Chapter 6: Formal Proofs and Boolean Logic

The Fitch program, like the system $\mathcal{F}$, uses "introduction" and "elimination" rules. The ones we've seen so far deal with the logical symbol =. The next group of rules deals with the Boolean connectives $\wedge$, $\vee$, and $\neg$.

§ 6.1 Conjunction rules

Conjunction Elimination ($\wedge$ Elim)

$$\begin{array}{c|l} & P_1 \wedge \ldots \wedge P_i \wedge \ldots \wedge P_n \\ \triangleright & P_i \end{array}$$

This rule tells you that if you have a conjunction in a proof, you may enter, on a new line, any of its conjuncts. (P_i here represents any of the conjuncts, including the first or the last.)

Notice this important point: the conjunction to which you apply $\wedge$ **Elim** must appear **by itself** on a line in the proof. You cannot apply this rule to a conjunction that is **embedded** as part of a larger sentence. For example, this is **not** a valid use of $\wedge$ **Elim**:

$$\begin{array}{c|ll} & 1.\ \neg(\text{Cube}(a) \wedge \text{Large}(a)) & \\ \hline \triangleright & 2.\ \neg\text{Cube}(a) & \quad\quad \text{X} \quad \wedge\ \textbf{Elim}: 1 \end{array}$$

The reason this is not valid use of the rule is that $\wedge$ **Elim** can only be applied to conjunctions, and the line that this "proof" purports to apply it to is a **negation**. And it's a good thing that this move is not allowed, for the inference above is not valid—from the premise that *a* is not a large cube it does not follow that *a* is not a cube. *a* might well be a small cube (and hence not a large cube, but still a cube).

This same restriction—the rule applies to the sentence on the entire line, and not to an embedded sentence—holds for all of the rules of $\mathcal{F}$, by the way. And so Fitch will not let you apply $\wedge$ **Elim** or any of the rules of inference to sentences that are embedded within larger sentences.

Conjunction Introduction ($\wedge$ Intro)

$$\begin{array}{c|l} & P_1 \\ & \Downarrow \\ & P_n \\ \triangleright & P_1 \wedge \ldots \wedge P_n \end{array}$$

This rule tells you that if you have a number of sentences in a proof, you may enter, on a new line, their **conjunction**. Each conjunct must appear individually on its own line, although they may occur in any order. Thus, if you have A on line 1 and B on line 3, you may enter B $\wedge$ A on a subsequent line. (Note that the lines need not be consecutive.) You may, of course, also enter A $\wedge$ B.

Default and generous uses of rules

Unlike system $\mathcal{F}$, Fitch has both default and generous uses of its rules. A **default** use of a rule is what will happen if you cite a rule and a previous line (or lines) as justification, but do not enter any new sentence. If you ask Fitch to check out the step, it will enter a sentence for you. A **generous** use of a rule is one that is not is not strictly in accordance with the rule as stated in $\mathcal{F}$ (i.e., $\mathcal{F}$ would not allow you to derive it in a single step), but is still a valid inference. Fitch will often let you do this in one step.

Default and generous uses of the $\wedge$ rules

- **Default use**: if you cite a conjunction and the rule $\wedge$ **Elim**, and ask Fitch to check out the step, Fitch will enter the **leftmost** conjunct on the new line.

- **Generous use**: if you cite a conjunction and the rule $\wedge$ **Elim**, you may manually enter **any of its conjuncts**, or you may enter **any conjunction** whose conjuncts are among those in the cited line. Fitch will check out the step as a valid use of the rule.

Note just how generous Fitch is about $\wedge$ **Elim**—from the premise

$$A \wedge B \wedge C \wedge D$$

Fitch will allow you to obtain any of the following (among others!) by a "generous" use of the rule:

$$A$$
$$B$$
$$C$$
$$D$$
$$A \wedge B$$
$$A \wedge C$$
$$A \wedge D$$
$$B \wedge C$$
$$B \wedge D$$
$$C \wedge D$$
$$A \wedge B \wedge C$$
$$B \wedge A \wedge D$$
$$D \wedge A \wedge C$$
$$B \wedge A \wedge C \wedge D$$

§ 6.2 Disjunction rules

Disjunction Introduction ($\vee$ Intro)

$$P_i$$
$$\triangleright \quad P_1 \vee \ldots \vee P_i \vee \ldots \vee P_n$$
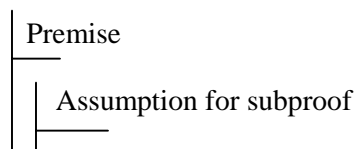
This rule tells you that if you have a sentence on a line in a proof, you may enter, on a new line, any disjunction of which it is a disjunct. (P_i here represents any of the disjuncts, including the first or the last.)

