# Introduction: What Is Modal Logic?

Strictly speaking, modal logic studies reasoning that involves the use of the expressions 'necessarily' and 'possibly'. The main idea is to introduce the symbols □ (necessarily) and ◊ (possibly) to a system of logic so that it is able to distinguish three different *modes* of assertion: □ A (A is necessary), A (A is true), and ◊ A (A is possible). Introducing these symbols (or operators) would seem to be essential if logic is to be applied to judging the accuracy of philosophical reasoning, for the concepts of necessity and possibility are ubiquitous in philosophical discourse.

However, at the very dawn of the invention of modal logics, it was recognized that necessity and possibility have kinships with many other philosophically important expressions. So the term 'modal logic' is also used more broadly to cover a whole family of logics with similar rules and a rich variety of different operators. To distinguish the narrow sense, some people use the term 'alethic logic' for logics of necessity and possibility. A list describing some of the better known of these logics follows.

| **System** | **Symbols** | **Expression Symbolized** |
| --- | --- | --- |
| Modal logic | □ | It is necessary that |
| (or Alethic logic) | ◊ | It is possible that |
| Tense logic | G | It will always be the case that |
| | F | It will be the case that |
| | H | It has always been the case that |
| | P | It was the case that |
| Deontic logic | O | It is obligatory that |
| | P | It is permitted that |
| | F | It is forbidden that |

| | | |
|---|---|---|
| Locative logic | Tx | It is the case at x that |
| Doxastic logic | Bx | x believes that |
| Epistemic logic | Kx | x knows that |

This book will provide you with an introduction to all these logics, and it will help sketch out the relationships among the different systems. The variety found here might be somewhat bewildering, especially for the student who expects uniformity in logic. Even within the above subdivisions of modal logic, there may be many different systems. I hope to convince you that this variety is a source of strength and flexibility and makes for an interesting world well worth exploring.

# 1

# The System K: A Foundation for Modal Logic

## 1.1. The Language of Propositional Modal Logic

We will begin our study of modal logic with a basic system called K in honor of the famous logician Saul *K*ripke. K serves as the foundation for a whole family of systems. Each member of the family results from strengthening K in some way. Each of these logics uses its own symbols for the expressions it governs. For example, modal (or alethic) logics use $\Box$ for necessity, tense logics use **H** for what has always been, and deontic logics use **O** for obligation. The rules of K characterize each of these symbols and many more. Instead of rewriting K rules for each of the distinct symbols of modal logic, it is better to present K using a generic operator. Since modal logics are the oldest and best known of those in the modal family, we will adopt $\Box$ for this purpose. So $\Box$ need not mean *necessarily* in what follows. It stands proxy for many different operators, with different meanings. In case the reading does not matter, you may simply call $\Box$A 'box A'.

First we need to explain what a *language for propositional modal logic* is. The symbols of the language are $\perp$, $\rightarrow$, $\Box$; the propositional variables: p, q, r, p′, and so forth; and parentheses. The symbol $\perp$ represents a contradiction, $\rightarrow$ represents 'if . . then', and $\Box$ is the modal operator. A *sentence of propositional modal logic* is defined as follows:

$\perp$ and any propositional variable is a sentence.
If A is a sentence, then $\Box$A is a sentence.
If A is a sentence and B is a sentence, then (A$\rightarrow$B) is a sentence.
No other symbol string is a sentence.

In this book, we will use letters 'A', 'B', 'C' for sentences. So A may be a propositional variable, p, or something more complex like (p→q), or ((p→ ⊥)→q). To avoid eyestrain, we usually drop the outermost set of parentheses. So we abbreviate (p→q) to p→q. (As an aside for those who are concerned about use-mention issues, here are the conventions of this book. We treat '⊥', '→', '□', and so forth as *used* to refer to symbols with similar shapes. It is also understood that '□A', for example, refers to the result of concatenating □ with the sentence A.)

The reader may be puzzled about why our language does not contain negation: ~ and the other familiar logical connectives: &, ∨, and ↔. Although these symbols are not in our language, they may be introduced as abbreviations by the following definitions:

| | | |
|---|---|---|
| (Def~) | ~A | $=_{df}$ A→⊥ |
| (Def&) | A&B | $=_{df}$ ~(A→~B) |
| (Def∨) | A∨B | $=_{df}$ ~A→B |
| (Def↔) | A↔B | $=_{df}$ (A→B)&(B→A) |

Sentences that contain symbols introduced by these definitions are understood as shorthand for sentences written entirely with → and ⊥. So, for example, ~p abbreviates p→⊥, and we may replace one of these with the other whenever we like. The same is true of complex sentences. For example, ~p&q is understood to be the abbreviation for (p→⊥)&q, which by (Def&) amounts to ~((p→⊥)→~q). Replacing the two occurrences of ~ in this sentence, we may express the result in the language of K as follows: ((p→⊥)→(q→⊥))→⊥. Of course, using such primitive notation is very cumbersome, so we will want to take advantage of the abbreviations as much as possible. Still, it simplifies much of what goes on in this book to assume that when the chips are down, all sentences are written with only the symbols ⊥, →, and □.

**EXERCISE 1.1** Convert the following sentences into the primitive notation of K.

a) ~~p
b) ~p&~q
c) p∨(q&r)
d) ~(p∨q)
e) ~(p↔q)

Our use of $\perp$ and the definition for negation (Def~) may be unfamiliar to you. However, it is not difficult to see why (Def~) works. Since $\perp$ indicates a contradiction, $\perp$ is always false. By the truth table for material implication, $A\rightarrow\perp$ is true (T) iff either A is false (F) or $\perp$ is T. But, as we said, $\perp$ cannot be T. Therefore $A\rightarrow\perp$ is T iff A is F. So the truth table for $A\rightarrow\perp$ corresponds exactly to the truth table for negation.

The notion of an argument is fundamental to logic. In this book, an *argument* H / C is composed of a list of sentences H, which are called the *hypotheses,* and a sentence C called the *conclusion.* In the next section, we will introduce rules of proof for arguments. When argument H / C is provable (in some system), we write '$H \vdash C$'. Since there are many different systems in this book, and it may not be clear which system we have in mind, we subscript the name of the system S (thus: $H \vdash_S C$) to make matters clear. According to these conventions, p, ~q→~p / q is the argument with hypotheses p and ~q→~p and conclusion q. The expression '$p, {\sim}q\rightarrow{\sim}p \vdash_K q$' indicates that the argument p, ~q→~p / q has a proof in the system K.

## 1.2. Natural Deduction Rules for Propositional Logic: PL

Let us begin the description of K by introducing a system of rules called PL (for *p*ropositional *l*ogic). We will use natural deduction rules in this book because they are especially convenient both for presenting and finding proofs. In general, natural deduction systems are distinguished by the fact that they allow the introduction of (provisional) assumptions or hypotheses, along with some mechanism (such as vertical lines or dependency lists) for keeping track of which steps of the proof depend on which hypotheses. Natural deduction systems typically include the rules Conditional Proof (also known as Conditional Introduction) and Indirect Proof (also known as Reductio ad Absurdum or Negation Introduction). We assume the reader is already familiar with some natural deduction system for propositional logic. In this book, we will use vertical lines to keep track of subproofs. The notation:

$$\begin{array}{|l} \underline{A} \\ \vdots \\ B \end{array}$$

indicates that B has been proven from the hypothesis A. The dots indicate intervening steps, each of which follows from previous steps by one of the following five rules. The abbreviations for rule names to be used in proofs are given in parentheses.

*The System PL*

**Hypothesis**

| | |
|---|---|
| \|_ A | A new hypothesis A may be added to a proof at any time, as long as A begins a new subproof. |

**Modus Ponens**

| | |
|---|---|
| A<br>$A\rightarrow B$<br>------<br>B (MP) | This is the familiar rule Modus Ponens. It is understood that A, $A\rightarrow B$, and B must all lie in *exactly the same subproof*. |

**Conditional Proof**

| | |
|---|---|
| \|_ A<br>\| :<br>\| B<br>-------<br>$A\rightarrow B$ (CP) | When a proof of B is derived from the hypothesis A, it follows that $A\rightarrow B$, where $A\rightarrow B$ lies outside hypothesis A. |

**Double Negation**

| | |
|---|---|
| ~~ A<br>--------<br>A (DN) | The rule allows the removal of double negations. As with (MP), ~~A and A must lie in the same subproof. |

**Reiteration**

| | |
|---|---|
| A<br>:<br>\|_ B<br>\| :<br>\|-----<br>\| A (Reit) | Sentence A may be copied into a new subproof. (In this case, into the subproof headed by B.) |

These five rules comprise a system for propositional logic called PL. The rules say that if you have proven what appears above the dotted line, then you may write down what appears below the dotted line. Note that in applying (MP) and (DN), *all sentences involved must lie in the same subproof.* Here is a sample proof of the argument p→q, ~q / ~p, to illustrate how we present proofs in this book.

```
| p→q
|_ ~q
| q→⊥       (Def~)
| |_ p
| | p→q     (Reit)
| | q       (MP)
| | q→⊥     (Reit)
| | ⊥       (MP)
| p→⊥       (CP)
| ~p        (Def~)
```

The proof begins by placing the premises of the argument (namely, p→q and ~q) at the head of the outermost subproof. Then the conclusion (~p) is derived from these using the five rules of PL. Since there are no rules concerning the negation sign, it is necessary to use (Def~) to convert all occurrences of ~ into → and ⊥ as we have done in the third and last steps. We do not bother writing the name (Hyp) where we have used the hypothesis rule. That the (Hyp) rule is being used is already clear from the presence of the subproof bracket (the horizontal "diving board" at the head of a subproof).

Most books use line numbers in the justification of steps of a proof. Since we only have four rules, the use of line numbers is really not necessary. For example, when (CP) is used, the steps at issue must be the beginning and end of the preceding subproof; when (DN) is used to produce A, it is easy to locate the sentence ~~A to which it was applied; when (MP) is used to produce B, it is easy enough to find the steps A and A→B to which (MP) was applied. On occasion, we will number steps to highlight some parts of a proof under discussion, but step numbers will not be part of the official notation of proofs, and they are not required in the solutions to proof exercises.

