Chapter 13: Formal Proofs and Quantifiers

§ 13.1 Universal quantifier rules

Universal Elimination (∀ Elim)

```
  | ∀x S(x)
▹ | S(c)
```

Here x stands for any variable, c stands for any individual constant, and S(c) stands for the result of replacing **all** free occurrences of x in S(x) with c.

Example

```
| 1.  ∀x ∃y (Adjoins(x, y) ∧ SameSize(y, x))
|—
| 2.  ∃y (Adjoins(b, y) ∧ SameSize(y, b))          ∀ Elim: 1
```

General Conditional Proof (∀ Intro)

```
  | | [c]  P(c)
  | |—
  | |                                   Where c does not occur outside
  | |  Q(c)                             the subproof where it is
▹ | ∀x (P(x) → Q(x))                    introduced.
```

There is an important bit of new notation here— $\boxed{c}$, the "boxed constant" at the beginning of the assumption line. This says, in effect, "let's call it c." To enter the boxed constant in Fitch, start a new subproof and click on the downward pointing triangle ▿. This will open a menu that lets you choose the constant you wish to use as a name for an arbitrary object. Your subproof will typically end with some sentence containing this constant.

In giving the justification for the universal generalization, we cite the entire subproof (as we do in the case of → **Intro**).

Notice that although c may not occur outside the subproof where it is introduced, it may occur again inside a subproof within the original subproof.

Universal Introduction (∀ Intro)

```
  | | [c]
  | |—
  | |                                   Where c does not occur outside
  | |                                   the subproof where it is
  | |  P(c)                             introduced.
▹ | ∀x P(x)
```

Remember, any time you set up a subproof for ∀ **Intro**, you **must** choose a "boxed constant" on the assumption line of the subproof, even if there is no sentence on the assumption line.

For practice, do the **You try it** on p. 344.

Default and generous uses of the ∀ rules

Default

∀ Elim: If you cite a universal generalization and apply the rule **∀ Elim**, Fitch will enter an instance of the generalization containing its best guess of the constant you want to replace the variable. If you are within a subproof containing a boxed constant, Fitch will use that constant. Otherwise, Fitch will use the alphabetically first constant not already in use in the sentence.

If you want to use a different constant, there are three ways to do it.

(1) The slow way: type the entire sentence in manually.

(2) A faster way: let Fitch guess, and then correct the sentence manually.

(3) The fastest way: suppose the quantifier is ∀x and you want to replace x with c. Cite the universal generalization, apply **∀ Elim**, and type in **`:x > c`**. What this says to Fitch is "replace x with c." Fitch will then enter an instance of the universal generalization with c plugged in for x.

∀ Intro: If you apply **∀ Intro** to a subproof containing a boxed constant (but no sentence) on the assumption line, Fitch will enter the universal generalization of the last line in the subproof. If there is a sentence on the assumption line, Fitch will enter the universal generalization of the conditional whose antecedent is the assumption sentence and whose consequent is the last line of the subproof.

Do the **You try it** on p. 345.

Generous

Fitch lets you remove (or introduce) more than one quantifier at a time.

∀ Elim: You can remove several quantifiers simultaneously. To go from ∀x ∀y SameCol(x, y) to SameCol(b, c), you may type in the new sentence manually, cite the universal generalization, and apply the rule. Or, cite the supporting sentence, apply the rule, and tell Fitch:

```
:x > b :y > c
```

This tells Fitch to replace x with b and y with c.

∀ Intro: You may also introduce more than one quantifier at a time. The trick here is to box **more than one constant** at the start of the subproof. Then, at the end of the subproof, Fitch will enter the appropriate universal generalization (of a conditional, if there is a sentence in the assumption line, otherwise of the last line in the subproof). You can use the "colon" notation as above to tell Fitch which variables to use. For example, if your boxed constants are b and c, then you tell Fitch to replace b with x and c with y by writing:

```
:b > x :c > y
```

The order in which you write these replacement instructions makes a difference. The instruction we wrote above tells Fitch not only to replace b with x and c with y, it also tells Fitch to put the quantifiers in the order ∀x∀y. If we wanted to make the same replacements (x for b and y for c), but have the quantifiers in the opposite order, ∀y∀x, we'd give the instruction this way:

`:c > y :b > x`

To see how this works, go to the Supplementary Exercises page and open the file Ch13Ex1. Open a new subproof with boxed constants b and c. (Choose them in this order.) Then add a new step after the assumption, cite the premise, and choose rule ∀ **Elim**. If you click Check Step at this point, Fitch will enter an instance of the premise, but with only the outer quantifier removed—it will replace x with b. (If you had chosen the boxed constants in the other order, $\boxed{c\ b}$, Fitch would have replaced x with c.)

