Propositional Calculus

- There are only two truth values, true and false
- Variables are typically represented by upper case letters
- Commonly used connectives
  - negation, \( \neg \)
  - and (conjunction), \( \wedge \)
  - or (disjunction), \( \vee \)
  - implication, \( \Rightarrow \)
  - equivalence, \( = \)
- Legal sentences are well-formed formulas, such as

\[
((P \land Q) \Rightarrow R) = \neg P \lor \neg Q \lor R
\]
Semantics

**Propositional Calculus Semantics**

An interpretation of a set of propositions is the assignment of a truth value, either T or F, to each propositional symbol.

- The symbol true is always assigned T, and the symbol false is assigned F.
- The interpretation or truth value for sentences is determined by:
  - The truth assignment of negation, \( \neg P \), where \( P \) is any propositional symbol, is F if the assignment to \( P \) is T and T if the the assignment to \( P \) is F.
  - The truth assignment of conjunction, \( P \land Q \), is T only when both conjuncts have truth value T; otherwise it is F.
  - The truth assignment of disjunction, \( P \lor Q \), is F only when both disjuncts have truth value F; otherwise it is T.
  - The truth assignment of implication, \( P \rightarrow Q \), is T only when the premise or symbol before the implication is T and the truth value of the consequent or symbol after the implication is F; otherwise it is always T.
  - The truth assignment of equivalence, \( P = Q \), is T only when both expressions have the same truth assignment for all possible interpretations; otherwise it is F.

**DeMorgan’s Law**

- Prove DeMorgan’s law
  \( \neg(P \land Q) = (\neg P \lor \neg Q) \)
  by constructing an appropriate truth table

- The following laws can be shown to be true by using truth tables

  \[
  \neg(\neg P) = P \\
  (P \lor Q) = (\neg P \rightarrow Q) \\
  \text{De Morgan's law: } \neg(P \land Q) = (\neg P \lor \neg Q) \\
  \text{De Morgan's law: } \neg(P \lor Q) = (\neg P \land \neg Q) \\
  \text{Distributive law: } P \land (Q \lor R) = (P \land Q) \lor (P \land R) \\
  \text{Distributive law: } P \lor (Q \land R) = (P \lor Q) \land (P \lor R) \\
  \text{Commutative law: } (P \land Q) = (Q \land P) \\
  \text{Commutative law: } (P \lor Q) = (Q \lor P) \\
  \text{Associative law: } (P \land (Q \land R)) = (P \land (Q \land R)) \\
  \text{Associative law: } ((P \lor Q) \lor R) = (P \lor (Q \lor R)) \\
  \text{Contrapositive law: } (P \rightarrow Q) = (\neg Q \rightarrow \neg P)
  \]
Predicate Calculus

• Symbols
  – true and false
  – constants, start with lowercase letter
  – variables, start with uppercase letter
  – functions start with a lowercase and many have an “arity”, that is an attached list of elements in parentheses

• Examples
  – mother(mary,bill)
  – father(bill,george)
  – likes(jim,X)
  – father(X,Y)

Qualifiers

• Universal qualifier, ∀
  ∀ X mortal(X)
  ∀ Y likes(bill,Y)

• Existential qualifier, ∃
  ∃ Y likes(bill,Y)
  ∃ Z mother(Z, george)

DEFINITION

PREDICATE CALCULUS SENTENCES

Every atomic sentence is a sentence.

1. If s is a sentence, then so is its negation, ¬s.
2. If s₁ and s₂ are sentences, then so is their conjunction, s₁ ∧ s₂.
3. If s₁ and s₂ are sentences, then so is their disjunction, s₁ ∨ s₂.
4. If s₁ and s₂ are sentences, then so is their implication, s₁ ⇒ s₂.
5. If s₁ and s₂ are sentences, then so is their equivalence, s₁ ≡ s₂.
6. If X is a variable and s a sentence, then ∀ X s is a sentence.
7. If X is a variable and s a sentence, then ∃ X s is a sentence.