Proofs in PL generally require many uses of Reiteration (Reit). That is because (MP) cannot be applied to A and A→B unless both of these sentences lie in the same subproof. This constant use of (Reit) is annoying, especially in longer proofs, so we will adopt a convention to leave out the (Reit) steps where it is clear that an official proof could be constructed by adding them back in. According to this more relaxed policy, the proof just given may be abbreviated as follows:

$$
\begin{array}{lll}
\mid & p\to q & \\
\mid & \underline{\sim q} & \\
\mid & q\to\bot & (\text{Def}{\sim}) \\
\mid & \mid\ \underline{p} & \\
\mid & \mid\ q & (\text{MP}) \\
\mid & \mid\ \bot & (\text{MP}) \\
\mid & p\to\bot & (\text{CP}) \\
\mid & \sim p & (\text{Def}{\sim})
\end{array}
$$

We will say that an argument H / C is provable in PL (in symbols: H $\vdash_{PL}$ C) exactly when it is possible to fill in a subproof headed by members of H to obtain C.

$$
\begin{array}{ll}
\mid & \underline{H} \\
\mid & \vdots \\
\mid & C
\end{array}
$$

It is possible to prove some sentences outside of any subproof. These sentences are called *theorems.* Here, for example, is a proof that p→(q→p) is a theorem.

$$
\begin{array}{lll}
\mid & \underline{p} & \\
\mid & \mid\ \underline{q} & \\
\mid & \mid\ p & (\text{Reit}) \\
\mid & q\to p & (\text{CP}) \\
\multicolumn{2}{l}{p\to(q\to p)} & (\text{CP})
\end{array}
$$

**EXERCISE 1.2** Prove the following in PL.

a) p→q / (q→⊥)→(p→⊥)
b) p→q, p→(q→⊥) / p→⊥
c) Show (p→q)→(~q→~p) is a theorem of PL.

## 1.3. Derivable Rules of PL

PL is a complete system for propositional logic. Every valid argument written in the language of propositional logic has a proof in PL. However, proofs involving the abbreviations ~, &, ∨, and ↔ may be very complicated. The task of proof finding is immensely simplified by introducing derivable rules to govern the behavior of the defined connectives. (A rule is derivable in a system iff it can be proven in the system.) It is easy to show that the rule Indirect Proof (IP) is derivable in PL. Once this is established, we may use (IP) in the future, with the understanding that it abbreviates a sequence of steps using the original rules of PL.

Proof of Derivability:

$$
\begin{array}{llll}
\begin{array}{|l} \underline{\ \sim A} \\ \ \ \vdots \\ \ \ \bot \end{array} & & \begin{array}{|l} \underline{\ \sim A} \\ \ \ \vdots \\ \ \ \bot \end{array} & \\
\text{------} & & \text{-------} & \\
A & (\text{IP}) & \sim A \rightarrow \bot & (\text{CP}) \\
 & & \sim\sim A & (\text{Def}\sim) \\
 & & A & (\text{DN})
\end{array}
$$

The (IP) rule has been stated at the left, and to the right we have indicated how the same result can be obtained using only the original rules of PL. Instead of using (IP) to obtain A, (CP) is used to obtain ~A→⊥. This by (Def~) is really ~~A, from which we obtain A by (DN). So whenever we use (IP), the same result can be obtained by the use of these three steps instead. It follows that adding (IP) to PL cannot change what is provable.

We may also show derivable a rule (⊥In) that says that ⊥ follows from a contradictory pair of sentences A, ~A.

Proof of Derivability:

$$
\begin{array}{llll}
A & & A & \\
\sim A & & \sim A & \\
\text{-----} & & \text{-----} & \\
\bot & (\bot\text{In}) & A \rightarrow \bot & (\text{Def}\sim) \\
 & & \bot & (\text{MP})
\end{array}
$$

Once (IP) and (⊥In) are available, two more variations on the rule of Indirect Proof may be shown derivable.

Proof of Derivability:

```
|_ ~A            |_ ~A
|   :            |   :
|   B            |   B
|   :            |   :
|  ~B            |  ~B
-------          |--------
A      (~Out)    | ⊥      (⊥In)
                 A        (IP)
```

> **EXERCISE 1.3** Show that the following variant of Indirect Proof is also derivable. (Feel free to appeal to (⊥In) and (IP), since they were previously shown derivable.)
>
> ```
> |_ A
> |  :
> |  B
> |  :
> | ~B
> -------
> ~A    (~In)
> ```

With (~Out) available it is easy to show the derivability of (~~In), a variant of Double Negation.

Proof of Derivability:

```
A               A
----            ----
~~A (~~In)      |_~~~A
                |  ~A     (DN)
                |   A     (Reit)
                ~~A       (~Out)
```

Now it is easy to prove the rule of Contradiction (Contra), which says that from a contradiction anything follows:

Proof of Derivability:

```
⊥                 ⊥
----              ----
A   (Contra)      |_ ~A
                  |  ⊥      (Reit)
                  A         (IP)
```

It is possible to show that the standard natural deduction rules for the propositional connectives &, v, and ↔ are also derivable.

```
A                 A&B           A&B
B                 -----         -----
-----             A (&Out)      B (&Out)
A&B (&In)
```

```
A                 B                AvB
-----             -----            |_ A
AvB (vIn)         AvB (vIn)        |  :
                                   |  C

                                   |_ B
                                   |  :
                                   |  C
                                   -----
                                   C (vOut)
```

```
A→B                        A↔B                A↔B
B→A                        ---------          --------
--------                   A→B (↔Out)         B→A (↔Out)
A↔B (↔In)
```

(It is understood that all steps in these derivable rules must lie in the same subproof.) The hardest demonstrations of derivability concern (&Out)

and (∨Out). Here are derivations for (∨Out) and (one half of) (&Out) to serve as models for proofs of this kind. You will show derivability of the other rules in the next exercise.

```
(∨Out)
A∨B
|_A
|  :
| C

|_B
| :
| C
--------
A→C          (CP)  (first subproof)
B→C          (CP)  (second subproof)
~A→B         (Def∨) (first line)
|_ ~C
| |_ A
| |  C       (MP)
| | ~C       (Reit)
|  ~A        (~In)
|  B         (MP)
|  C         (MP)
|  ⊥         (⊥In)
C            (IP)

(&Out)
A&B
---------
~(A→~B)      (Def&)
|_ ~B
| |_ A
| |  ~B      (Reit)
|  A→~B      (CP)
| ~(A→~B)    (Reit)
B            (~Out)
```

**EXERCISE 1.4** Show that (&In), the other half of (&Out), (vIn), (↔In), and (↔Out) are all derivable. You may use rules already shown to be derivable ((~Out) and (~In) are particularly useful), and you may abbreviate proofs by omitting (Reit) steps wherever you like. (Hint for (&Out). Study the proof above. If you still have a problem, see the discussion of a similar proof below.)

The following familiar derivable rules: Modus Tollens (MT), Contraposition (CN), De Morgan's Law (DM), and (→F) may also come in handy during proof construction. (Again it is assumed that all sentences displayed in these rules appear in the same subproof.) Showing they are derivable in PL provides excellent practice with the system PL.

| | |
|---|---|
| A→B<br>~B<br>--------<br>~A (MT) | A→B<br>--------<br>~B→~A (CN) |
| ~(AvB)<br>------------<br>~A&~B (DM) | ~(A&B)<br>-----------<br>~Av~B (DM) |
| ~(A→B)<br>-----------<br>A (→F) | ~(A→B)<br>-----------<br>~B (→F) |

To illustrate the strategies for showing these are derivable rules, the proof for (→F) will be worked out in detail here. (It is similar to the proof for (&Out).) We are asked to start with ~(A→B) and obtain a proof of A. The only strategy that has any hope at all is to use (~Out) to obtain A. To do that, assume ~A and try to derive a contradiction.

```
1.  ~(A→B)
    ------------
2.  |_ ~A
    |  ????
    A            (~Out)
```

The problem is to figure out what contradiction to try to prove to complete the subproof headed by ~A. There is a simple principle to help guide the solution. When choosing a contradiction, watch for sentences containing ~ that have already become available. Both ~A and ~(A→B) qualify, but there is a good reason not to attempt a proof of the contradiction A and ~A. The reason is that doing so would put us in the position of trying to find a proof of A all over again, which is what we were trying to do in the first place. In general, it is best to choose a sentence different from the hypothesis for a (~In) or (~Out). So the best choice of a contradiction will be ~(A→B) and its opposite A→B.

```
1.  ~(A→B)
    ------------
2.  |_ ~A
    |   ????
    |  A→B
    |  ~(A→B)   1 (Reit)
    A           (~Out)
```

The remaining problem is to provide a proof of A→B. Since (CP) is the best strategy for building a sentence of this shape, the subproof necessary for (CP) is constructed.

```
1.  ~(A→B)
    ------------
2.  |_ ~A
    |   |_ A
    |   |  ????
    |   |  B
    |  A→B      (CP)
    |  ~(A→B)   1 (Reit)
    A           (~Out)
```

At this point the proof looks near hopeless. However, that is simply a sign that (~Out) is needed again, this time to prove B. So a new subproof headed by ~B is constructed with the hope that a contradiction can be proven there. Luckily, both A and ~A are available, which solves the problem.

| | | |
|---|---|---|
| 1. | $\sim(A \to B)$ | |
| | -------- | |
| 2. | $\sim A$ | |
| 3. | $A$ | |
| 4. | $\sim B$ | |
| 5. | $A$ | 3 (Reit) |
| 6. | $\sim A$ | 2 (Reit) |
| 7. | $B$ | 5 6 (~Out) |
| 8. | $A \to B$ | 7 (CP) |
| 9. | $\sim(A \to B)$ | 1 (Reit) |
| 10. | $A$ | 8 9 (~Out) |

> **EXERCISE 1.5.** Show that (MT), (CN), (DM), and the second version of ($\to$F) are derivable rules of PL.

In the rest of this book we will make use of these derivable rules without further comment. Remember, however, that our official system PL for propositional logic contains only the symbols $\to$ and $\perp$, and the rules (Hyp), (MP), (CP), (Reit), and (DN). Given the present collection of derivable rules, constructing proofs in PL is a fairly straightforward matter.

Proofs involving $\vee$ tend to be difficult. However, they are often significantly easier if ($\vee$Out) can be used in the appropriate way. Let us illustrate by proving $p \vee q$ / $q \vee p$. We make $p \vee q$ a hypothesis and hope to derive $q \vee p$.