Disjunction Elimination (∨ Elim)

This is the formal rule that corresponds to the method of proof by cases. It incorporates the formal device of a **subproof**.
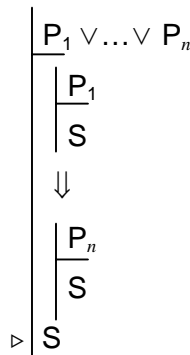
A subproof involves the temporary use of an additional assumption, which functions in a subproof the way the premises do in the main proof under which it is subsumed.

We place a subproof within a main proof by introducing a new vertical line, inside the vertical line for the main proof. We **begin the subproof with an assumption** (any sentence of our choice), and place a new Fitch bar under the assumption:

Premise

    Assumption for subproof

The subproof may be ended at any time. When the subproof ends, the vertical line stops, and the next line either "jumps out" to the original vertical proof line, or a new subproof may be begun. As we'll see, ∨ **Elim** involves the use of two (or more) subproofs, typically (although not necessarily) entered one immediately after the other.

The rule:

$$P_1 \vee \ldots \vee P_n$$

$$P_1$$
$$S$$

$$\Downarrow$$

$$P_n$$
$$S$$

$$\rhd \quad S$$

What the rule says is this: if have a disjunction in a proof, and you have shown, through a sequence of subproofs, that each of the disjuncts (together with any other premises in the main proof) leads to the same conclusion, then you may derive that conclusion from the disjunction (together with any main premises cited within the subproofs).

This is clearly a **formal version of the method of proof by cases**. Each of the P_i represents one of the cases. Each subproof represents a demonstration that, in each case, we may conclude S. Our conclusion is that S is a consequence of the disjunction together with any of the main premises cited within the subproofs.

When you do the **You try it** on p. 151, notice, as you proceed through the proof, that after step 4 you must **end the subproof** first, before you **begin the next subproof**.

To do these things, you can click on the options in the Proof menu. But it is easier and quicker to use the **keyboard shortcuts**: to end a subproof, press **Control-E;** to begin a new subproof, press **Control-P**. Another handy shortcut is **Control-A** for adding a new line after the current line, as part of the same proof or subproof. (Any time you add a new line, Fitch will wait for you to write in a sentence and cite a justification for it.)

 Revised 6/1/04

Note also that the use of **Reit** is strictly optional. For example, in the proof on p. 151, step 5 is not required. The proof might look like the one in (on Supplementary Exercises page) and it will check out.

Default and generous uses of the ∨ rules

- **Default uses**

  o ∨ **Elim**: if you cite a disjunction and some subproofs, with each subproof beginning with a different disjunct of the disjunction, and all subproofs ending in the same sentence, S, cite the rule ∨ **Elim,** and ask Fitch to check it out, Fitch will enter S.

  o ∨ **Intro**: if you cite a sentence and the rule ∨ **Intro**, and ask Fitch to check it out, Fitch will enter the cited sentence followed by a ∨, and wait for you to enter whatever disjunct you wish.

- **Generous use**: if your cited disjunction contains more than two disjuncts, you don't need a separate subproof for each disjunct. A subproof may begin with a **disjunction** of just some of the disjuncts of the cited disjunction. When you ask Fitch to check the step, Fitch will check it out as a valid use of the rule, so long as every disjunct of the cited disjunction is either a subproof assumption or a disjunct of such an assumption.

§ 6.3 Negation rules

Negation Elimination (¬ Elim)

This simple rule allows us to eliminate "double negations."

```
   │ ¬¬P
 ▷ │ P
```

Negation Introduction (¬ Intro)

This is our formal version of the method of indirect proof, or proof by contradiction. It requires the use of a subproof. The idea is this: if an assumption made in a subproof leads to ⊥, you may close the subproof and derive as a conclusion the **negation** of the sentence that was the assumption.

```
   │ │ P
   │ ├─
   │ │ ⊥
 ▷ │ ¬P
```

To use this rule, we will need a way of getting the contradiction symbol, ⊥, into a proof. We will have a special rule for that, one which allows us to enter a ⊥ if we have, on separate lines in our proof (or subproof) both a sentence and its negation.

⊥ Introduction (⊥ Intro)

```
   │ P
   │ ¬P
 ▷ │ ⊥
```

Note that the cited lines must be **explicit** contradictories, i.e., sentences of the form P and ¬P. This means that the two sentences must be symbol-for-symbol identical, except for the negation sign at the beginning of one of them. It is not enough that the two sentences be TT-inconsistent with one another, such as A ∨ B and ¬A ∧ ¬B. Although these two are contradictories (semantically speaking) since they must always have opposite truth-values, they are not **explicit** contradictories (syntactically speaking) since they are not written in the form P and ¬P.