If you want to use ∀ **Elim** to get SameCol(b, c) from the premise ∀x∀y SameCol(x, y) in just one step, you must specify the replacements using the colon notation, **`:x > b :y > c`**. Then cite the premise and apply ∀ **Elim**; Fitch will enter SameCol(b, c).

Next, end the subproof, cite it, and choose rule ∀ **Intro**. This time, be sure you specify not only the replacements but also the order in which the quantifiers are to appear. Since the conclusion is ∀y∀x SameCol(x, y), the instruction is

`:c > y :b > x`

Fitch will enter the desired conclusion, ∀y∀x SameCol(x, y). (Notice what happens to the conclusion if you write **`:b > x :c > y`**.) What we proved here, by the way, is that in a string of universal quantifiers, the order of the quantifiers is semantically irrelevant.

§ 13.2 Existential quantifier rules

Existential Introduction (∃ Intro)

| S(c)

▷ | ∃x S(x)

Here x stands for any variable, c stands for any individual constant, and S(c) stands for the result of replacing **all** free occurrences of x in S(x) with c. Note that there may be other occurrences of c in S(x).

Example 1

| 1. ∀y (Adjoins(b, y) → SameSize(y, b))

| 2. ∃x∀y (Adjoins(x, y) → SameSize(y, x)) ∃ **Intro**: 1

In example 1, there are no occurrences of b in the existential generalization we derived by using ∃ **Intro**. But now look at the next example:

Example 2

| 1. ∀y (Adjoins(b, y) → SameSize(y, b))

| 2. ∃x∀y (Adjoins(b, y) → SameSize(y, x)) ∃ **Intro**: 1

In example 2, there is an occurrence of b in the existential generalization we derived by using ∃ **Intro**. But that is perfectly all right. We require that the instance (line 1, in this case) have b wherever the generalization (line 2) has x, but **not conversely**.

Note carefully the wording of the rule. It talks about S(c) being the **result** of replacing free occurrences of x in S(x) with c, even though when we apply the rule, we tend to think of it differently—we **start out** with S(c) and then replace c with x, and attach ∃x.

So a good way to think about ∃ **Intro** is as follows: you start out with an instance of a general sentence, containing (perhaps) one or more occurrences of the constant, c. You then get to replace **one or more** of the occurrences of c (you don't have to replace all of the occurrences of c, although you may if you wish) with a variable x, and then attach the quantifier ∃x.

> The reason for the "perhaps" above is because of the possibility of *null* quantification (recall §10.4, pp. 280-82)—that is, a sentence in which an x quantifier contains no other occurrence of x within its scope. Strictly speaking, this peculiar inference, whose conclusion is a null quantification, is valid:
>
> $$\begin{array}{|l} \text{Cube}(b) \\ \hline \exists x\ \text{Cube}(b) \end{array}$$
>
> Therefore, the ∃ **Intro** rule had better allow us to draw it. And notice that its careful wording insures that it does just this. In this case, S(x) is Cube(b), which contains no occurrences of x at all. So S(c), which is "the result of replacing **all** free occurrences of x in S(x) with c," is also just our original sentence Cube(b). Then when we attach the quantifier ∃x, it becomes null, since there is no free occurrence of x in Cube(b) for the quantifier to bind. So the ∃ **Intro** rule permits this inference. If you're in doubt, try out this use of ∃ **Intro** in Fitch!

For a good illustration of the versatility of ∃ **Intro**, look at the file EI Varieties (on the Supplementary Exercises page). You'll see that there are four different conclusions that can be obtained by ∃ **Intro** from Likes(max, max), including the null quantification case described above.

Existential Elimination (∃ Elim)

$$\begin{array}{r|l} & \exists x\ S(x) \\ & \begin{array}{|l} \boxed{c}\ \ S(c) \\ \hline \\ Q \end{array} \\ \triangleright & Q \end{array}$$

Where c does not occur outside the subproof where it is introduced.

