The Predicate Calculus

2.0 Introduction
2.1 The Propositional Calculus
2.2 The Predicate Calculus
2.3 Using Inference Rules to Produce Predicate Calculus Expressions
2.4 Application: A Logic-Based Financial Advisor
2.5 Epilogue and References
2.6 Exercises
DEFINITION

PROPOSITIONAL CALCULUS SYMBOLS

The *symbols* of propositional calculus are the propositional symbols:

\[ P, Q, R, S, \ldots \]

truth symbols:

true, false

and connectives:

\[ \land, \lor, \neg, \rightarrow, \equiv \]
**Definition**

**Propositional Calculus Sentences**

Every propositional symbol and truth symbol is a sentence.

For example: true, P, Q, and R are sentences.

The *negation* of a sentence is a sentence.

For example: ¬P and ¬false are sentences.

The *conjunction*, or *and*, of two sentences is a sentence.

For example: P ∧ ¬P is a sentence.

The *disjunction*, or *or*, of two sentences is a sentence.

For example: P ∨ ¬P is a sentence.

The *implication* of one sentence from another is a sentence.

For example: P → Q is a sentence.

The *equivalence* of two sentences is a sentence.

For example: P ∨ Q ≡ R is a sentence.

Legal sentences are also called *well-formed formulas* or *WFFs*. 
DEFINITION

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 assignment to \( P \) is F.

- The truth assignment of *conjunction*, \( \land \), is T only when both conjuncts have truth value T; otherwise it is F.

- The truth assignment of *disjunction*, \( \lor \), is F only when both disjuncts have truth value F; otherwise it is T.

- The truth assignment of *implication*, \( \rightarrow \), is F 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 T.

- The truth assignment of *equivalence*, \( \equiv \), is T only when both expressions have the same truth assignment for all possible interpretations; otherwise it is F.
For propositional expressions $P$, $Q$ and $R$:

\[ \neg (\neg P) \equiv P \]

\[ (P \lor Q) \equiv (\neg P \rightarrow Q) \]

the contrapositive law: $(P \rightarrow Q) \equiv (\neg Q \rightarrow \neg P)$

de Morgan’s law: $\neg (P \lor Q) \equiv (\neg P \land \neg Q)$ and $\neg (P \land Q) \equiv (\neg P \lor \neg Q)$

the commutative laws: $(P \land Q) \equiv (Q \land P)$ and $(P \lor Q) \equiv (Q \lor P)$

the associative law: $((P \land Q) \land R) \equiv (P \land (Q \land R))$

the associative law: $((P \lor Q) \lor R) \equiv (P \lor (Q \lor R))$

the distributive law: $P \lor (Q \land R) \equiv (P \lor Q) \land (P \lor R)$

the distributive law: $P \land (Q \lor R) \equiv (P \land Q) \lor (P \land R)$
**Figure 2.1:** Truth table for the operator $\land$.

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$P \land Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Figure 2.2: Truth table demonstrating the equivalence of $P \land Q$ and $\neg P / Q$.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
DEFINITION

PREDICATE CALCULUS SYMBOLS

The alphabet that makes up the symbols of the predicate calculus consists of:

1. The set of letters, both upper- and lowercase, of the English alphabet.
2. The set of digits, 0, 1, ..., 9.
3. The underscore, _.

Symbols in the predicate calculus begin with a letter and are followed by any sequence of these legal characters.

Legitimate characters in the alphabet of predicate calculus symbols include

a R 6 9 p _ z

Examples of characters not in the alphabet include

# % @ / & “ ”

Legitimate predicate calculus symbols include

George fire3 tom_and_jerry bill XXXX friends_of

Examples of strings that are not legal symbols are

3jack “no blanks allowed” ab%cd ***71 duck!!!
DEFINITION

SYMBOLS and TERMS

Predicate calculus symbols include:

1. *Truth symbols* **true** and **false** (these are reserved symbols).

2. *Constant symbols* are symbol expressions having the first character lowercase.

3. *Variable symbols* are symbol expressions beginning with an uppercase character.

4. *Function symbols* are symbol expressions having the first character lowercase. Functions have an attached arity indicating the number of elements of the domain mapped onto each element of the range.