To try out these two rules, do the **You try it** on p. 156.

Other kinds of contradictions

The rule of ⊥ **Intro** lets us derive ⊥ whenever we have a pair of sentences that are explicit contradictories. But there are other kinds of contradictory pairs: non-explicit TT-contradictories, FO-contradictories that are TT-consistent, logical contradictories that are FO-consistent, and TW-contradictories that are logically consistent. Here are some examples of these other types of contradictory pairs:

1. Tet(a) ∨ Tet(b)   and   ¬Tet(a) ∧ ¬Tet(b)
2. Cube(b) ∧ a = b   and   ¬Cube(a)
3. Cube(b)   and   Tet(b)
4. ¬Tet(a) ∧ ¬Cube(a)   and   ¬Dodec(a)

In example (1) we have TT-contradictory sentences but not an explicit contradiction, as defined above. In (2) we have a pair of sentences that are FO-inconsistent (they cannot both be true in any possible circumstance), but not TT-inconsistent (a truth-table would not detect their inconsistency). In (3) we have a pair that are logically inconsistent but not FO-inconsistent (or TT-inconsistent). Finally, in (4) we have a pair that are TW-contradictories (there is no Tarski world in which both of these sentences are simultaneously true), although they are logically consistent—it is possible for an object to be neither a tetrahedron nor a cube nor a dodecahedron (it may be a sphere).

The rule of ⊥ **Intro** does not apply directly in any of these examples. In each case it takes a bit of maneuvering first before we come up with an **explicitly** contradictory pair of sentences, as required by the rule.

Example 1

1. Tet(a) ∨ Tet(b)		
2. ¬Tet(a) ∧ ¬Tet(b)		
3. ¬Tet(a)	∧ **Elim**: 2	
4. ¬Tet(b)	∧ **Elim**: 2	
5. Tet(a)		
6. ⊥	⊥ **Intro**: 3, 5	
7. Tet(b)		
8. ⊥	⊥ **Intro**: 4, 7	
9. ⊥	∨ **Elim**: 1, 5-6, 7-8	

Here we used ∧ **Elim** twice, to get the two conjuncts of (2) separately, and then constructed a proof by cases to show that whichever disjunct of line (1) we choose, we get to an explicit contradiction.

Example 2

1. Cube(b) ∧ a = b
2. ¬Cube(a)

3. Cube(b)   ∧ **Elim**: 1
4. a = b   ∧ **Elim**: 1
5. ¬Cube(b)   = **Elim**: 2, 4
6. ⊥   ⊥ **Intro**: 3, 5

Here we used ∧ **Elim** to get Cube(b) and a = b to stand alone, and then = **Elim** (substituting b for a in line 2) to get the explicit contradictory of Cube(b).

Example 3

1. Cube(b)
2. Tet(b)

3. ¬Tet(b)   **Ana Con**: 1
4. ⊥   ⊥ **Intro**: 2, 3

Here we had to use **Ana Con**. Of course, as long as we were going to use **Ana Con** at all, we could have used it instead of ⊥ **Intro** to get our contradiction, as follows:

1. Cube(b)
2. Tet(b)

3. ⊥   **Ana Con**: 1, 2

Example 4

1. ¬Tet(a) ∧ ¬Cube(a)
2. ¬Dodec(a)

3. ⊥   **Ana Con**: 1, 2

To see these different forms of contradictions in action, do the **You try it** on p. 159. It's an excellent illustration of these differences. You'll find that you often need to use the **Con** mechanisms to introduce a ⊥ into a proof, since ⊥ **Intro** requires that there be an explicit contradiction in the form of a pair of sentences P and ¬P.

⊥ Elimination (⊥ Elim)

⊥

▷ P

The rule of ⊥ elimination is added to our system strictly as a convenience—we do not really need it. It allows us, once we have a ⊥ in a proof, to enter any sentence we like. (We've already seen that every sentence follows from a contradiction.) As p. 161 shows, we can easily do without this rule with a four step "workaround."

Default and generous uses of the ¬ rules

Note the default and generous uses of these rules in Fitch (p. 161). With **¬ Elim**, you don't need two steps to get from ¬¬¬¬P to P (passing through the intermediate step ¬¬P). You can do it in one step. In fact, this is also the default use of the rule (if you cite the rule and ask Fitch to fill in the derived line).

In the case of **¬ Intro**, where the subproof assumption is a **negation**, ¬P, and the subproof ends with a ⊥:

- **Default use**: if you end the subproof, cite the subproof and rule **¬ Intro**, and ask Fitch to check the step, Fitch will enter the line ¬¬P.