$p \vee q$
:
$q \vee p$

At this point many students will attempt to prove either p or q, and obtain the last step by ($\vee$In). This is a poor strategy. As a matter of fact, it is impossible to prove either p or q from the available hypothesis $p \vee q$. When faced with a goal of the form $A \vee B$, it is a bad idea to assume the goal comes from ($\vee$In), unless it is obvious how to prove A or B. Often when the goal has the shape $A \vee B$, one of the available lines is also a disjunction. When this happens, it is always a good strategy to assume that the goal comes from (∨Out). In our example, we have p∨q, so we will use this step to get our goal q∨p using (∨Out).

```
|_ p∨q
|  :
|  q∨p   (∨Out)
```

If q∨p follows from p∨q by (∨Out), we will need to complete two subproofs, one headed by p and ending with q∨p and the other headed by q and ending with q∨p.

```
|_ p∨q
|   |_ p
|   |  :
|   |  q∨p
|
|   |_ q
|   |  :
|   |  q∨p
|  q∨p        (∨Out)
```

Now all we need to do is complete each subproof, and the goal q∨p will be proven by (∨Out). This is easily done using (∨In).

```
|_ p∨q
|   |_ p
|   |  q∨p   (∨In)
|
|   |_ q
|   |  q∨p   (∨In)
|  q∨p        (∨Out)
```

In order to save paper, and to see the structure of the (∨Out) process more clearly, I suggest that you put the two subproofs that are introduced by the (∨Out) rule side by side:

```
|_ p∨q
|   |_ p               |_ q
|   |  q∨p   (∨In)     |  q∨p   (∨In)
|  q∨p        (∨Out)
```

This way of notating proofs will play an important role in showing parallels between proofs and the truth tree method in Section 7.1.

> **EXERCISE 1.6** Prove the following using the (vOut) strategy just described. Place the paired subproofs introduced by (vOut) side by side to save space.
>
> a) pvq, p→r, q→s / rvs
> b) pv(q&r) / (pvq)&(pvr)
> c) ~pv~q / ~(p&q)
> d) pv(qvr) / (pvq)vr

## 1.4. Natural Deduction Rules for System K

Natural deduction rules for the operator □ can be given that are economical and easy to use. The basic idea behind these rules is to introduce a new kind of subproof, called a boxed subproof. A *boxed subproof* is a subproof headed by □ instead of a sentence:

```
|_ □
|
|
```

One way to interpret a boxed subproof is to imagine that it prefixes each sentence it contains with □. For example, suppose A is proven in a subproof headed by □:

```
|_ □
|  :
| A
```

This means that □A has been proven outside that subproof.

Given this understanding of boxed subproofs, the following (□Out) and (□In) rules seem appropriate.

```
□A                    |_ □
:                     |  :
|_ □                  | A
|-------              --------
|  A     (□Out)       □A      (□In)
```

The (□Out) rule says that when we have proven □A, we may put A in a boxed subproof (which indicates that A prefixed by a □ is proven). The (□In) rule says that once we have proven A in a boxed subproof (indicating that A prefixed by □ is proven), it follows that □A is proven outside that subproof. (□Out) and (□In) together with natural deduction rules for PL comprise the system K.

**System K = PL + (□Out)+ (□In).**

There is an important difference between boxed and ordinary subproofs when it comes to the use of (Reit). (Reit) allows us to copy a sentence into the next-deepest subproof, provided the subproof is headed by a *sentence* B.

```
A
|_ B
|  :
|-----
|  A   (Reit)
```

But the (Reit) rule does *not* allow A to be copied into a boxed subproof:

```
A
|_ □
|  :
|-----
|  A   (INCORRECT USE OF (REIT))
```

This is incorrect because it amounts to reasoning from A to □A, which is clearly fallacious. (If A is so, it doesn't follow that A is necessary, obligatory, etc.) So be very careful when using (Reit) *not* to copy a sentence into a boxed subproof.

Strategies for finding proofs in K are simple to state and easy to use. In order to prove a sentence of the form □A, simply construct a boxed subproof and attempt to prove A inside it. When the proof of A in that boxed subproof is complete, □A will follow by (□In). In order to use a sentence of the form □A, remove the box using (□Out) by putting A in a boxed subproof. The following proof of □p&□q ⊢ □(p&q) illustrates these strategies.

| | | | |
|---|---|---|---|
| 1. | □p&□q | | |
| 2. | □p | (&Out) | [1] |
| 3. | □q | (&Out) | [2] |
| 4. | □ | | [4] |
| 5. | p | (□Out) | [6] |
| 6. | q | (□Out) | [7] |
| 7. | p&q | (&In) | [5] |
| 8. | □(p&q) | (□In) | [3] |

The numbers to the right in square brackets are discovery numbers. They indicate the order in which steps were written during the process of proof construction. Most novices attempt to construct proofs by applying rules in succession from the top of the proof to the bottom. However, the best strategy often involves working backwards from a goal. In our sample, (&Out) was applied to line 1 to obtain the conjuncts: □p and □q. It is always a good idea to apply (&Out) to available lines in this way.

| | | | |
|---|---|---|---|
| 1. | □p&□q | | |
| 2. | □p | (&Out) | [1] |
| 3. | □q | (&Out) | [2] |

Having done that, however, the best strategy for constructing this proof is to consider the conclusion: □(p&q). This sentence has the form □A. Therefore, it is a good bet that it can be produced from A (and a boxed subproof) by (□In). For this reason a boxed subproof is begun on line 4, and the goal for that subproof (p&q) is entered on line 7.

| | | | |
|---|---|---|---|
| 1. | □p&□q | | |
| 2. | □p | (&Out) | [1] |
| 3. | □q | (&Out) | [2] |
| 4. | □ | | [4] |
| 5. | | | |
| 6. | | | |
| 7. | p&q | | [5] |
| 8. | □(p&q) | (□In) | [3] |

The proof is then completed by applying (□ Out) to lines 2 and 3, from which p&q is obtained by (&In).

> **EXERCISE 1.7** Prove the following in K (derivable rules are allowed):
>
> a) □p / □(p∨q)
> b) □(p→q) / □p→□q
> c) □(p&q) / □p&□q
> d) □(p∨q), □(p→r), □(q→r) / □r
> e) □p∨□q / □(p∨q) (Hint: Set up (∨Out) first.)

## 1.5. A Derivable Rule for ◇

In most modal logics, there is a strong operator (□) and a corresponding weak one (◇). The weak operator can be defined using the strong operator and negation as follows:

(Def◇) ◇A $=_{df}$ ~□~A

(◇A may be read 'diamond A'.) Notice the similarities between (Def◇) and the quantifier principle ∀xA ↔ ~∃x~A. (We use ∀ for the universal and ∃ for the existential quantifier.) There are important parallels to be drawn between the universal quantifier ∀ and □, on the one hand, and the existential quantifier ∃ and ◇ on the other. In K and the systems based on it, □ and ◇ behave very much like ∀ and ∃, especially in their interactions with the connectives →, &, and ∨. For example, □ distributes through & both ways, that is, □(A&B) entails □A&□B and □A&□B entails □(A&B). However, □ distributes through ∨ in only one direction, □A∨□B entails □(A∨B), but *not* vice versa. This is exactly the pattern of distribution exhibited by ∀. Similarly, ◇ distributes through ∨ both ways, and through & in only one, which mimics the distribution behavior of ∃. Furthermore, the following theorems of K:

□(A → B) → (□A → □B)

and

□(A → B) → (◇A → ◇B)

parallel important theorems of quantificational logic:

$\forall x(Ax \rightarrow Bx) \rightarrow (\forall xAx \rightarrow \forall xBx)$

and

$\forall x(Ax \rightarrow Bx) \rightarrow (\exists xAx \rightarrow \exists xBx)$.

To illustrate how proofs involving $\Diamond$ are carried out, we will explain how to show $\Box(p\rightarrow q) \rightarrow (\Diamond p\rightarrow\Diamond q)$ is a theorem. The strategies used in this proof may not be obvious, so it is a good idea to explain them in detail. The conclusion is the conditional, $\Box(p\rightarrow q) \rightarrow (\Diamond p\rightarrow\Diamond q)$, so the last line will be obtained by (CP), and we need to construct a proof from $\Box(p\rightarrow q)$ to $\Diamond p\rightarrow\Diamond q$. Since the latter is also a conditional, it will be obtained by (CP) as well, so we need to fill in a subproof from $\Diamond p$ to $\Diamond q$. At this stage, the proof attempt looks like this:

| | | | |
|---|---|---|---|
| 1. | $\vert\!\underline{\ }\ \Box(p\rightarrow q)$ | | [1] |
| 2. | $\vert\ \ \vert\!\underline{\ }\ \Diamond p$ | | [3] |
| | $\vert\ \ \vert\ \ ???$ | | |
| | $\vert\ \ \vert\ \Diamond q$ | | |
| | $\vert\ \ \Diamond p\rightarrow\Diamond q$ | (CP) | [4] |
| | $\Box(p\rightarrow q)\rightarrow(\Diamond p\rightarrow\Diamond q)$ | (CP) | [2] |

Since we are left with $\Diamond q$ as a goal, and we lack any derivable rules for $\Diamond$, the only hope is to convert $\Diamond q$ (and the hypothesis $\Diamond p$) into $\Box$ using (Def$\Diamond$).

| | | | |
|---|---|---|---|
| 1. | $\vert\!\underline{\ }\ \Box(p\rightarrow q)$ | | [2] |
| 2. | $\vert\ \ \vert\!\underline{\ }\ \Diamond p$ | | [4] |
| 3. | $\vert\ \ \vert\ \sim\Box\sim p$ | (Def$\Diamond$) | [6] |
| | $\vert\ \ \vert\ \ ???$ | | |
| | $\vert\ \ \vert\ \sim\Box\sim q$ | | [7] |
| | $\vert\ \ \vert\ \Diamond q$ | (Def$\Diamond$) | [5] |
| | $\vert\ \ \Diamond p\rightarrow\Diamond q$ | (CP) | [3] |
| | $\Box(p\rightarrow q)\rightarrow(\Diamond p\rightarrow\Diamond q)$ | (CP) | [1] |