A *function expression* consists of a function constant of arity \( n \), followed by \( n \) terms, \( t_1, t_2, ..., t_n \), enclosed in parentheses and separated by commas.

A predicate calculus *term* is either a constant, variable, or function expression.
DEFINITION

PREDICATES and ATOMIC SENTENCES

Predicate symbols are symbols beginning with a lowercase letter.

Predicates have an associated positive integer referred to as the *arity* or “argument number” for the predicate. Predicates with the same name but different arities are considered distinct.

An atomic sentence is a predicate constant of arity $n$, followed by $n$ terms, $t_1, t_2, ..., t_n$, enclosed in parentheses and separated by commas.

The truth values, **true** and **false**, are also atomic sentences.
DEFINITION

PREDICATE CALCULUS SENTENCES

Every atomic sentence is a sentence.

1. If $s$ is a sentence, then so is its negation, $\neg s$.
2. If $s_1$ and $s_2$ are sentences, then so is their conjunction, $s_1 \land s_2$.
3. If $s_1$ and $s_2$ are sentences, then so is their disjunction, $s_1 \lor s_2$.
4. If $s_1$ and $s_2$ are sentences, then so is their implication, $s_1 \rightarrow s_2$.
5. If $s_1$ and $s_2$ are sentences, then so is their equivalence, $s_1 \equiv s_2$.
6. If $X$ is a variable and $s$ a sentence, then $\forall X s$ is a sentence.
7. If $X$ is a variable and $s$ a sentence, then $\exists X s$ is a sentence.
verify_sentence algorithm

function verify_sentence(expression);
begin
  case
    expression is an atomic sentence: return SUCCESS;
    expression is of the form $Q \, X \, s$, where $Q$ is either $\forall$ or $\exists$, $X$ is a variable, and $s$ is an expression;
      if verify_sentence($s$) returns SUCCESS
        then return SUCCESS
      else return FAIL;
    expression is of the form $\neg \, s$:
      if verify_sentence($s$) returns SUCCESS
        then return SUCCESS
      else return FAIL;
    expression is of the form $s_1 \, \text{op} \, s_2$, where \text{op} is a binary logical operator:
      if verify_sentence($s_1$) returns SUCCESS and verify_sentence($s_2$) returns SUCCESS
        then return SUCCESS
      else return FAIL;
    otherwise: return FAIL
  end
end.
DEFINITION

INTERPRETATION

Let the domain $D$ be a nonempty set.

An interpretation over $D$ is an assignment of the entities of $D$ to each of the constant, variable, predicate, and function symbols of a predicate calculus expression, such that:

1. Each constant is assigned an element of $D$.
2. Each variable is assigned to a nonempty subset of $D$; these are the allowable substitutions for that variable.
3. Each function $f$ of arity $m$ is defined on $m$ arguments of $D$ and defines a mapping from $D^m$ into $D$.
4. Each predicate $p$ of arity $n$ is defined on $n$ arguments from $D$ and defines a mapping from $D^n$ into $\{T, F\}$.
DEFINITION

TRUTH VALUE OF PREDICATE CALCULUS EXPRESSIONS

Assume an expression $E$ and an interpretation $I$ for $E$ over a nonempty domain $D$. The truth value for $E$ is determined by:

1. The value of a constant is the element of $D$ it is assigned to by $I$.
2. The value of a variable is the set of elements of $D$ it is assigned to by $I$.
3. The value of a function expression is that element of $D$ obtained by evaluating the function for the parameter values assigned by the interpretation.
4. The value of truth symbol “true” is $T$ and “false” is $F$.
5. The value of an atomic sentence is either $T$ or $F$, as determined by the
6. The value of the negation of a sentence is $T$ if the value of the sentence is $F$ and is $F$ if the value of the sentence is $T$.
7. The value of the conjunction of two sentences is $T$ if the value of both sentences is $T$ and is $F$ otherwise.
8.-10. The truth value of expressions using $\lor$, $\rightarrow$, and $\equiv$ is determined from the value of their operands as defined in Section 2.1.2.