Here, again, the "boxed constant" indicates that we are choosing c as a name for some arbitrary object satisfying S(x). Note that the restriction that c may not occur outside the subproof means, in effect, that c cannot occur in the last line of the subproof, either. (∃ **Elim** instructs us to end the subproof and enter its last line, Q, as a new line, outside the subproof).

It is this restriction on ∃ **Elim** that blocks the fallacious inference we discussed in Chapter 12 [the pseudo-"proof" deducing ∃y ∀x Admires(x, y) from ∀x ∃y Admires(x, y)]. To see how this works, look at Exercise 13.17 on page 351 (which is a homework problem). There you will see that the mistake in this pseudo "proof" is an incorrect application of ∃ **Elim**.

Default and generous uses of the ∃ rules

Default

∃ Intro: If you cite a sentence and apply **∃ Intro**, Fitch will replace the alphabetically first name in the sentence with the first variable in the list (x, y, z, u, v, w) not already in the sentence.

∃ Elim: If you end a subproof, cite it, and apply **∃ Elim**, Fitch will enter the last line of the subproof on a new line (provided it does not contain any occurrences of the boxed constant).

Generous

∃ Intro: You can attach several existential quantifiers simultaneously. (Of course, they will have to be attached to the **front** of the sentence.) To go from SameCol(b, c) to ∃x ∃y SameCol(x, y), you may cite the supporting sentence, apply the rule, and tell Fitch:

```
:b > x :c > y
```

This tells Fitch to replace b with x and c with y. The instruction

```
:b > y :c > x
```

will produce the sentence ∃y ∃x SameCol(y, x). On the other hand, the instruction

```
:c > x :b > y
```

will produce the sentence ∃x ∃y SameCol(y, x).

∃ Elim: The trick here is to start a subproof with more than one boxed constant. If your subproof ends with a sentence, Q, that does not contain either of these constants, you may use **∃ Elim** to enter Q on the next line. (Q is typically, although not always, an existential generalization, i.e., an ∃-sentence.)

§ 13.3 Strategy and tactics

Working out strategy and tactics for a given proof is best accomplished in the following way:

1. **Try to understand what the FOL sentences mean.**
2. **Try to come up with an informal proof.**
3. **Convert your informal proof into a Fitch proof.**

The example on p. 352 gives you a good idea of how this works. For some hands-on experience, do the **You try it** on p. 356.

Now let's try working through one of the exercises. Open **Exercise 13.23**. First, figure out what the sentences mean. You'll come up with something like this:

Everything is either a cube or a dodecahedron.
Every cube is large.
Something is not large.
Therefore, there is a dodecahedron.

Next, try to develop an informal proof. It might run as follows:

> We know that at least one thing is not large. Let’s pick a thing that isn’t large and call it ***b***. Now since every cube is large and ***b*** isn’t large, we know that ***b*** is not a cube. But everything is either a cube or a dodecahedron, and ***b*** is not a cube. Therefore, ***b*** is a dodecahedron. So we have proved that there is a dodecahedron.

Now convert this into a Fitch proof. We have obviously used existential instantiation strategy, based on premise 3. So our proof will begin with a subproof containing the assumption ¬Large(b), with b as a boxed constant (see the file Proof 13.23a.prf). We will aim for Dodec(b), from which we can obtain ∃x Dodec(x) by **∃ Intro**. Then we can use **∃ Elim** to end the subproof and infer our conclusion ∃x Dodec(x).

Premise 2 tells us that all the cubes are large, so we need to infer the relevant instance concerning *b*. This means using **∀ Elim**, replacing x with b. We now have Cube(b) → Large(b) on one line and ¬Large(b) on another. Since we are allowed to use **Taut Con** with this problem, we may immediately infer ¬Cube(b). This leaves us in the position shown in Proof 13.23b.prf.

We now go back to premise 1 and apply **∀ Elim** again, with b replacing y, obtaining Cube(b) ∨ Dodec(b). And this, together with ¬Cube(b), gives us Dodec(b)—once again we use **Taut Con**. And that gives us our completed proof (see the file Proof 13.23.prf).

In Chapter 11 we translated some arguments into FOL and promised to return to them later. The one about Doris Day provides good practice in proof strategy.

The Doris Day principle (again)

Everybody loves a lover.

Doris is a lover.

Everybody loves Doris.
Doris loves everybody.

