THE PREDICATE CALCULUS

We come to the full possession of our power of drawing inferences, the last of our faculties; for it is not so much a natural gift as a long and difficult art.

—C. S. Pierce

The essential quality of a proof is to compel belief.

—Fermat

2.0 Introduction

In this chapter we introduce the predicate calculus as a representation language for artificial intelligence. The importance of the predicate calculus was discussed in the introduction to Part II; these advantages include a well-defined *formal semantics* and *sound* and *complete* inference rules. This chapter begins with a brief review of the propositional calculus (Section 2.1). Section 2.2 defines the syntax and semantics of the predicate calculus. In Section 2.3 we discuss predicate calculus inference rules and their use in problem solving. Finally, the chapter demonstrates the use of the predicate calculus to implement a knowledge base of financial investment advice.

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.

For example,

∀ (Likes) Likes(george,kate)

is not a well-formed expression in the first-order predicate calculus. There are *higher-order* predicate calculi where such expressions are meaningful. Some researchers (McCarthy

1968, Appelt 1985) have used higher-order languages to represent knowledge in natural language understanding programs.

Many grammatically correct English sentences can be represented in the first-order predicate calculus using the symbols, connectives, and variable symbols defined in this section. It is important to note that there is no unique mapping of sentences into predicate calculus expressions; in fact, an English sentence may have any number of different predicate calculus representations. A major challenge for AI programmers is to find a scheme for using these predicates that optimizes the expressiveness and efficiency of the resulting representation. Examples of English sentences represented in predicate calculus are:

If it doesn't rain on Monday, Tom will go to the mountains.
$\neg$ weather(rain, monday) $\rightarrow$ go(tom, mountains)

Emma is a Doberman pinscher and a good dog.
gooddog(emma) $\wedge$ isa(emma, doberman)

All basketball players are tall.
$\forall$ X (basketball_player(X) $\rightarrow$ tall(X))

Some people like anchovies.
$\exists$ X (person(X) $\wedge$ likes(X, anchovies))

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

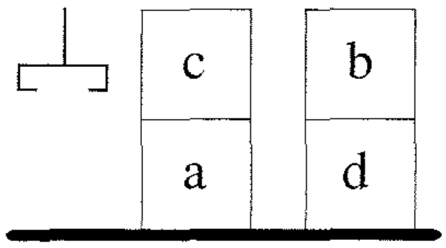
Nobody likes taxes.
$\neg$ $\exists$ X likes(X, taxes)

2.2.3 A "Blocks World" Example of Semantic Meaning

We conclude this section by giving an extended example of a truth value assignment to a set of predicate calculus expressions. Suppose we want to model the blocks world of Figure 2.3 to design, for example, a control algorithm for a robot arm. We can use predicate calculus sentences to represent the qualitative relationships in the world: does a given block have a clear top surface? can we pick up block a? etc. Assume that the computer has knowledge of the location of each block and the arm and is able to keep track of these locations (using three-dimensional coordinates) as the hand moves blocks about the table.

We must be very precise about what we are proposing with this "blocks world" example. First, we are creating a set of predicate calculus expressions that is to represent a static snapshot of the blocks world problem domain. As we will see in Section 2.3, this set of blocks offers an *interpretation* and a possible *model* for the set of predicate calculus expressions.

Second, the predicate calculus is *declarative*, that is, there is no assumed timing or order for considering each expression. Nonetheless, in the planning section of this book, Section 8.4, we will add a "procedural semantics", or a clearly specified methodology for evaluating these expressions over time. A concrete example of a procedural semantics for



on(c,a)
on(b,d)
ontable(a)
ontable(d)
clear(b)
clear(c)
hand_empty

Figure 2.3 A blocks world with its predicate calculus description.

predicate calculus expressions is PROLOG, Chapter 15. This situation calculus we are creating will introduce a number of issues, including the *frame problem* and the issue of *non-monotonicity* of logic interpretations, that will be addressed later in this book. For this example, however, it is sufficient to say that our predicate calculus expressions will be evaluated in a left-to-right fashion.