At this point there seems little hope of obtaining $\sim\Box\sim q$. In situations like this, it is a good idea to obtain your goal with (~Out) or (~In). In our case we will try (~In). So we need to start a new subproof headed by □~q and try to derive a contradiction within it.

```
1.  ⎿ □(p→q)                          [2]
2.  │  ⎿ ◇p                           [4]
3.  │  │ ~□~p           (Def◇)        [6]
4.  │  │  │⎿ □~q                      [8]
    │  │  │ ???
    │  │  │
    │  │ ~□~q           (~In)         [7]
    │  │ ◇q             (Def◇)        [5]
    │  ◇p→◇q            (CP)          [3]
   □(p→q)→(◇p→◇q)       (CP)          [1]
```

The most crucial stage in finding the proof is to find a contradiction to finish the (~In) subproof. A good strategy in locating a likely contradiction is to inventory steps of the proof already available that contain ~. Step 3 (namely, ~□~p) qualifies, and this suggests that a good plan would be to prove □~p and reiterate ~□~p to complete the subproof.

```
1.  ⎿ □(p→q)                          [2]
2.  │  ⎿ ◇p                           [4]
3.  │  │ ~□~p           (Def◇)        [6]
4.  │  │  │⎿ □~q                      [8]
    │  │  │ ???
    │  │  │ □~p                       [10]
    │  │  │ ~□~p        (Reit)        [9]
    │  │ ~□~q           (~In)         [7]
    │  │ ◇q             (Def◇)        [5]
    │  ◇p→◇q            (CP)          [3]
   □(p→q)→(◇p→◇q)       (CP)          [1]
```

At this point our goal is □~p. Since it begins with a box, (□In) seems the likely method for obtaining it, and we create a boxed subproof and enter ~p at the bottom of it as a new goal.

| | | | |
|---|---|---|---|
| 1. | □(p→q) | | [2] |
| 2. | ◇p | | [4 ] |
| 3. | ~□~p | (Def◇) | [6] |
| 4. | □~q | | [8] |
| | □ | | [11] |
| | ??? | | |
| | ~p | | [12] |
| | □~p | (□In) | [10] |
| | ~□~p | (Reit) | [9] |
| | ~□~q | (~In) | [7] |
| | ◇q | (Def◇) | [5] |
| | ◇p→◇q | (CP) | [3] |
| | □(p→q)→(◇p→◇q) | (CP) | [1] |

But now it is possible to use (□Out) (and (Reit)) to place p→q and ~q into the boxed subproof, where the goal ~p can be obtained by (MT), Modus Tollens. So the proof is complete.

| | | | |
|---|---|---|---|
| 1. | | | [2] |
| 2. | ◇p | | [4] |
| 3. | ~□~p | (Def◇) | [6] |
| 4. | □~q | | [8] |
| | □ | | [11] |
| 5. | p→q | 1 (□Out) | [13] |
| 6. | ~q | 4 (□Out) | [14] |
| 7. | ~p | 5, 6 (MT) | [12] |
| 8. | □~p | 7 (□In) | [10] |
| 9. | ~□~p | 3 (Reit) | [9] |
| 10. | ~□~q | 8, 9 (~In) | [7] |
| 11. | ◇q | 10 (Def◇) | [5] |
| 12. | ◇p→◇q | 11 (CP) | [3] |
| 13. | □(p→q)→(◇p→◇q) | 12 (CP) | [1] |

**EXERCISE 1.8** Show that the following sentences are theorems of K by proving them outside any subproofs:

a) □(p&q) ↔ (□p&□q)
b) (□pv□q) → □(pvq)
c) (◇pv◇q) ↔ ◇(pvq)
d) ◇(p&q) → (◇p&◇q)

As you can see from Exercises 1.8c–d, proofs in K can be rather complex when ◇ is involved. We have no rules governing ◇, and so the only strategy available for working with a sentence of the form ◇A is to translate it into ~□~A by (Def◇). This introduces many negation signs, which complicates the proof. To help overcome the problem, let us introduce a derivable rule called (◇Out).

```
◇A
|_ □
|  |_A
|  |  :
|  |  B
-------
◇B      (◇Out)
```

> **EXERCISE 1.9** Showthat (◇Out) is a derivable rule of the natural deduction formulation for **K.** (Hint: From the two subproofs use (CP) and then (□In) to obtain □(A→B). Now use the strategy used to prove □(p→q)→(◇p→◇q) above.)

To illustrate the use of this rule, we present a proof of problem d) of Exercise 1.8: ◇(p&q) → (◇p&◇q). Since this is a conditional, a subproof headed by ◇(p&q) is constructed in hopes of proving ◇p&◇q. This latter sentence may be obtained by (&In) provided we can find proofs of ◇p and ◇q. So the proof attempt looks like this so far:

```
|_ ◇(p&q)
|   ???
|  ◇p
|   ???
|  ◇q
|  ◇p&◇q              (&In)
◇(p&q) → (◇p&◇q)   (CP)
```

The (◇Out) rule comes in handy whenever a sentence of the shape ◇A is available, and you are hoping to prove another sentence of the same shape. Here we hope to prove ◇p, and ◇(p&q) is available. To set up the (◇Out), subproofs headed by □ and p&q must be constructed, within which p must be proven. But this is a simple matter using (&Out).

```
1.  |_ ◇(p&q)
2.  | |_ □
3.  | | |_ p&q
4.  | | |  p                 3      (&Out)
5.  | ◇p                     1, 4   (◇Out)
    | ???
    | ◇q
    | ◇p&◇q                         (&In)
   ◇(p&q) → (◇p&◇q)                 (CP)
```

Using the same strategy to obtain ◇q completes the proof.

```
1.  |_ ◇(p&q)
2.  | |_ □
3.  | | |_ p&q
4.  | | |  p                 3      (&Out)
5.  | ◇p                     1, 4   (◇Out)
6.  | |_ □
7.  | | |_ p&q
8.  | | |  q                 7      (&Out)
9.  | ◇q                     1, 8   (◇Out)
10. | ◇p&◇q                  5, 9   (&In)
11. ◇(p&q) → (◇p&◇q)         10     (CP)
```

Since we will often use (◇Out), and the double subproof in this rule is cumbersome, we will abbreviate the rule as follows:

```
◇A
|_ □, A
|  :
|  B
------
◇B          (◇Out)
```

Here the subproof with □, A at its head is shorthand for the double subproof.

```
                            |_ □
|_ □, A                     | |_ A
|   :       abbreviates:    | |  :
|  B                        | |  B
```

We will call this kind of abbreviated subproof a *world-subproof.* This abbreviation is a special case of the idea that we will adopt for arguments, namely, that a sequence of subproofs can be abbreviated by listing the hypotheses in a single subproof. For example, instead of writing

```
|_ A
| |_ B
| | |_ C
```

we may write:

```
|_ A, B, C
```

instead.

Given the world-subproof abbreviation, it should be clear that (□Out) can be applied to a boxed sentence □A to place A into a world-subproof directly below where □A appears. Using world-subproofs, we may rewrite the last proof in a more compact format.

```
|_ ◇(p&q)
|   |_ □, p&q
|   |  p                  (&Out)
|   ◇p                    (◇Out)
|   |_ □, p&q
|   |  q                  (&Out)
|   ◇q                    (◇Out)
|   ◇p&◇q                 (&In)
◇(p&q) → (◇p&◇q)          (CP)
```

**EXERCISE 1.10**

a) Redo Exercise 1.8c using (◇Out) with world-subproofs.
b) Show that the following useful rules are derivable in K:

```
~□A                      ~◇A
-------                  -------
◇~A   (~□)               □~A   (~◇)
```

c) Using the rules (~□) and (~◇) and other derivable rules if you like, prove □~□p / □◇~p and ◇~◇p / ◇□~p.

## 1.6. Horizontal Notation for Natural Deduction Rules

Natural deduction rules and proofs are easy to use, but presenting them is sometimes cumbersome since it requires the display of vertical subproofs. Let us develop a more convenient notation. When sentence A is proven under the following hypotheses:

$$
\begin{array}{l}
|\underline{\ \ }\ B \\
|\ |\underline{\ \ }\ \Box \\
|\ |\ |\underline{\ \ }\ C \\
|\ |\ |\ |\underline{\ \ }\ D \\
|\ |\ |\ |\ |\underline{\ \ }\ \Box \\
|\ |\ |\ |\ |\ \ \vdots \\
|\ |\ |\ |\ |\ \ A
\end{array}
$$

we may first abbreviate it as follows:

$$
\begin{array}{l}
|\underline{\ \ }\ B, \Box, C, D, \ \Box \\
|\ \ \vdots \\
|\ \ A
\end{array}
$$

This can in turn be expressed in what we will call horizontal notation as follows:

B, $\Box$, C, D, $\Box \vdash$ A

Notice that B, $\Box$, C, D, $\Box$ is a list of the hypotheses (in order) under which A lies, so we can think of B, $\Box$, C, D, $\Box$ / A as a kind of argument. Of course, $\Box$ is not strictly speaking a hypothesis, since hypotheses are *sentences,* but we will treat $\Box$ as an honorary hypothesis nevertheless, to simplify our discussion. When we write 'B, $\Box$, C, D, $\Box \vdash$ A', we mean that there is a proof of A under the hypotheses B, $\Box$, C, D, $\Box$, in that order. We will use the letter 'L' to indicate such lists of the hypotheses, and we will write 'L $\vdash$ A' to indicate that A is provable given the list L. Notice that L is a list; *the order of the hypotheses matters.* Given this new notation, the rules of K may be reformulated in horizontal notation. To illustrate, consider Conditional Proof.

$$
\begin{array}{l}
|\underline{\ \ }\ A \\
|\ \ \vdots \\
|\ \ B \\
\text{-----------} \\
A\rightarrow B \quad \text{(CP)}
\end{array}
$$

This rule may be applied in any subproof, so let L be a list of all the hypotheses under which A→B lies in the use of this rule. Then the conclusion of this rule may be expressed in horizontal notation as L ⊢ A→B. To indicate the portion of the rule above the dotted line we consider each sentence that is not a hypothesis. In this case, the only such sentence is B. Now B lies under the hypothesis A, and all hypotheses L under which A→B lies. So the horizontal notation for this line is L, A ⊢ B. Putting the two results together, the horizontal notation for the rule (CP) is the following:

L, A ⊢ B<br>
--------------<br>
L ⊢ A → B

In similar fashion, all the rules of K can be written in horizontal notation. A complete list follows for future reference.

**Horizontal Formulation of the Rules of K**

Hypothesis L, A ⊢ A (Hyp)

Reiteration L ⊢ A<br>
----------<br>
L, B ⊢ A (Reit)<br>
(Note that B in the conclusion is the head of the subproof into which A is moved.)