Finally, for a variable $X$ and a sentence $S$ containing $X$:

11. The value of $\forall X S$ is $T$ if $S$ is $T$ for all assignments to $X$ under $I$, and it is $F$ otherwise.
12. The value of $\exists X S$ is $T$ if there is an assignment to $X$ in the interpretation under which $S$ is $T$; otherwise it is $F$. 
DEFINITION

FIRST-ORDER PREDICATE CALCULUS

First-order predicate calculus allows quantified variables to refer to objects in the domain of discourse and not to predicates or functions.
Figure 2.3: A blocks world with its predicate calculate description.
**Definition**

Satisfy, Model, Valid, Inconsistent

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

- If $X$ has a value of $T$ under $I$ and a particular variable assignment, then $I$ is said to satisfy $X$.

- If $I$ satisfies $X$ for all variable assignments, then $I$ is a model of $X$.

$X$ 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 $X$ has a value $T$ for all possible interpretations, $X$ 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 12.
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 $S$. 
DEFINITION

MODUS PONENTS, MODUS TOLLENS, 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 $a$ is from the domain of $X$, $\forall X p(X)$ lets us infer $p(a)$. 
DEFINITION

MOST GENERAL UNIFIER (mgu)

If \( s \) is any unifier of expressions \( E \), and \( g \) is the most general unifier of that set of expressions, then for \( s \) applied to \( E \) there exists another unifier \( s' \) such that \( Es = Egs' \), where \( Es \) and \( Egs' \) are the composition of unifiers applied to the expression \( E \).
function unify(E1, E2);
    begin
        case
            both E1 and E2 are constants or the empty list: %recursion stops
                if E1 = E2 then return {} 
                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}
            either E1 or E2 are empty then return FAIL %the lists are of different sizes
            otherwise: %both E1 and E2 are lists
                begin
                    HE1 := first element of E1;
                    HE2 := first element of E2;
                    SUBS1 := unify(HE1,HE2);
                    if SUBS1 := FAIL then return FAIL;
                    TE1 := apply(SUBS1, rest of E1);
                    TE2 := apply (SUBS1, rest of E2);
                    SUBS2 := unify(TE1, TE2);
                    if SUBS2 = FAIL then return FAIL;
                        else return composition(SUBS1,SUBS2)
                end
        end case
    end
end
Figure 2.5: Further steps in the unification of \((\text{parents } X \ (\text{father } X) \ (\text{mother bill}))\) and \((\text{parents bill } (\text{father bill}) \ Y))\).
Figure 2.6: Final trace of the unification of (parents X (father X) (mother bill)) and (parents bill (father bill) Y).
1. \( \text{savings\_account(inaequate)} \rightarrow \text{investment(savings)}. \)

2. \( \text{savings\_account(adequate)} \land \text{income(adequate)} \rightarrow \text{investment(stocks)}. \)

3. \( \text{savings\_account(adequate)} \land \text{income(inaequate)} \rightarrow \text{investment(combination)}. \)

4. \( \forall \text{amount\_saved}(X) \land \exists Y (\text{dependents}(Y) \land \text{greater}(X, \text{minsavings}(Y))) \rightarrow \text{savings\_account(adequate)}. \)

5. \( \forall X \text{amount\_saved}(X) \land \exists Y (\text{dependents}(Y) \land \neg \text{greater}(X, \text{minsavings}(Y))) \rightarrow \text{savings\_account(inaequate)}. \)

6. \( \forall X \text{earnings}(X, \text{steady}) \land \exists Y (\text{dependents}(Y) \land \text{greater}(X, \text{minincome}(Y))) \rightarrow \text{income(adequate)}. \)

7. \( \forall X \text{earnings}(X, \text{steady}) \land \exists Y (\text{dependents}(Y) \land \neg \text{greater}(X, \text{minincome}(Y))) \rightarrow \text{income(inaequate)}. \)

8. \( \forall X \text{earnings}(X, \text{unsteady}) \rightarrow \text{income(inaequate)}. \)

9. \( \text{amount\_saved(22000)}. \)

10. \( \text{earnings(25000, steady)}. \)

11. \( \text{dependents(3)}. \)