To pick up a block and stack it on another block, both blocks must be clear. In Figure 2.3, block a is not clear. Because the arm can move blocks, it can change the state of the world and clear a block. Suppose it removes block c from block a and updates the knowledge base to reflect this by deleting the assertion on(c,a). The program needs to be able to infer that block a has become clear.

The following rule describes when a block is clear:

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

That is, for all X, X is clear if there does not exist a Y such that Y is on X.

This rule not only defines what it means for a block to be clear but also provides a basis for determining how to clear blocks that are not. For example, block d is not clear, because if variable X is given value d, substituting b for Y will make the statement false. Therefore, to make this definition true, block b must be removed from block d. This is easily done because the computer has a record of all the blocks and their locations.

Besides using implications to define when a block is clear, other rules may be added that describe operations such as stacking one block on top of another. For example: to stack X on Y, first empty the hand, then clear X, then clear Y, and then pick_up X and put_down X on Y.

$$\forall X \forall Y ((\text{hand\_empty} \wedge \text{clear}(X) \wedge \text{clear}(Y) \wedge \text{pick\_up}(X) \wedge \text{put\_down}(X,Y))$$
$$\rightarrow \text{stack}(X,Y))$$

Modus ponens is a sound inference rule. If we are given an expression of the form P → Q and another expression of the form P such that both are true under an interpretation I, then modus ponens allows us to infer that Q is also true for that interpretation. Indeed, because modus ponens is sound, Q is true for *all* interpretations for which P and P → Q are true.

Modus ponens and a number of other useful inference rules are defined below.

DEFINITION

MODUS PONENS, MODUS TOLLENS, AND ELIMINATION, AND INTRODUCTION, and UNIVERSAL INSTANTIATION

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

Under the inference rule *modus tollens*, if P → Q is known to be true and Q is known to be false, we can infer ¬ P.

And elimination allows us to infer the truth of either of the conjuncts from the truth of a conjunctive sentence. For instance, P ∧ 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 ∧ 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, ∀ X p(X) lets us infer p(a).

As a simple example of the use of modus ponens in the propositional calculus, assume the following observations: "if it is raining then the ground will be wet" and "it is raining." If P denotes "it is raining" and Q is "the ground is wet" then the first expression becomes P → Q. Because it is indeed now raining (P is true), our set of axioms becomes

P → Q
P

Through an application of modus ponens, the fact that the ground is wet (Q) may be added to the set of true expressions.

Modus ponens can also be applied to expressions containing variables. Consider as an example the common syllogism "all men are mortal and Socrates is a man; therefore Socrates is mortal." "All men are mortal" may be represented in predicate calculus by

∀ X (man(X) → mortal(X)).

"Socrates is a man" is

man(socrates).

Because the X in the implication is universally quantified, we may substitute any value in the domain for X and still have a true statement under the inference rule of universal instantiation. By substituting socrates for X in the implication, we infer the expression

man(socrates) → mortal(socrates).

We can now apply modus ponens and infer the conclusion mortal(socrates). This is added to the set of expressions that logically follow from the original assertions. An algorithm called *unification* can be used by an automated problem solver to determine that socrates may be substituted for X in order to apply modus ponens. Unification is discussed in Section 2.3.2.

Chapter 13 discusses a more powerful rule of inference called *resolution*, which is the basis of many automated reasoning systems.

2.4 Application: A Logic-Based Financial Advisor

As a final example of the use of predicate calculus to represent and reason about problem domains, we design a simple financial advisor using predicate calculus. Although a simple example, it illustrates many of the issues involved in realistic applications.

The function of the advisor is to help a user decide whether to invest in a savings account or the stock market. Some investors may want to split their money between the two. The investment that will be recommended for individual investors depends on their income and the current amount they have saved according to the following criteria:

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.