Modus Ponens L ⊢ A<br>
L ⊢ A→B<br>
-------<br>
L ⊢ B (MP)

Conditional Proof L, A ⊢ B<br>
-------<br>
L ⊢ A→B (CP)

Double Negation L ⊢ ~~A<br>
-------<br>
L ⊢ A (DN)

□In L, □ ⊢ A<br>
----------<br>
L ⊢ □A (□In)

| | |
|---|---|
| $\Box$Out | $L \vdash \Box A$ |
| | ---------- |
| | $L, \Box \vdash A$     ($\Box$Out) |

> **EXERCISE 1.11** Express (&Out), (IP), and ($\Diamond$Out) in horizontal notation.

Instead of presenting proofs in subproof notation, we could also write them out in horizontal notation instead. For each line A of the proof, one constructs the list L of all hypotheses under which A lies, and then writes $L \vdash A$. In the case of a hypothesis line A, the sentence A is understood to lie under **itself** as a hypothesis, so the horizontal notation for a hypothesis always has the form $L, A \vdash A$. When several sentences head a subproof, like this:

```
| p
| q
|_ r
```

it is understood that this abbreviates three separate subproofs, one for each sentence. Therefore, the horizontal notation for these steps is given below:

| | |
|---|---|
| \|_ p | $p \vdash p$ |
| \| \|_ q | $p, q \vdash q$ |
| \| \| \|_ r | $p, q, r \vdash r$ |

For example, here is a proof written in subproof form on the left with the horizontal version to the right.

| | | | |
|---|---|---|---|
| \|_ $\Box p \& \Box q$ | | $\Box p \& \Box q \vdash \Box p \& \Box q$ | (Hyp) |
| \| $\Box p$ | (&Out) | $\Box p \& \Box q \vdash \Box p$ | (&Out) |
| \| $\Box q$ | (&Out) | $\Box p \& \Box q \vdash \Box q$ | (&Out) |
| \| \|_ $\Box$ | | | |
| \| \| p | ($\Box$Out) | $\Box p \& \Box q, \Box \vdash p$ | ($\Box$Out) |
| \| \| q | ($\Box$Out) | $\Box p \& \Box q, \Box \vdash q$ | ($\Box$Out) |
| \| \| p&q | (&In) | $\Box p \& \Box q, \Box \vdash p \& q$ | (&In) |
| \| $\Box$(p&q) | ($\Box$In) | $\Box p \& \Box q \vdash \Box(p \& q)$ | ($\Box$In) |

> **EXERCISE 1.12** Convert solutions to Exercises 1.7c–d into horizontal notation.

When proofs are viewed in horizontal notation, it becomes apparent that the rules of K apply to *arguments* L / C. In all proofs, the (Hyp) rule first introduces arguments of the form L, A ⊢ A (where L is empty in the first step), and then rules are applied to these arguments over and over again to create new provable arguments out of old ones. You are probably more familiar with the idea that rules of logic apply to sentences, not arguments. However, the use of subproof notation involves us in this more general way of looking at how rules work.

## 1.7. Necessitation and Distribution

There are many alternative ways to formulate the system K. Using boxed subproofs is quite convenient, but this method had not been invented when the first systems for modal logic were constructed. In the remainder of this chapter, two systems will be presented that are *equivalent* to K, which means that they agree with K exactly on which arguments are provable. The traditional way to formulate a system with the effect of K is to add to propositional logic a rule called Necessitation (Nec) and an axiom called Distribution (Dist). We will call this system TK, for the *t*raditional formulation of K.

**System TK = PL + (Nec) + (Dist).**

⊢ A $\qquad\qquad$ ⊢ □(A→B)→(□A→□B) (Dist)

-------

⊢ □A (Nec)

The rule of Necessitation may appear to be incorrect. It is wrong, for example, to conclude that grass is *necessarily* green (□A) given that grass is green (A). This objection, however, misinterprets the content of the rule. The notation '⊢ A' above the dotted line indicates that sentence A is a *theorem,* that is, it has been proven *without the use of any hypotheses.* The rule does not claim that □A follows from A, but rather that □A follows when A is a *theorem.* This is quite reasonable. There is little reason to object to the view that the theorems of logic are necessary.

The derivation of a sentence within a subproof does not show it to be a theorem. So Necessitation does not apply within a subproof. For example, it is incorrectly used in the following "proof":

```
1. |_ p
2. |  □p        1 (Nec)  INCORRECT USE!  (Line 1 is in a subproof.)
3.  p→□p        (CP)
```

We surely do not want to prove the sentence p→□p, which says that if something is so, it is so necessarily. The next proof illustrates a correct use of (Nec) to generate an acceptable theorem.

```
1. |_ p
2. |  p∨q            (∨In)
3. p→(p∨q)           (CP)
4. □(p→(p∨q))  3 (Nec)   CORRECT USE (Line 3 is not in
                         a subproof.)
```

It is easy enough to show that (Nec) and (Dist) are already available in K. To show that whatever is provable using (Nec) can be derived in K, assume that ⊢ A, that is, there is a proof of A outside of all hypotheses:

```
:
A
```

For example, suppose A is the theorem p→(p∨q), which is provable as follows:

```
|_ p
|  p∨q      (∨In)
p→(p∨q)     (CP)
```

The steps of this proof may all be copied inside a boxed subproof, and (□In) applied at the last step.

```
|_ □
|  :
|  A
□A   (□In)
```

The result is a proof of $\Box A$ outside all hypotheses, and so we obtain $\vdash \Box A$. In the case of our example, it would look like this:

```
|_ □
|  |_ p
|  | p∨q          (∨In)
|  p→(p∨q)        (CP)
□(p→(p∨q))        (□In)
```

To show that (Dist) is also derivable, we simply prove it under no hypotheses as follows:

```
|_ □(A→B)
|  |_ □A
|  |  |_ □
|  |  |  A                (□Out)
|  |  |  A→B              (□Out)
|  |  |  B                (MP)
|  |  □B                  (□In)
|  □A→□B                  (CP)
□(A→B)→(□A→□B)            (CP)
```

## 1.8. General Necessitation

K can also be formulated by adding to PL a single rule called General Necessitation (GN). Let H be a list of *sentences,* and let $\Box H$ be the list that results from prefixing $\Box$ to each sentence in H. So, for example, if H is the list p, q, r, then $\Box H$ is $\Box p, \Box q, \Box r$.

$H \vdash A$
-------------
$\Box H \vdash \Box A$ (GN)

The premise of General Necessitation (GN) indicates that A has a proof from H. The rule says that once such an argument is proven, then there is also a proof of the result of prefixing $\Box$ to the hypotheses and the conclusion.

General Necessitation can be used to simplify proofs that would otherwise be fairly lengthy. For example, we proved □p, □q ⊢ □(p&q) above in eight steps (Section 1.4). Using (GN), we can give a much shorter proof, using horizontal notation. Simply begin with p, q ⊢ p&q (which is provable by (Hyp) and (&In)), and apply (GN) to obtain the result.

| | |
|---|---|
| p ⊢ p | (Hyp) |
| p, q ⊢ p | (Reit) |
| p, q ⊢ q | (Hyp) |
| p, q ⊢ p&q | (&In) |
| □p, □q ⊢ □(p&q) | (GN) |

**EXERCISE 1.13.** Produce "instant" proofs of the following arguments using (GN).

a) □p ⊢ □(pvq)
b) □p, □(p→q) ⊢ □q
c) □(pvq), □(p→r), □(q→r) ⊢ □r
d) □~p, □(pvq) ⊢ □q
e) □(p→q), □~q ⊢ □~p

Now let us prove that (GN) is derivable in PL + (Nec) + (Dist). Since we showed that (Nec) and (Dist) are derivable in K, it will follow that (GN) is derivable in K. First we show that the following rule (□MP) is derivable in any propositional logic that contains Distribution.

H ⊢ □(A→B)
-------------------
H, □A ⊢ □B    (□MP)

The proof is as follows:

| | |
|---|---|
| H ⊢ □(A→B) | Given |
| H, □A ⊢ □(A→B) | (Reit) |
| ⊢ □(A→B)→(□A→□B) | (Dist) |
| H, □A ⊢ □(A→B)→(□A→□B) | (Reit) (many times) |
| H, □A ⊢ □A→□B | (MP) |
| H, □A ⊢ □A | (Hyp) |
| H, □A ⊢ □B | (MP) |

To show that (GN) is derivable, we must show that if $H \vdash A$, then $\Box H \vdash \Box A$ for any list of sentences H. This can be shown by cases depending on the length of H. It should be clear that (GN) holds when H is empty, because in that case, (GN) is just (Nec). Now suppose that H contains exactly one sentence B. Then the proof proceeds as follows:

| | |
|---|---|
| $B \vdash A$ | Given |
| $\vdash B \rightarrow A$ | (CP) |
| $\vdash \Box(B \rightarrow A)$ | (Nec) |
| $\Box B \vdash \Box A$ | ($\Box$MP) |

In case H contains two members $B_1$ and $B_2$, the proof is as follows:

| | |
|---|---|
| $B_1, B_2 \vdash A$ | Given |
| $\vdash B_1 \rightarrow (B_2 \rightarrow A)$ | (CP) (two times) |
| $\vdash \Box(B_1 \rightarrow (B_2 \rightarrow A))$ | (Nec) |
| $\Box B_1, \Box B_2 \vdash \Box A$ | ($\Box$MP) (two times) |

**EXERCISE 1.14.** Now carry out the same reasoning in case H contains three members $B_1$, $B_2$, and $B_3$.

(GN) can be shown in general when H is an arbitrary list $B_1, \ldots, B_i$ using the same pattern of reasoning.

| | |
|---|---|
| $B_1, \ldots, B_i \vdash A$ | Given |
| $\vdash B_1 \rightarrow \ldots (B_i \rightarrow A)$ | (CP) (i times) |
| $\vdash \Box(B_1 \rightarrow \ldots (B_i \rightarrow A))$ | (Nec) |
| $\Box B_1, \ldots, \Box B_i \vdash \Box A$ | ($\Box$MP) (i times) |

This completes the proof that (GN) is derivable in K. It follows that anything provable in PL + (GN) has a proof in K. In Section 9.4 it will be shown that whatever is provable in K is provable in PL + (GN). So PL + (GN) and K are simply two different ways to formulate the same notion of provability.