- **Generous use**: if you end the subproof, enter the line P manually, cite the subproof and rule **¬ Intro**, and ask Fitch to check the step, Fitch will check it out as a valid use of the rule.

§ 6.4 The proper use of subproofs

Once a subproof has ended, none of the lines in that subproof may be cited in any subsequent part of the proof. Look at the "proof" on p. 163 to see what can happen if this restriction is violated.

How Fitch keeps you out of trouble

When you are working in system $\mathcal{F}$, you can enter erroneous lines like line 8 on p. 163 and never be aware of it. But **Fitch won't let you do this!** To see what happens, look at Page163.prf.

Notice that when we try to justify line 8 by means of ∧ **Intro**, Fitch will not let us cite the line that occurs inside the subproof that has already been closed.

When a subproof **ends**, we say that its assumption has been **discharged**. After an assumption is discharged, one may not cite any line that depended on that assumption.

Note that it is **permissible**, while within a subproof, to cite lines that occur **outside** that subproof. So, for example, one may, while within a subproof, refer back to the original premises, or conclusions derived from them. One must just take care not to cite lines that occur in subproofs whose assumptions have been discharged.

Subproofs may be **nested**—one subproof may begin before another is ended. In such cases, the last subproof begun must be ended first. The example on p. 165 illustrates such a nested subproof.

§ 6.5 Strategy and tactics

Keep in mind what the sentences in your proof mean

Don't just look at the sentences in your proof as meaningless collections of symbols. Remember what the sentences mean as you try to discover whether the argument is valid.

If you're not told whether the argument is valid, you can use Fitch's **Taut Con** mechanism to check it out. If you discover that the argument is not valid, you should not waste time trying to find a proof.

Try to sketch out an informal proof

This will often give you a good formal proof strategy. An informal indirect proof can be turned into a use of ¬ **Intro** in $\mathcal{F}$. An informal proof by cases can be turned into a use of ∨ **Elim** in $\mathcal{F}$.

Try "working backwards"

This is a very basic strategy. It involves figuring out what intermediate conclusion you might reach that would enable you to obtain your ultimate conclusion, and then taking that intermediate conclusion as your new goal. You can then work backwards to achieve this new goal: figure out what other intermediate conclusion you might reach that would enable you to obtain your first intermediate conclusion, and so on. Working backward in this way, you may discover that it is obvious to you how to obtain one of those intermediate conclusions. You then have all the pieces you need to assemble the proof.

Fitch is very helpful to you in using this strategy, for you can work from the "bottom up" as well as from the "top down." To see this, do the **You try it** on p. 168 (open the file Strategy 1). You will note that you can cite a line, or a subproof, as part of a justification even before you have justified the line itself. This shows up with the two innermost subproofs (3-5 and 6-8) which can be used in the justification of line 9 even before lines 5 and 8 themselves have been justified.

This gives you a good method for checking out your strategy.

An example

> (A ∧ ¬B) ∨ (C ∨ D)
> ¬C ∧ B
> ──
> ¬¬D

Open Ch6Ex2a – you'll find this problem on the Supplementary Exercises page of the web site. We can start by working backwards. We can get ¬¬D from D by assuming ¬D and using ¬ **Intro**. So our goal will be to get D.

Our first premise is a disjunction, so that suggests a proof by cases. We will have a separate subproof for each case, deriving D at the end of each subproof. Open Ch6Ex2b Notice that our strategy checks out when we apply ∨ **Elim**, and that our strategy for obtaining ¬¬D also checks out.

Case 1: A ∧ ¬B

The second conjunct, ¬B, contradicts the second conjunct of premise 2. So we can derive ⊥ by ⊥ **Intro** and then derive D by ⊥ **Elim**.

Case 2: C

The first conjunct, C, contradicts the first conjunct of premise 2. So we can derive ⊥ by ⊥ **Intro** and then derive D by ⊥ **Elim**.

Case 3: D

> We already have D, so we can use **Reit** to enter it as a conclusion in the subproof. In fact, we can even skip the **Reit** step, as we'll see.

We'll now go after case 1. Open Ch6Ex2c, and follow the strategy above. Notice that the steps check out. Finally, we'll complete cases 2 and 3. Open Ch6Ex2d, and follow the strategy for cases 2 and 3.

Notice that in case 3, we did not need to use **Reit**. In this case, our subproof contains **only** the assumption line. In such a case, we count the assumption line itself as the last line in the subproof, and hence we take that line to have been established, **given the assumption**. This is obviously acceptable, since every sentence is a consequence of itself.