In FOL:

$$\forall x\ \forall y\ (\exists z\ \mathrm{Loves}(y, z) \to \mathrm{Loves}(x, y))$$

$$\exists z\ \mathrm{Loves}(\mathrm{doris}, z)$$

$$\forall x\ \mathrm{Loves}(x, \mathrm{doris})$$
$$\forall x\ \mathrm{Loves}(\mathrm{doris}, x)$$

Informal proof

> The first premise tells us that everybody loves a lover, so it follows that everybody loves Doris if she’s a lover. But the second premise tells us that Doris is a lover—so it follows that everybody loves her. That is our first conclusion. But if everybody loves Doris, then any randomly chosen person, ***a***, loves Doris. Since ***a*** loves Doris, it follows that ***a*** loves someone (i.e., ***a*** is a lover). But it follows from the first premise that if ***a*** is a lover, everybody loves ***a***. Now we have proved that ***a*** is a lover, so it follows that everybody loves ***a***. From this it follows that Doris loves ***a***. Since ***a*** was chosen at random, it follows that Doris loves everyone.

Formal proof

Now convert this into a Fitch proof (see Ch13Ex3.prf). Start a new subproof with boxed constant a. We will prove, first, that ***a*** loves Doris, and, second, that Doris loves ***a***. We apply ∀ **Elim** to the first premise twice, replacing x with a and y with Doris. The resulting sentence says that if Doris is a lover, then ***a*** loves Doris. We then use → **Elim** on that sentence together with the second premise to obtain Loves(a, doris). We then reapply ∀ **Elim** to the first premise, this time replacing x with Doris and y with a. The resulting sentence says that if ***a*** is a lover, then Doris loves ***a***. We then end the subproof and apply ∀ **Intro** twice, once to get ∀x Loves(x, doris) and a second time to get ∀x Loves(doris, x). Notice that the sentence (containing the boxed constant a) on which we are generalizing does not have to occur in the last line of the subproof. For the complete proof, see ProofCh13Ex3.prf.

§ 13.4 Soundness and completeness

We saw earlier (Chapter 8) that the restricted system $\mathcal{F}_T$ for propositional logic is both sound and complete. We now note that the full system $\mathcal{F}$ for FOL is also both sound and complete. Let us briefly review what soundness and completeness amount to.

Soundness

To say that a deductive system is sound is to say that all of the inferences it permits are (semantically) correct. That is, it never permits you to infer a falsehood from a truth. In the case of system $\mathcal{F}_T$, this meant that every conclusion that can be proved by the rules of $\mathcal{F}_T$ is a tautological consequence of its premises.

Completeness

To say that a deductive system is complete is to say that there is no (semantically) correct inference that it cannot prove. In the case of system $\mathcal{F}_T$, this meant that any tautological consequence of any set of premises can be proved (i.e., derived from those premises) by the rules of $\mathcal{F}_T$.

Obviously, the notions of soundness and completeness of the full system $\mathcal{F}$ are exactly analogous to those for the restricted system $\mathcal{F}_T$. There are only two differences:

1. In place of *tautological consequence* (for system $\mathcal{F}_T$) we now have *first-order consequence* (for system $\mathcal{F}$). If you are unclear on the notion of first-order consequence, review §10.2.
2. In place of the notion of *provability in system* $\mathcal{F}_T$, we now have *provability in system* $\mathcal{F}$. Just as we used the "turnstile" notation, $\vdash_T$, to express the former notion, we now write simply $\vdash$ to express the latter. Here is what the difference amounts to:

   $P_1, \ldots, P_n \vdash_T S$ means that S can be proved, from premises $P_1, \ldots, P_n$, using **only the truth-functional rules** (i.e., the rules of $\mathcal{F}_T$).

   $P_1, \ldots, P_n \vdash S$ means that S can be proved, from premises $P_1, \ldots, P_n$, using **any of the rules** of $\mathcal{F}$.

We can now state the soundness and completeness theorems for $\mathcal{F}$:

The **Soundness Theorem** for $\mathcal{F}$: If $P_1, \ldots, P_n \vdash S$, then S is a first-order consequence of $P_1, \ldots, P_n$.

The **Completeness Theorem** for $\mathcal{F}$: If a sentence S is a first-order consequence of $P_1, \ldots, P_n$, then $P_1, \ldots, P_n \vdash S$.