## 1.9. Summary of the Rules of K

*Rules of PL*

**Hypothesis**

```
|_ A   (Hyp)
```

**Reiteration**

```
A
:
|_ B
|   :
|-----
|  A   (Reit)
```

**Modus Ponens**

```
A
A→B
--------
B   (MP)
```

**Conditional Proof**

```
|_ A
|   :
|  B
-------
A→B   (CP)
```

**Double Negation**

```
~~A
---------
A   (DN)
```

*Derivable Rules of PL*

```
|_ ~A
|   :
|   ⊥
------
A    (IP)
```

```
A
~A
-----
⊥   (⊥In)
```

```
A
-----
~~A   (~~In)
```

```
|_ ~A
|   :
|   B
|   :
|  ~B
-------
A   (~Out)
```

```
|_ A
|   :
|   B
|   :
|  ~B
--------
~A   (~In)
```

```
A
B
-----
A&B   (&In)
```

```
A&B
------
A   (&Out)
```

```
A&B
-----
B   (&Out)
```

```
A              B              AvB
-----          -----          |_ A   |_B
AvB  (vIn)     AvB  (vIn)     |  :   |  :
                              |  C   |  C
                              ------
                              C   (vOut)

A→B                           A↔B                A↔B
B→A                           -----              -----
-----                         A→B   (↔Out)       B→A  (↔Out)
A↔B   (↔In)

A→B                           A→B
~B                            --------
-----                         ~B→~A   (CN)
~A   (MT)

~(AvB)                        ~(A&B)
-------                       --------
~A&~B   (DM)                  ~Av~B   (DM)

~(A→B)                        ~(A→B)
--------                      --------
A   (→F)                      ~B   (→F)
```

*K=PL+(□Out)+(□In)*

```
□A             |_ □
:              |  :
|_ □           |  A
|------        ---------
|  A   (□Out)  □A   (□In)
```

*Derivable Rules of K*

$\Diamond A$
| $\Box$, $A$
| :
| $B$
------
$\Diamond B$ ($\Diamond$Out)

$\sim\Box A$
-----
$\Diamond\sim A$ ($\sim\Box$)

$\sim\Diamond A$
-----
$\Box\sim A$ ($\sim\Diamond$)

**General Necessitation**

$H \vdash A$
--------
$\Box H \vdash \Box A$ (GN)

*The Traditional Formulation of K: tK=PL+(Nec)+(Dist)*

**Necessitation**

$\vdash A$
------
$\vdash \Box A$ (Nec)

**Distribution**

$\vdash \Box(A\rightarrow B)\rightarrow(\Box A\rightarrow\Box B)$ (Dist)

# 2

# Extensions of K

## 2.1. Modal or Alethic Logic

A whole series of interesting logics can be built by adding axioms to the basic system K. Logics for necessity and possibility were the first systems to be developed in the modal family. These modal (or alethic) logics are distinguished from the others in the modal family by the presence of the axiom (M). (M stands for 'modal'.)

(M) $\Box A \rightarrow A$

(M) claims that whatever is necessary is the case. Notice that (M) would be incorrect for the other operators we have discussed. For example, (M) is clearly incorrect when $\Box$ is read 'John believes', or 'it was the case that' (although it would be acceptable for 'John knows that'). The basic modal logic M is constructed by adding the axiom (M) to K. (Some authors call this system T.) Notice that this book uses uppercase letters, for example: 'M' for systems of logic, and uses the same letter in parentheses: '(M)' for their characteristic axioms. Adding an axiom to K means that instances of the axiom may be placed within any subproof, including boxed subproofs. For example, here is a simple proof of the argument $\Box\Box p$ / p in the system M.

| | | |
|---|---|---|
| 1. | $\Box\Box p$ | |
| 2. | $\Box\Box p \rightarrow \Box p$ | (M) |
| 3. | $\Box p$ | 1, 2 (MP) |
| 4. | $\Box p \rightarrow p$ | (M) |
| 5. | p | 3, 4 (MP) |

Line 2: $\Box\Box p\to\Box p$ counts as an instance of (M) because it has the shape $\Box A\to A$. (Just let A be $\Box p$.) Proof strategies in M often require using complex instances of (M) in this way.

Any interesting use of an axiom like (M) pretty much requires the use of (MP) in the next step. This can make proofs cumbersome. To make proofs in M shorter, it is useful to introduce the following derivable rule, which we will also call (M).

$\Box$A
------
A (M)

With this rule in hand, the proof of $\Box\Box p$ / p is simplified.

| | | |
|---|---|---|
| 1. | $\Box\Box p$ | |
| 2. | $\Box p$ | 1 (M) |
| 3. | p | 2 (M) |

In the future, whenever axioms of the form $A\to B$ are introduced, it will be understood that a corresponding derived rule of the form A / B with the same name is available.

**EXERCISE 2.1**

a) Prove $A\to\Diamond A$ in M. (Hint: Use the following instance of (M): $\Box{\sim}A\to{\sim}A$.)
b) Prove $\Box A\to\Diamond A$ in M.
c) Prove (M): $\Box A\to A$ in K plus $A\to\Diamond A$.

The rule (M) allows one to drop a $\Box$ from a formula whenever it is the main connective. You might think of this as an elimination rule for $\Box$. Exercise 2.1c shows that the system M may be formulated equivalently using $A\to\Diamond A$ in place of (M), or by adopting a $\Diamond$ introduction rule that allows one to prefix any proven formula with a $\Diamond$. This corresponds to the intuition that A must be possible if it is true.

Many logicians believe that M is too weak, and that further principles must be added to govern the *iteration,* or repetition, of modal operators. Here are three well-known iteration axioms with their names.

(4) $\Box A\to\Box\Box A$
(B) $A\to\Box\Diamond A$
(5) $\Diamond A\to\Box\Diamond A$

> **EXERCISE 2.2** Write an essay giving your reasons for either accepting or rejecting each of (4), (B), and (5).

To illustrate the use of these axioms (and their corresponding rules), here are some sample proofs that appeal to them. Here is a proof of $\Box p$ / $\Box\Box\Box p$ that uses (4).

| | | |
|---|---|---|
| 1. | $\Box p$ | |
| 2. | $\Box p \rightarrow \Box\Box p$ | (4) |
| 3. | $\Box\Box p$ | 1, 2 (MP) |
| 4. | $\Box\Box p \rightarrow \Box\Box\Box p$ | (4) |
| 5. | $\Box\Box\Box p$ | 3, 4 (MP) |

Using the derived rule (4), the proof can be shortened.

| | | |
|---|---|---|
| 1. | $\Box p$ | |
| 2. | $\Box\Box p$ | 1 (4) |
| 3. | $\Box\Box\Box p$ | 2 (4) |

Next we illustrate a somewhat more complex proof that uses (B) to prove the argument p / $\Box\Diamond\Diamond p$.

| | | |
|---|---|---|
| 1. | $p$ | |
| 2. | $\Diamond p$ | Solution to Exercise 2.1a |
| 3. | $\Box\Diamond\Diamond p$ | 2 (B) |

Note that we have taken advantage of the solution to Exercise 2.1a to save many steps in this proof. Feel free to do likewise in coming exercises. Finally, here is a proof that uses (5) to prove $\Diamond p$ / $\Box\Box\Diamond p$.

| | | |
|---|---|---|
| 1. | $\Diamond p$ | |
| 2. | $\Box\Diamond p$ | 1 (5) |
| | $\quad\Box$ | |
| 3. | $\quad\Diamond p$ | 2 ($\Box$Out) |
| 4. | $\quad\Box\Diamond p$ | 3 (5) |
| 5. | $\Box\Box\Diamond p$ | 4 ($\Box$In) |

You can see that strategies for proof finding can require more creativity when the axioms (4), (B), and (5) are available.

Although names of the modal logics are not completely standard, the system M plus (4) is commonly called S4. M plus (B) is called B (for

Brouwer's system) and M plus (5) is called S5. The following chart reviews the systems we have discussed so far.

**System M** = K + (M): $\Box A \rightarrow A$.
**System S4** = M + (4): $\Box A \rightarrow \Box\Box A$.
**System B** = M + (B): $A \rightarrow \Box\Diamond A$.
**System S5** = M + (5): $\Diamond A \rightarrow \Box\Diamond A$.

It would be more consistent to name systems after the axioms they contain. Under this proposal, S4 would be named M4 (the system M plus (4)), and S5 would be M5. This is, in fact, the common practice for naming systems that are less well known. However, the systems S4 and S5 were named by their inventor, C. I. Lewis, before systems like K and M were proposed, and so the names 'S4' and 'S5' have been preserved for historical reasons.

**EXERCISE 2.3** Prove in the systems indicated. You may appeal to any results established previously in this book or proven by you during the completion of these exercises. Try to do them without looking at the hints.

a) $\Box\Box A \leftrightarrow \Box A$ in S4. (Hint: Use a special case of (M) for one direction.)
b) $\Box\Box{\sim}A \,/\, \Box{\sim}{\sim}\Box{\sim}A$ in K.
c) $\Diamond\Diamond A \leftrightarrow \Diamond A$ in S4. (Hint: Use the solution to Exercise 2.1a for one direction, and use 2.3b for the other.)
d) $\Box\Diamond\Diamond A \leftrightarrow \Box\Diamond A$ in S4. (Hint: Use (GN) with the solution for 2.3c.)
e) $\Box\Diamond A \leftrightarrow \Diamond A$ in S5. (Hint: Use a special case of (M) for one direction.)
f) (B) in S5. (Hint: Use the solution to Exercise 2.1a.)
g) $\Box{\sim}\Box{\sim}{\sim}A \,/\, \Box{\sim}\Box A$ in K.
h) $\Diamond\Box A \rightarrow A$ in B. (Hint: Use this version of B: ${\sim}A \rightarrow \Box\Diamond{\sim}A$, and the previous exercise.)
i) $\Diamond\Box A \leftrightarrow \Box A$ in S5. (Hint: In one direction, use Exercise 2.1a. In the other, use (${\sim}\Box$) (see Exercise 1.10b), this instance of (5): $\Diamond{\sim}A \rightarrow \Box\Diamond{\sim}A$, and the solution to g.)

The scheme that names a system by listing the names of its axioms is awkward in another respect. There are many equivalent ways to define provability in S5. All of the following collections of axioms are equivalent to S5 = M+(5).