§ 6.6 Proofs without premises

Both system $\mathcal{F}$ and the program Fitch are set up so that a proof may begin with some line other than a premise. For example, it might begin with a use of **= Intro**. Or, it may begin with a subproof assumption.

This means that we may have a proof that has no premises at all! What does such a proof establish? Since a proof establishes that a conclusion is a logical consequence of its premises (i.e., that it must be true if they are), a proof without premises establishes that its conclusion is a logical consequence of the empty set of premises. That is, it establishes that its conclusion must be true, period.

In other words, such a proof establishes that its conclusion is a **logical truth**. See pages 173-4 for examples of such proofs. (The conclusion of a proof without premises is often called a **theorem**, although Barwise and Etchemendy do not use that terminology.)

For a try at proving a logical truth in Fitch, try exercise 6.33. Can you think of a simpler proof of the same logical truth? You can find one at Proof 6.33 simpler.

A Difficult example: 6.41

> We'll try to prove this "starred" tautology: (A ∧ B) ∨¬A ∨ ¬B (And we will do so the hard way, without using **Taut Con** to justify an instance of Excluded Middle.)

> Our tautology says that either A and B are both true, or at least one of them is false. To prove a tautology, it is often easiest to use **indirect proof**: assume the negation of what we're trying to prove, and show that it leads to a contradiction. That is the method we will use.

> To see the general strategy for our proof, open Proof 6.41a. We assume the negation of our desired conclusion, and aim to derive ⊥. We can then apply rule **¬ Intro** (generous version) to get our theorem. Note that this last step checks out.

> Now what we have assumed is the negation of a disjunction. So if what we've assumed is true, each of the disjuncts is false. In particular, both ¬A and ¬B are false. So, given our assumption, we should be able to prove both A and B. That is what we will do next. We will do so by indirect proof: open Proof 6.41b. Note that our indirect method for proving both A and B checks out.

Our strategy is to assume ¬A, reach a ⊥, and deduce A. Then we do the same for B. So we have three questions to answer. (1) How do we show that ¬A leads to a contradiction? (2) How do we show that ¬B leads to a contradiction? (3) Having established both A and B, how do we show that that in turn leads to a contradiction? The answer is the same in every case: by using judiciously chosen applications of ∨ **Intro**.

Our two (inner) assumptions (¬A and ¬B) are, in fact, disjuncts of the theorem we're trying to prove. Hence, we can get from each of those assumptions to the theorem in one application of ∨ **Intro**. That won't prove the theorem (we still have an open assumption), but it will give us a sentence that contradicts our assumption, which is exactly what we want.

To see the complete proof, open Proof 6.41c.

An alternative strategy for 6.41: proof by cases

Notice that a different strategy might yield an equally correct, but much more complicated proof. To see the alternative strategy, open Proof 6.41d.

The idea here is to do a proof by cases:

Case 1

Assume (A ∧ B) and derive the theorem.

Case 2

Assume ¬(A ∧ B) and derive the theorem.

We can use **Taut Con** to obtain the disjunction (A ∧ B) ∨ ¬(A ∧ B) that we need. Then we can apply rule ∨ **Elim** and complete the proof by cases. Note that both of these rule applications check out in Fitch.

Case 1 is easy: it takes only one step of ∨ **Intro**. But case 2 is complicated. To develop the alternative strategy further, open Proof 6.41e. The idea is to try to obtain the right-most disjunction, ¬A ∨ ¬B, by indirect proof. So we assume its negation, viz., ¬(¬A ∨ ¬B).

Note what this sentence says: *neither not A nor not B*, which is equivalent to *A and B*. But (A ∧ B) contradicts our assumption line, so once we have it in our proof, we can obtain ⊥. (Note that our use of ⊥ **Intro** will check out in Fitch.) Our next task will be to obtain the conjunction (A ∧ B) from our indirect proof assumption. We know that it follows, because it's an instance of one of DeMorgan's laws. But those laws are not part of system $\mathcal{F}$, so we will need a different strategy. We will obtain each of A and B separately, and then use ∧**Intro** to get A ∧ B.

To obtain A we use an indirect proof; then we do the same for B. To see how the strategy now looks, open Proof 6.41f. The remaining steps are simple. We assume ¬A for indirect proof. The line we need to contradict is ¬(¬A ∨ ¬B). But ¬A is one of the disjuncts of our negated disjunction. So we use ∨ **Intro** to get the disjunction ¬A ∨ ¬B, and we have our contradiction. This lets us obtain A by means of a generous use of ¬ **Intro**. We repeat this for B.

To see the complete proof, open Proof 6.41g.