study physics or chemistry (or both). The following constraints are known:

- All pupils who take physics are bad at practical work.
- Pupils who take chemistry are good at written work.
- Alice is bad at everything that Belinds is good at.
- Belinda is good at both practical and written work.

We shall use logic programming to determine if there is a pupil in the class who takes physics.

Remember that $q \leftarrow p$ is logically equivalent to $\neg p \vee q$.

$$\text{pupil(Alice)} \leftarrow \tag{7}$$

$$\text{pupil(Belinda)} \leftarrow \tag{8}$$

$$\text{physics}(x), \text{chemistry}(x) \leftarrow \text{pupil}(x) \tag{9}$$

$$\leftarrow \text{good}(x, \text{practical}), \text{physics}(x) \tag{10}$$

$$\text{good}(x, \text{written}) \leftarrow \text{chemistry}(x) \tag{11}$$

$$\leftarrow \text{good}(\text{Belinda}, x), \text{good}(\text{Alice}, x) \tag{12}$$

$$\text{good(Belinda, practical)} \leftarrow \tag{13}$$

$$\text{good(Belinda, written)} \leftarrow \tag{14}$$

$$\text{Goal:} \quad \text{goal}(x) \leftarrow \text{pupil}(x), \text{physics}(x) \tag{15}$$

$$\text{Negated goal:} \quad \leftarrow \text{goal}(z) \tag{16}$$

Notice that (9) is not a Horn clause. We have used the fact that bad is the opposite of good, so $\text{bad}(x, \text{practical}) \leftarrow \text{physics}(x)$ is equivalent to (10).

(14) and (12)		$\leftarrow$	good(Alice, written)	(17)
(11) and (17)		$\leftarrow$	chemistry(Alice)	(18)
(9) and (18)	physics(Alice)	$\leftarrow$	pupil(Alice)	(19)
(7) and (19)	physics(Alice)	$\leftarrow$		(20)
(15) and (20)	goal(Alice)	$\leftarrow$	pupil(Alice)	(21)
(7) and (21)	goal(Alice)	$\leftarrow$		(22)
(16) and (22)		$\leftarrow$		(23)

Notice how false statements (such as (17)) appear as implications with no conclusion. We deduce the empty statement, and hence the goal is true: there is a pupil who studies physics.

Session Objectives

- Use logical equivalences in predicate logic.
- Define the Natural Deduction System for predicate calculus.
- Use resolution as a means of inference for propositional and predicate calculus.
- Check if a clause is a Horn clause.