M+(B)+(5)
M+(4)+(5)
M+(4)+(B)+(5)
M+(4)+(B)

By saying S5 is equivalent to a collection of rules, we mean that the arguments provable in S5 are exactly the ones provable using the rules in that collection. For example, consider M+(B)+(5). This is equivalent to S5, because we showed in Exercise 2.3f that (B) is provable in S5. Therefore, (B) adds nothing new to the powers of S5. Whenever we have a proof of an argument using (B), we can replace the use of (B) with its derivation in S5.

**EXERCISE 2.4**

a) Prove (4) in S5. (Hint: First prove $\Box A \rightarrow \Box\Diamond\Box A$ (a special case of (B)) and then prove $\Box\Diamond\Box A \rightarrow \Box\Box A$ using the solution to Exercise 2.3i.)
b) Using the previous result, explain why S5 is equivalent to M+(4)+(5), and M+(4)+(B)+(5).
c) Prove S5 is equivalent to M+(4)+(B) by proving (5) in M+(4)+(B). (Hint: Begin with this special case of (B): $\Diamond A \rightarrow \Box\Diamond\Diamond A$. Then use (4) to obtain $\Box\Diamond\Diamond A \rightarrow \Box\Diamond A$.)

It is more natural to identify a formal system by what it proves rather than by how it is formulated. We want to indicate, for example, that M+(5) and M+(4)+(B) are really the same system, despite the difference in their axioms. If we name systems by their axioms, we will have many different names ('M5', 'MB5', 'M45', . . and so on) for the same system. For a system like S5, which has many equivalent formulations, it is just as well that there is a single name, even if it is somewhat arbitrary.

Exercise 2.3 was designed to familiarize you with some of the main features of S4 and S5. In S4, a string of two boxes ($\Box\Box$) is equivalent to one box ($\Box$). As a result, any string of boxes is equivalent to a single box, and the same is true of strings of diamonds.

**EXERCISE 2.5** Prove $\Box\Box\Box A \leftrightarrow \Box A$, $\Diamond\Diamond\Diamond A \leftrightarrow \Diamond A$, and $\Box\Box\Box\Box A \leftrightarrow \Box A$ in S4, using the strategies employed in Exercises 2.3a and 2.3c.

The system S5 has stronger principles for simplifying strings of modal operators. In S4 a string of modal operators of the same kind can be replaced for the operator, but in S5 strings containing both boxes and diamonds are equivalent to the last operator in the string. This means that one never needs to iterate (repeat) modal operators in S5, since the additional operators are superfluous.

| |
|---|
| **EXERCISE 2.6** Prove $\Diamond\Box\Diamond A \leftrightarrow \Diamond A$ and $\Box\Diamond\Box A \leftrightarrow \Box A$ in S5. |

The following chart reviews the iteration principles for S4 and S5.

S4: $\Box\Box \,..\, \Box = \Box$ $\quad\Diamond\Diamond \,..\, \Diamond = \Diamond$
S5: $00 \,..\, \Box = \Box$ $\quad 00 \,..\, \Diamond = \Diamond$, where 0 is $\Box$ or $\Diamond$

The axiom (B): $A \rightarrow \Box\Diamond A$ raises an important point about the interpretation of modal formulas. (B) says that if A is the case, then A is necessarily possible. One might argue that (B) should always be adopted in modal logic, for surely if A is the case, then it is necessary that A is possible. However, there is a problem with this claim that can be exposed by noting that $\Diamond\Box A \rightarrow A$ is provable from (B). (See Exercise 2.3.h.) So $\Diamond\Box A \rightarrow A$ should be acceptable if (B) is. However, $\Diamond\Box A \rightarrow A$ says that if A is possibly necessary, then A is the case, and this is far from obvious.

What has gone wrong? The answer is that we have not been careful enough in dealing with an ambiguity in the English rendition of $A \rightarrow \Box\Diamond A$. We often use the expression 'if A then necessarily B' to express that the conditional 'if A then B' is necessary. This interpretation of the English corresponds to $\Box(A \rightarrow B)$. On other occasions we mean that if A, then B is necessary: $A \rightarrow \Box B$. In English, 'necessarily' is an adverb, and since adverbs are usually placed near verbs, we have no natural way to indicate whether the modal operator applies to the whole conditional, or to its consequent. This unfortunate feature creates ambiguities of scope, that is, ambiguities that result when it is not clear which portion of a sentence is governed by an operator.

For these reasons, there is a tendency to confuse (B): $A \rightarrow \Box\Diamond A$ with $\Box(A \rightarrow \Diamond A)$. But $\Box(A \rightarrow \Diamond A)$ is not the same as (B), for $\Box(A \rightarrow \Diamond A)$ is a theorem of M, and (B) is not. So one must take special care that our positive reaction to $\Box(A \rightarrow \Diamond A)$ does not infect our evaluation of (B). One simple way to protect ourselves is to consider the sentence: $\Diamond\Box A \rightarrow A$, where ambiguities of scope do not arise.

| |
|---|
| **EXERCISE 2.7** Prove $\Box(A \rightarrow \Diamond A)$ in M. |

One could engage in endless argument over the correctness or incorrectness of (4), (B), (5), and the other iteration principles that have been suggested for modal logic. Failure to resolve such controversy leads some people to be very suspicious of modal logic. "How can modal logic be logic at all," they say, "if we can't decide what the axioms should be?" My answer is to challenge the idea that we must decide on the axioms in order for modal logic to be coherent. Necessity is a many-sided notion, and so we should not expect it to correspond to a single logic. There are several viable modal systems, each one appropriate to a different way in which we understand and use the word 'necessarily'. This idea will be explored in more detail when we provide semantics for modal logics in Chapter 3.

## 2.2. Duals

The idea of the dual of a sentence is a useful notion in modal logic. The following pairs of symbols are defined to be *mates* of each other.

$\&$ $\vee$
$\Box$ $\Diamond$
$\forall$ $\exists$

We have not introduced quantifiers $\forall$ and $\exists$ in our logics yet, but we will later, and so they are included now for future reference. Let $A^*$ be the sentence that results from replacing each symbol in A on the above list with its mate. Now we may define the *dual* for sentences that have the shapes $A \to B$ and $A \leftrightarrow B$, provided $\to$, $\leftrightarrow$, and ~ do not appear in A or B. The dual of $A \to B$ is $B^* \to A^*$, and the dual of $A \leftrightarrow B$ is $A^* \leftrightarrow B^*$. Notice that sentences that do not have the shapes $A \to B$ or $A \leftrightarrow B$ do not have duals. The best way to understand what duals are is to construct a few. The dual of (B): $A \to \Box\Diamond A$ is $(\Box\Diamond A)^* \to (A)^*$, that is, $\Diamond\Box A \to A$. The dual of $\Box(A\&B) \to (\Diamond A \vee \Diamond B)$ is $(\Diamond A \vee \Diamond B)^* \to \Box(A\&B)^*$. But $(\Diamond A \vee \Diamond B)^*$ is $\Box A\&\Box B$ and $\Box(A\&B)^*$ is $\Diamond(A \vee B)$, and so we obtain $(\Box A\&\Box B) \to \Diamond(A \vee B)$, which is, therefore, its own dual.

**EXERCISE 2.8** Find the duals of the following sentences.

a) $\Box A \to \Box\Box A$
b) $(\Box A\&\Box B) \leftrightarrow \Box(A\&B)$
c) $\Diamond A \to \Box\Diamond A$
d) $\Box(A \vee B) \to (\Box A \vee \Box B)$
e) $\forall x \Box Ax \leftrightarrow \Box \forall x Ax$
f) $\Box(\Box A \to A)$ (trick question)
g) $\Box A \to \Diamond A$
h) $A \to \Box\Diamond A$

The reason duals are interesting is that adding an axiom to K is equivalent to adding its dual as an axiom. Since sentences with the shape

$A\rightarrow\Box\Diamond A$ are provable in B, it follows that all sentences of the (dual) shape $\Diamond\Box A\rightarrow A$ are provable in B as well. In fact, we could have used $\Diamond\Box A\rightarrow A$ instead of $A\rightarrow\Box\Diamond A$ to define the system B. Being able to recognize duals can be very helpful in working out proof strategies and for appreciating the relationships among the various modal logics.

**EXERCISE 2.9** Using duals, produce alternatives to the axioms (M), (4), and (5).

**EXERCISE 2.10** To help verify that an axiom is equivalent to its dual, reconstruct proofs of the following facts:

a) The dual of (M) is derivable in K plus (M). (Exercise 2.1a)
b) The dual of (4) is derivable in K plus (4). (Exercise 2.3c)
c) The dual of (B) is derivable in K plus (B). (Exercise 2.3h)
d) The dual of (5) is derivable in K plus (5). (Exercise 2.3i)

## 2.3. Deontic Logic

A number of modal logics that are not appropriate for necessity and possibility can be built from the basic system K. They lack the characteristic axiom of M: $\Box A\rightarrow A$. Deontic logics, the logics of obligation, are an important example. Deontic logics introduce the primitive symbol **O** for 'it is obligatory that', from which symbols for 'it is permitted that' and 'it is forbidden that' are defined as follows:

(DefP) $\mathbf{P}A =_{df} \sim\mathbf{O}\sim A$
(DefF) $\mathbf{F}A =_{df} \mathbf{O}\sim A$

The symbol **'O'** in deontic logic plays exactly the same role as $\Box$ did in the system K. A basic system D of deontic logic can be constructed by adding the characteristic deontic axiom (D) to the rules of K, with **O** playing the role of $\Box$.

**System D = (OOut) + (OIn) + (D).**

```
OOut            OIn
OA              |_O
:               |  :
|_O             |  A
|------         |--------
|  A  (OOut)    OA   (OIn)
```

**(D)** $\mathbf{O}A\rightarrow\mathbf{P}A$

Although the principles of K seem reasonable for deontic logic, one feature has bothered some people. The rule of Necessitation is derivable in K, so **O**A follows when A is a theorem. For example, since p→p is provable in PL, **O**(p→p) will follow. However, it is odd to say that p→p is obligatory (though just as odd, I would think, to deny that p→p is obligatory). Questions about whether A is obligatory do not arise when A is a theorem, because the language of obligation and permission applies to sentences whose truth values depend on our actions. No matter what we do, p→p will remain true, so there is no point in commanding or even permitting it.