Recall that in propositional logic, there are soundness and completeness corollaries that relate the notions of *tautology* and *provability in* $\mathcal{F}_T$. There are analogous corollaries for the full system $\mathcal{F}$. These concern the relation between proofs **without premises**, on the one hand, and first-order validities, on the other. (Remember that $\vdash S$ means that there is a proof without premises of S in system $\mathcal{F}$.)

Soundness Corollary: If $\vdash S$, then S is a first-order validity.

The soundness corollary tells us that every sentence of FOL that can be proved without premises in system $\mathcal{F}$ is a first-order validity, that is, a logical truth of FOL.

Completeness Corollary: If S is a first-order validity, then $\vdash S$.

The completeness corollary tells us that every sentence of FOL that is a first-order validity, that is, a logical truth of FOL, can be proved without premises in system $\mathcal{F}$.

§ 13.5 Some review exercises

This section contains 19 problems (of which 8 are assigned on problem set H_{19}). One of these is to prove the famous "Drinking Theorem" (13.51★★). The theorem is: $\exists x(P(x) \rightarrow \forall y\, P(y))$. If you read P(x) as '*x* drinks', you get the theorem:

There are some people such that, if they drink, everyone drinks.

There are two separate issues here: (1) to see that this is a logical truth (indeed, a first-order validity), and (2) to figure out how to prove it. Doing (1) first is extremely helpful before tackling (2).

Since this is a hard problem, here are some hints:

1. Who are these people who are such that, if they drink, everyone drinks?
2. Remember that conditionals are treated by Tarski's World as just abbreviations of disjunctions. That is:

   $$\text{Drinks}(x) \rightarrow \forall y\, \text{Drinks}(y)$$

   is just an abbreviation of:

   $$\neg\text{Drinks}(x) \lor \forall y\, \text{Drinks}(y)$$
3. So our question (1) is really: who are these people who are such that, either they don't drink, or everyone drinks? And the answer to this is easy: they are the **non-drinkers**.
4. But, what if there are no such people, i.e., no non-drinkers? In that case, everyone drinks. But that makes the right disjunct true.

5. So if there are non-drinkers, they falsify the antecedent, making the conditional true of them; and if there are no non-drinkers, that means that everyone's a drinker, making the consequent true, and thereby the whole conditional true of every *x*, and hence true of at least one thing. So the existential generalization is true in any case. So it's a first-order validity.

[You can check this out by writing the sentence ∃x (Cube(x) → ∀y Cube(y)) in Tarski's World and then trying to construct a world it which it is false. You'll see that it's impossible to do so. Try playing the game, committed to false. If your world contains all cubes, Tarski will ask you to find something that falsifies Cube(y), and you will fail. If your world contains at least one non-cube, Tarski will pick a non-cube, call it n_1, and show that you are committed to the truth of Cube(n_1).]

This suggests a proof strategy: proof by cases. Case (1): there are non-drinkers; case (2) there are no non-drinkers. You may use **Taut Con** in your proof to enter the disjunction of (1) and (2), which is an instance of excluded middle.

More exercises

The Doris Day principle (one last time)

We've already deduced some of the logical consequences of the principle *everybody loves a lover*. We'll now try to prove that the principle is false, using a couple of empirical premises that are, I think, uncontroversial:

| Madonna loves herself.
| Rush Limbaugh does not love Hillary Clinton.
|—
| It is not true that everybody loves a lover.

Here's the argument in FOL:

| Loves(madonna, madonna)
| ¬Loves(rush, hillary)
|—
| ¬∀x ∀y (∃z Loves(y, z) → Loves(x, y))

Informal Proof:

We'll do this by indirect proof. Suppose that everybody loves a lover. Now, Madonna loves herself, and so she loves someone (by ∃ **Intro**). That means that she's a lover. So everybody loves her (from our indirect proof assumption, by ∀ **Elim**). In particular, Hillary loves her (another application of ∀ **Elim**). So Hillary loves someone (by ∃ **Intro**), and that makes her a lover. So everybody loves her (yet another application of ∀ **Elim**). Therefore (by ∀ **Elim**!), Rush loves her, and that contradicts the second premise. So, by **¬Intro**, it follows that the Doris Day principle is false.

We will now implement this strategy in a Fitch proof. You can find it in Doris Day's Argument on the Supplementary Exercises page.