• A complex sentence
  ∃X (person(X) ∧ likes(X, anchovies))
  Write this as an everyday English sentence.
Interpretation of sentences

- If D is a nonempty set of values
  each constant has a value from D
  each variable has a set of allowable
  substitutions
  each function f with arity m defines a
  mapping from $D^m$ into D
  each predicate p with arity n defines a
  mapping from $D^n$ into \{T,F\}

First Order Predicate Calculus

- In the first-order predicate calculus, quantified variables cannot refer to
  predicates or functions
  - $\forall X \text{ likes}(\text{bill}, X)$ is allowed
  - $\forall \text{ Likes} \exists X \text{ Likes}(\text{bill}, X)$ is not allowed
- Some relationships in the predicate calculus between quantifiers

\[
\begin{align*}
\neg \exists X \; p(X) &= \forall X \; \neg p(X) \\
\neg \forall X \; p(X) &= \exists X \; \neg p(X)
\end{align*}
\]

Give an “everyday” sentence illustrating each of these rules.

\[
\begin{align*}
\exists X \; p(X) &= \exists Y \; p(Y) \\
\forall X \; q(X) &= \forall Y \; q(Y) \\
\forall X \; (p(X) \land q(X)) &= \forall X \; p(X) \land \forall Y \; q(Y) \\
\exists X \; (p(X) \lor q(X)) &= \exists X \; p(X) \lor \exists Y \; q(Y)
\end{align*}
\]
Converting Quantifies

- $\neg \exists X \, p(X) = \forall X \rightarrow \neg p(X)$
- Provide an “informal” proof that this equivalence is true

Examples

- Translating English sentences into the predicate calculus
  
  If it doesn’t rain tomorrow, Tom will go to the mountains.
  $\neg \text{weather}(\text{rain}, \text{tomorrow}) \rightarrow \text{go}(\text{tom}, \text{mountains})$

  Emma is a Doberman pinscher and a good dog.
  $\text{gooddog}(\text{emma}) \land \text{isa}(\text{emma}, \text{doberman})$

  All basketball players are tall.
  $\forall X \, (\text{basketball}\_\text{player}(X) \rightarrow \text{tall}(X))$

  Some people like anchovies.
  $\exists X \, (\text{person}(X) \land \text{likes}(X, \text{anchovies})).$

  If wishes were horses, beggars would ride.
  $\text{equal}(\text{wishes}, \text{horses}) \rightarrow \text{ride}(\text{beggars})$.

  Nobody likes taxes.
  $\neg \exists X \, \text{likes}(X, \text{taxes})$.

- A classic example - the Blocks World

  $\forall X \, (\neg \exists Y \, \text{on}(Y, X) \rightarrow \text{clear}(X))$

  $\forall Y \, (\text{on}\_\text{table}(Y) \rightarrow \neg \exists X \, \text{on}(X, Y))$
**Inference Rules**

- Classic example, given
  \[ \forall X \ human(X) \Rightarrow mortal(X) \]
  \[ human(socrates) \]
  then the conclusion using modus ponens
  \[ mortal(socrates) \]
- An inference rule is *sound* if every sentence produced by the inference rule operation on S logically follows from S
- An inference rule is *complete* if it can produce every expression that logically follows from S
- Two logic systems that are sound and complete
  - Modus Ponens, developed by the Greeks, is covered in this chapter
  - Resolution, developed in the 1960s, is covered in Chapter 12

**Another Modus Ponens**

- If 20th day of the month is on Tuesday, then the 22nd day of the month must be on a Thursday.
- President Bush gives the state of the union address to a joint session of Congress on Tuesday, January 20, 2004.
- What can we conclude?
Modus Tolens - 1

• Modus Tolens has the form:
  \( P \implies Q \) and not \( Q \) therefore not \( P \)