Even though our feelings about K are, for this reason, neutral, K does lead to reasonable results where we do have strong intuitions. For example, the theorems about K concerning the distribution of operators over the connectives all seem reasonable enough. We will be able to prove that **O**(A&B) is equivalent to **O**A&**O**B, that **O**(A∨B) is entailed by **O**A∨**O**B but not vice versa, that **P**(A∨B) is equivalent to **P**A∨**P**B, that **O**(A→B) entails **P**A→**P**B, and so forth. These are widely held to be exactly the sort of logical properties that **O** and **P** should have. Later, when we learn about modal semantics, we will find further support for the view that deontic logics can be built on the principles of K.

## 2.4. The Good Samaritan Paradox

There is a second problem with using K for deontic logic that has been widely discussed (Åqvist, 1967). The objection concerns a special case of the deontic version of General Necessitation (GN):

A ⊢ B
-----------
OA ⊢ OB

Now imagine that a Good Samaritan finds a wounded traveler by the side of the road. Assume that our moral system is one where the Good Samaritan is obliged to help the traveler. Consider the following instance of (GN):

1. The Good Samaritan binds the traveler's wound ⊢ the traveler is wounded.
2. The Good Samaritan ought to bind the traveler's wound ⊢ the traveler ought to be wounded.

Argument (1) appears to be logically valid, for you can't fix a person's wounds if the person is not wounded. However, the second argument

(2) appears to be invalid. It is true that the Good Samaritan should help the traveler, but it is false that the traveler ought to be wounded. So it appears we must reject (GN), since it leads us from a valid to an invalid argument.

Let us resolve the paradox by symbolizing (1) in deontic logic. Although a full analysis requires predicate letters and quantifiers, it is still possible to present the gist of the solution to the problem using propositional logic. (For a more sophisticated treatment, see Exercise 18.18 in Chapter 18.) The central issue concerns how we are to translate sentence (3).

(3) The Good Samaritan binds the traveler's wound.

Sentence (3) really involves two different ideas: that the traveler is wounded, and that the Good Samaritan binds the wound. So let us use the following vocabulary:

W = The traveler is wounded.
B = The Good Samaritan binds the wound.

Now arguments (1) and (2) may be represented as an instance of (GN) as follows.

$$\frac{W\&B \vdash W}{\mathbf{O}(W\&B) \vdash \mathbf{O}W}$$

However, this does not count as a reason to reject (GN), for if it were, the argument $\mathbf{O}(W\&B) \vdash \mathbf{O}W$ would need to have a true premise and a false conclusion. However, the premise is false. It is wrong to say that it ought to be the case that both the traveler is wounded and the Good Samaritan binds the wound, because this entails that the traveler ought to be wounded, which is false.

One might object that the claim that the Good Samaritan ought to bind the traveler's wound appears to be true, not false. There is, in fact, a way to represent this where it is true, namely $W\&\mathbf{O}B$. This says that the traveler is wounded and the Good Samaritan ought to bind the wound. In this version, W does not lie in the scope of the modal operator, so it does not claim that the traveler ought to be wounded. But if this is how the claim is to be translated, then (1) and (2) no longer qualify as an instance of (GN), for in (GN) the $\mathbf{O}$ must include the whole sentence W&B.

$$\frac{W\&B \vdash W}{W\&\mathbf{O}B \vdash \mathbf{O}W} \quad \text{not an instance of (GN)!}$$

So the Good Samaritan paradox may be resolved by insisting that we pay close attention to the scope of the deontic operator **O**, something that is difficult to do when we present arguments in English. Sentence (3) is ambiguous. If we read it as $\mathbf{O}(W\&B)$, we have an instance of (GN), but the second argument's premise is false, not true. If we read (3) as $W\&\mathbf{O}B$, the premise of that argument is true, but the argument does not have the right form to serve as a case of (GN). Either way it is possible to explain why the reasoning is unsound without rejecting (GN).

## 2.5. Conflicts of Obligation and the Axiom (D)

We have already remarked that we do not want to adopt the analogue of (M), $\mathbf{O}A\rightarrow A$, in deontic logic. The reason is that if everything that ought to be is the case, then there is no point to setting up a system of obligations and permissions to regulate conduct. However, the basic deontic system D contains the weaker axiom (D), which is the analogue of $\Box A\rightarrow\Diamond A$, a theorem of M.

(D) $\mathbf{O}A\rightarrow\mathbf{P}A$

Axiom (D) guarantees the consistency of the system of obligations by insisting that when A is obligatory, it is permissible. A system that commands us to bring about A, but doesn't permit us to do so, puts us in an inescapable bind.

Some people have argued that D rules out conflicts of obligations. They claim we can be confronted with situations where we ought to do both A and ~A. For example, I ought to protect my children from harm, and I ought not to harbor a criminal, but if my child breaks the law and I am in a position to hide him so that he escapes punishment, then it seems I ought to turn him in because he is a criminal ($\mathbf{O}A$), and I ought not to turn him in to protect him from harm ($\mathbf{O}{\sim}A$). However, it is easy to prove ${\sim}(\mathbf{O}A\&\mathbf{O}{\sim}A)$ in D, because (D) amounts to $\mathbf{O}A\rightarrow{\sim}\mathbf{O}{\sim}A$, which entails ${\sim}(\mathbf{O}A\&\mathbf{O}{\sim}A)$ by principles of propositional logic. So it appears that $\mathbf{O}A\&\mathbf{O}{\sim}A$, which expresses the conflict of obligations, is denied by D.

I grant that conflicts of obligation are possible, but disagree with the conclusion that this requires the rejection of D. Conflicts of obligation arise not because a single system of obligations demands both A and ~A, but because *conflicting systems* of obligation pull us in different directions. According to the law, there is no question that I am obligated to turn in my son, but according to a more primitive obligation to my children, I should hide him. Very often, there are higher systems of obligation that are designed specifically to resolve such conflicts. If **O** is used to express obligation in a higher moral system that says that the law comes first in this situation, then it is simply false that I should refrain from turning him in, and it is no longer true that both **O**A and **O**~A.

Sometimes we have no explicit system that allows us to resolve conflicts between different types of obligation. Even so, we still do not have a situation where any one system commands both A and ~A. In our example, we have two systems, and so we ought to introduce two symbols: (say) $\mathbf{O}_l$ for legal, and $\mathbf{O}_f$ for familial obligation. Then $\mathbf{O}_l$A is true but $\mathbf{O}_l$~A is false, and $\mathbf{O}_f$~A is true while $\mathbf{O}_f$A is false when A is read 'I turn in my child'. The axiom (D) is then perfectly acceptable for both deontic operators $\mathbf{O}_l$ and $\mathbf{O}_f$, and so the conflict of obligations does not show that (D) is wrong.

## 2.6. Iteration of Obligation

Questions about the iteration of operators, which we discussed for modal logics, arise again in deontic logic. In some systems of obligation, we interpret **O** so that **OO**A just amounts to **O**A. 'It ought to be that it ought to be' is just taken to be a sort of stuttering; the extra 'oughts' don't add anything. If this is our view of matters, we should add an axiom to D to ensure the equivalence of **OO**A and **O**A.

(**OO**) $\mathbf{O}\mathrm{A} \leftrightarrow \mathbf{OO}\mathrm{A}$

If we view (**OO**) as composed of a pair of conditionals, we find that it "includes" the deontic analogue $\mathbf{O}\mathrm{A}\rightarrow\mathbf{OO}\mathrm{A}$ of the modal axiom (4), $\Box\mathrm{A}\rightarrow\Box\Box\mathrm{A}$. In system M, the converse $\Box\Box\mathrm{A}\rightarrow\Box\mathrm{A}$ is derivable, so it guarantees the equivalence of $\Box\Box$A and $\Box$A. But in deontic logic, we don't have (M), and so we need the equivalence in (**OO**). Once we have taken the point of view that adopts (**OO**), there seems to be no reason not to accept the policy of iteration embodied in S5 and simply ignore any extra deontic operators. So we would add an equivalence to guarantee the equivalence of **OP**A and **P**A.

(**OP**) $\mathbf{P}\mathrm{A} \leftrightarrow \mathbf{OP}\mathrm{A}$

There is another way to interpret **O** so that we want to reject both (**OO**) and (**OP**). On this view, 'it ought to be that it ought to be that A' commands adoption of some obligation that we may not already have. This is probably a good way to look at the obligations that come from the legal system, where we generally have legal methods for changing the laws and, hence, our obligations. Most systems that allow us to change our obligations do so only within limits, and these limits determine the obligations that are imposed on us concerning what we can obligate people to do. Under this reading, **OO**A says that according to the system, we have an obligation to obligate people to bring about A, that is, that no permissible changes in our obligations would release us from our duty to bring about A. Similarly, **OP**A says that we have an obligation in the system to permit A, that is, that we are not allowed to change the obligations so that people aren't permitted to do A. For example, according to a constitutional system, one might be allowed to make all sorts of laws, but not any that conflict with the fundamental principles of the constitution itself. So a system of law might obligate its citizens to permit freedom of speech (**OP**s), but this would be quite different from saying that the system permits freedom of speech (**P**s).

If this is how we understand **O** and **P**, it is clear that we cannot accept (**OO**) or (**OP**). If A is obligatory, it doesn't follow that it *has* to be that way, that is, that it is obligatory that A be obligatory. Also, if A is permitted, it doesn't follow that it has to be permitted. On this interpretation of obligation it is best to use the deontic logic D and drop (**OO**) and (**OP**).

There is one further axiom that we may want to add in deontic logics regardless of which interpretation we like. It is (OM).

(OM) $\mathbf{O}(\mathbf{O}A \to A)$

This says that it ought to be the case that if A ought to be the case, then it is the case. Of course, if A ought to be, it doesn't follow that A is the case. We already pointed out that $\mathbf{O}A \to A$ is not a logical truth. But even so, it *ought* to be true, and this is what (OM) asserts. In almost any system of obligation then, we will want to supplement D with (OM).

**EXERCISE 2.11** Show that sentences of the following form can be proven in D plus (OO): $\mathbf{O}A \to \mathbf{OP}A$.

## 2.7. Tense Logic

Tense logics (Burgess, 1984; Prior, 1967) have provoked much less philosophical controversy than have deontic or even modal logics. This is probably because the semantics for tense logics can be given in a very natural way, one that is hard to challenge. Still there is no one system