The adequacy of both savings and income is determined by the number of dependents an individual must support. Our rule is to have at least $5,000 in the bank for each dependent. An adequate income must be a steady income and supply at least $15,000 per year plus an additional $4,000 for each dependent.

To automate this advice, we translate these guidelines into sentences in the predicate calculus. The first task is to determine the major features that must be considered. Here, they are the adequacy of the savings and the income. These are represented by the predicates savings_account and income, respectively. Both of these are unary pre- dicates, and their argument could be either adequate or inadequate. Thus,

```
savings_account(adequate).
savings_account(inadequate).
income(adequate).
income(inadequate).
```

are their possible values.

Conclusions are represented by the unary predicate investment, with possible values of its argument being stocks, savings, or combination (implying that the investment should be split).

Using these predicates, the different investment strategies are represented by implications. The first rule, that individuals with inadequate savings should make increased savings their main priority, is represented by

savings_account(inadequate) $\rightarrow$ investment(savings).

Similarly, the remaining two possible investment alternatives are represented by

$\text{savings\_account(adequate)} \wedge \text{income(adequate)} \rightarrow \text{investment(stocks)}.$
$\text{savings\_account(adequate)} \wedge \text{income(inadequate)}$
$\quad \rightarrow \text{investment(combination)}.$

Next, the advisor must determine when savings and income are adequate or inadequate. This will also be done using implication. The need to do arithmetic calculations requires the use of functions. To determine the minimum adequate savings, the function minsavings is defined. minsavings takes one argument, the number of dependents, and returns 5000 times that argument.

Using minsavings, the adequacy of savings is determined by the rules

$\forall$ X amount_saved(X) $\wedge$ $\exists$ Y (dependents(Y) $\wedge$ greater(X, minsavings(Y)))
  $\rightarrow$ savings_account(adequate).
$\forall$ X amount_saved(X) $\wedge$ $\exists$ Y (dependents(Y) $\wedge$ $\neg$ greater(X, minsavings(Y)))
  $\rightarrow$ savings_account(inadequate).

where minsavings(X) $\equiv$ 5000 $*$ X.

In these definitions, amount_saved(X) and dependents(Y) assert the current amount in savings and the number of dependents of an investor; greater(X,Y) is the standard arithmetic test for one number being greater than another and is not formally defined in this example.

Similarly, a function minincome is defined as

minincome(X) $\equiv$ 15000 + (4000 $*$ X).

minincome is used to compute the minimum adequate income when given the number of dependents. The investor's current income is represented by a predicate, earnings. Because an adequate income must be both steady and above the minimum, earnings takes two arguments: the first is the amount earned, and the second must be equal to either steady or unsteady. The remaining rules needed for the advisor are

$\forall$ X earnings(X, steady) $\wedge$ $\exists$ Y (dependents(Y) $\wedge$ greater(X, minincome(Y)))
  $\rightarrow$ income(adequate).

$\forall$ X earnings(X, steady) $\wedge$ $\exists$ Y (dependents(Y) $\wedge$ $\neg$ greater(X, minincome(Y)))
  $\rightarrow$ income(inadequate).

$\forall$ X earnings(X, unsteady) $\rightarrow$ income(inadequate).

In order to perform a consultation, a description of a particular investor is added to this set of predicate calculus sentences using the predicates amount_saved, earnings, and dependents. Thus, an individual with three dependents, \$22,000 in savings, and a steady income of \$25,000 would be described by

amount_saved(22000).

```
earnings(25000, steady).
dependents(3).
```

This yields a logical system consisting of the following sentences:

1. savings_account(inadequate) → investment(savings).
2. savings_account(adequate) ∧ income(adequate) → investment(stocks).
3. savings_account(adequate) ∧ income(inadequate)
   → investment(combination).
4. ∀ X amount_saved(X) ∧ ∃ Y (dependents(Y) ∧
   greater(X, minsavings(Y))) → savings_account(adequate).
5. ∀ X amount_saved(X) ∧ ∃ Y (dependents(Y) ∧
   ¬ greater(X, minsavings(Y))) → savings_account(inadequate).
6. ∀ X earnings(X, steady) ∧ ∃ Y (dependents (Y) ∧
   greater(X, minincome(Y))) → income(adequate).
7. ∀ X earnings(X, steady) ∧ ∃ Y (dependents(Y) ∧
   ¬ greater(X, minincome(Y))) → income(inadequate).
8. ∀ X earnings(X, unsteady) → income(inadequate).
9. amount_saved(22000).
10. earnings(25000, steady).
11. dependents(3).

where minsavings(X) ≡ 5000 ∗ X and minincome(X) ≡ 15000 + (4000 ∗ X).

This set of logical sentences describes the problem domain. The assertions are numbered so that they may be referenced in the following trace.

Using unification and modus ponens, a correct investment strategy for this individual may be inferred as a logical consequence of these descriptions. A first step would be to unify the conjunction of 10 and 11 with the first two components of the premise of 7; i.e.,

```
earnings(25000,steady) ∧ dependents(3)
```

unifies with

```
earnings(X,steady) ∧ dependents(Y)
```

under the substitution {25000/X, 3/Y}. This substitution yields the new implication:

```
earnings(25000, steady) ∧ dependents(3) ∧ ¬ greater(25000, minincome(3))
  → income(inadequate).
```

Evaluating the function minincome yields the expression

earnings(25000, steady) $\wedge$ dependents(3) $\wedge \neg$ greater(25000, 27000)
$\rightarrow$ income(inadequate).

Because all three components of the premise are individually true, by 10, 3, and the mathematical definition of greater, their conjunction is true and the entire premise is true. Modus ponens may therefore be applied, yielding the conclusion income(inadequate). This is added as assertion 12.

12. income(inadequate).

Similarly,

amount_saved(22000) $\wedge$ dependents(3)

unifies with the first two elements of the premise of assertion 4 under the substitution {22000/X, 3/Y}, yielding the implication

amount_saved(22000) $\wedge$ dependents(3) $\wedge$ greater(22000, minsavings(3))
$\rightarrow$ savings_account(adequate).

Here, evaluating the function minsavings(3) yields the expression

amount_saved(22000) $\wedge$ dependents(3) $\wedge$ greater(22000, 15000)
$\rightarrow$ savings_account(adequate).

Again, because all of the components of the premise of this implication are true, the entire premise evaluates to true and modus ponens may again be applied, yielding the conclusion savings_account(adequate), which is added as expression 13.

13. savings_account(adequate).

As an examination of expressions 3, 12, and 13 indicates, the premise of implication 3 is also true. When we apply modus ponens a third time, the conclusion is investment(combination). This is the suggested investment for this individual.

This example illustrates how predicate calculus may be used to reason about a realistic problem, drawing correct conclusions by applying inference rules to the initial problem description. We have not discussed exactly how an algorithm can determine the correct inferences to make to solve a given problem or the way in which this can be implemented on a computer. These topics are presented in Chapters 3, 4, and 6.

Exercises from Luger chapter 2, page 78

10. Jane Doe has four dependents, a steady income of \$30,000, and \$15,000 in her savings account. Add the appropriate predicates describing her situation to the general investment advisor of the example in Section 2.4 and perform the unifications and inferences needed to determine her suggested investment.

12. The following story is from N. Wirth's (1976) *Algorithms + data structures = programs*.
I married a widow (let's call her W) who has a grown-up daughter (call her D). My father (F), who visited us quite often, fell in love with my step-daughter and married her. Hence my father became my son-in-law and my step-daughter became my mother. Some months later, my wife gave birth to a son (S_1), who became the brother-in-law of my father, as well as my uncle. The wife of my father, that is, my step-daughter, also had a son (S_2).
Using predicate calculus, create a set of expressions that represent the situation in the above story. Add expressions defining basic family relationships such as the definition of father-in-law and use modus ponens on this system to prove the conclusion that "I am my own grandfather."