• Translate the following sentence into the predicate calculus:
  \textit{I carry an umbrella only when it rains. If it rains it is cloudy. Today is not cloudy.}

Modus Tolens - 2

• Given “I carry an umbrella only when it rains. If it rains it is cloudy. Today is not cloudy”, use a truth table to prove “I am not carrying an umbrella” is true

• HINT: show that the three given predicates ANDed with the NOT of the conclusion is always false and therefore the conclusion must be true.
Some Definitions

**Definition**

SATISFY, MODEL, VALID, INCONSISTENT

For a predicate calculus expression $S$ and an interpretation $I$:

- If $S$ has a value of $T$ under $I$ and a particular variable assignment, then $I$ is said to satisfy $S$.
- If $I$ satisfies $S$ for all variable assignments, then $I$ is a *model* of $S$.
- $S$ is satisfiable if and only if there exist an interpretation and variable assignment that satisfy it; otherwise, it is unsatisfiable.
- A set of expressions is satisfiable if and only if there exist an interpretation and variable assignment that satisfy every element.
- If a set of expressions is not satisfiable, it is said to be inconsistent.
- If $S$ has a value $T$ for all possible interpretations, $S$ is said to be valid.

**Definition**

PROOF PROCEDURE

A proof procedure is a combination of an inference rule and an algorithm for applying that rule to a set of logical expressions to generate new sentences.

We present proof procedures for the resolution inference rule in Chapter 11.

**Definition**

LOGICALLY FOLLOWS, SOUND, AND COMPLETE

A predicate calculus expression $X$ logically follows from a set $S$ of predicate calculus expressions if every interpretation and variable assignment that satisfies $S$ also satisfies $X$.

An inference rule is sound if every predicate calculus expression produced by the rule from a set $S$ of predicate calculus expressions also logically follows from $S$.

An inference rule is complete if, given a set $S$ of predicate calculus expressions, the rule can infer every expression that logically follows from $S$.

Classical Proof Methods

**Definition**

MODUS PONES, MODUS TOLENS, AND ELIMINATION, AND INTRODUCTION, AND UNIVERSAL INSTANTIATION

If the sentences $P$ and $P \Rightarrow Q$ are known to be true, then *modus ponens* lets us infer $Q$.

Under the inference rule *modus tollens*, if $P \Rightarrow Q$ is known to be true and $Q$ is known to be false, we can infer $\neg P$.

*And elimination* allows us to infer the truth of either of the conjuncts from the truth of a conjunctive sentence. For instance, $P \land Q$ lets us conclude $P$ and $Q$ are true.

*And introduction* lets us infer the truth of a conjunction from the truth of its conjuncts. For instance, if $P$ and $Q$ are true, then $P \land Q$ is true.

*Universal instantiation* states that if any universally quantified variable in a true sentence is replaced by any appropriate term from the domain, the result is a true sentence. Thus, if $\forall x \in \text{domain}$, $\forall X p(X)$ lets us infer $p(a)$.

- We have already seen examples of modus ponens and modus tollens
- Give some everyday examples of AND elimination, AND introduction, and universal instantiation
Unification

- Assume foo is defined as foo(X, a, goo(Y))

- Some example unifications
  1) foo(red, a, goo(Z))
  2) foo(W, a, goo(jack))
  3) foo(Z, a, goo(moo(Z)))

- and the results
  1) \{fred/X, Z/Y\}
  2) \{W/X, jack/Y\}
  3) \{Z/X, moo(Z)/Y\}

Unification Algorithm

- Syntax for predicates

<table>
<thead>
<tr>
<th>PC SYNTAX</th>
<th>LIST SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(a,b)</td>
<td>(p a b)</td>
</tr>
<tr>
<td>p(((a), g(X,Y))</td>
<td>(p (l a) (g X Y))</td>
</tr>
<tr>
<td>equal(eve,mother(cain))</td>
<td>(equal eve (mother cain))</td>
</tr>
</tbody>
</table>

- A unification algorithm

```plaintext
function unity(E1, E2);
begin
  case
    both E1 and E2 are constants or the empty list: % recursion stops
      if E1 = E2 then return []; % both E1 and E2 are lists
      else return FAIL;
    E1 is a variable:
      if E1 occurs in E2 then return FAIL;
      else return (E2/E1);
    E2 is a variable:
      if E2 occurs in E1 then return FAIL;
      else return (E1/E2);
    otherwise: % both E1 and E2 are lists
      begin
        HE1 := first element of E1;
        HE2 := first element of E2;
        SUBS1 := unity(HE1, HE2);
        if SUBS1 := FAIL then return FAIL;
        TE1 := apply(SUBS1, rest of E1);
        TE2 := apply(SUBS1, rest of E2);
        SUBS2 := unity(TE1, TE2);
        if SUBS2 := FAIL then return FAIL;
        else return composition(SUBS1, SUBS2);
      end
  end
end
```

\[ g \] is any unifier of expressions E and \[ g' \] is the most general unifier of that set of expressions, then for \[ g \] applied to E there exists another unifier \[ g' \] such that \[ E = E g' \] (where \[ g' \] is the composition of unifiers, as seen above).
A Simple Resolution Proof

• Given
  If it is raining, then it must be cloudy.
  It is not cloudy today.

• Using modus tollens we would conclude it is not raining today.

• Using resolution proof we first negate what we want to prove and then show this leads to a contradiction.

• So we have
  raining(X) => cloudy(X)
  ¬ cloudy(today)
  raining(today) // negation of
  // conclusion

Resolution Proof Tree

raining(X) => cloudy(X)  raining(today)

X \ today

cloudy(today)  ¬ cloudy(today)

nil (contradiction)
A Financial Application

1. Individuals with an inadequate savings account should always make increasing the amount saved their first priority, regardless of their income.
2. Individuals with an adequate savings account and an adequate income should consider a riskier but potentially more profitable investment in the stock market.
3. Individuals with a lower income who already have an adequate savings account may want to consider splitting their surplus income between savings and stocks, to increase the cushion in savings while attempting to increase their income through stocks.

- Expressed in the predicate calculus

1. \( \text{savings\_account(inadequate)} \Rightarrow \text{investment(savings)} \).
2. \( \text{savings\_account(adequate)} \land \text{income(adequate)} \Rightarrow \text{investment(stocks)} \).
3. \( \text{savings\_account(adequate)} \land \text{income(inadequate)} \Rightarrow \text{investment(combination)} \).
4. \( \forall Y \text{ amount\_saved(X)} \land \exists Y \text{ (dependents(Y) } \land \text{ greater(X, minsavings(Y))} \Rightarrow \text{savings\_account(adequate)} \).
5. \( \forall Y \text{ amount\_saved(X)} \land \exists Y \text{ (dependents(Y) } \land \neg \text{ greater(X, minsavings(Y))} \Rightarrow \text{savings\_account(inadequate)} \).
6. \( \forall Y \text{ earnings(X, steady)} \land \exists Y \text{ (dependents(Y) } \land \text{ greater(X, minincome(Y))} \Rightarrow \text{income(adequate)} \).
7. \( \forall Y \text{ earnings(X, steady)} \land \exists Y \text{ (dependents(Y) } \land \neg \text{ greater(X, minincome(Y))} \Rightarrow \text{income(inadequate)} \).
8. \( \forall Y \text{ earnings(X, unsteady)} \Rightarrow \text{income(inadequate)} \).
9. \text{amount\_saved(22000)}.
10. \text{earnings(25000, steady)}.
11. \text{dependents(3)}.

Finding an Investment

- Use the previous set of data and assume that
  \( \text{minsavings(X)} = 5000 \times X \)
  \( \text{minincome(X)} = 14000 + (4000 \times X) \)
- Trace through the unifications of various rules to show that this logical database yields
  \( \text{investment(combination)} \)