Stage 1

Start a new subproof and assume ∀x ∀y (∃z Loves(y, z) → Loves(x, y)), for proof by contradiction. We intend to obtain our contradiction by proving Loves(rush, hillary), which contradicts premise (2), so we will enter that step (without justification at this point), use it to obtain ⊥, and end the subproof. To see where the proof stands at this stage, open Proof Doris Day Stage 1.

Stage 2

We need to prove that Madonna is a lover, so we apply **∃ Intro** to the first premise, being careful to substitute the variable z for the second occurrence of 'Madonna' only. (We want to prove that Madonna loves someone, not that someone loves herself!) We then apply **∀ Elim** to our assumption, replacing x with hillary and y with madonna. That enables us to use **→ Elim** to infer Loves(hillary, madonna). To see where the proof stands now, look at Proof Doris Day Stage 2.

Stage 3

Next we apply **∃ Intro** to Loves(hillary, madonna) to prove that Hillary is a lover (typing `:madonna > z`). Then we go back to the Doris Day principle (our subproof assumption) and obtain another instance by **∀ Elim**. This time we want to substitute rush for x and hillary for y. The resulting sentence tells us that Rush loves Hillary if Hillary is a lover. So by **→ Elim** we can obtain Loves(rush, hillary), which contradicts the second premise. This completes the proof — see Proof Doris Day's Argument.

This proof may not have convinced you that the Doris Day principle is false, since you may object that the argument is unsound. That is, you may think that one of the "empirical" premises we used is false. So let's look at an alternative version in which there are no names.

| Everybody loves a lover.
|—
| If there is even one lover, then everyone loves everyone.

The conclusion here seems clearly false: the antecedent is true (there exists at least one lover) and the consequent is false (there is at least one pair of people, *x* and *y*, such that *x* doesn't love *y*). So the Doris Day principle must be false if it has this blatantly false consequence.

Exercise: show that the Doris Day principle does have this consequence by proving the following in Fitch:

| ∀x ∀y (∃z Loves(y, z) → Loves(x, y))
|—
| ∃x ∃y Loves(x, y) → ∀x ∀y Loves(x, y)

You can find this exercise as Doris Day 2.prf on the Supplementary Exercises page.

Dangerfield's argument

You'll remember this one from the notes to Chapter 11, where we used it as a translation problem. You can find Dangerfield's Argument on the Supplementary Exercises page. Here it is:

| ∀x (¬Respects(x, x) → ∀y ¬Respects(y, x))
| ∀x ∀y (¬Respects(x, y) → ¬Hires(x, y))
|—
| ∀x (∀y¬Respects(x, y) → ∀y ¬Hires(y, x))

Informal Proof:

> We'll use the method of general conditional proof. Our conclusion is that anyone who respects no one will not be hired by anyone. So, let *d* be a person who respects no one; we will prove that no one will hire *d*.
>
> To do this, we will let *c* be any arbitrary person; we will prove that *c* will not hire *d*. Note that it follows from the first premise (by ∀ **Elim**) that (1) if *d* doesn't respect himself, then no one respects *d*. But we have assumed that *d* respects no one, and this logically implies that (2) *d* does not respect himself. And (1) and (2) imply (by → **Elim**) that no one respects *d*. Hence (by ∀ **Elim**), *c* does not respect *d*. But we may infer from the second premise that if *c* doesn't respect *d*, *c* will not hire *d*. Hence, *c* will not hire *d*. But *c* was arbitrary, so no one will hire *d*. Our conclusion then follows by general conditional proof.

We will now implement this strategy in a Fitch proof.

Stage 1

We begin by starting a subproof assuming ∀y¬Respects(d, y), with boxed constant d. The last line of the subproof will be ∀y ¬Hires(y, d). We then apply ∀ **Intro** to obtain the conclusion. To see where the proof stands at this stage, open Proof Dangerfield Stage 1.

Stage 2

The next step is to start a new subproof, within the first, with c as a boxed constant, but no sentence in the assumption line. Our goal for this subproof is ¬Hires(c, d). First we apply ∀ **Elim** twice, to the first premise and to line 3, with d replacing x. That sets up an application of → **Elim** to obtain ∀y¬Respects(y, d). We can then apply ∀ **Elim** to this sentence, with c replacing y. To see where the proof stands now, look at Proof Dangerfield Stage 2.

Stage 3

We now turn our attention to the second premise. We apply ∀ **Elim** again, with c replacing x and d replacing y. (You can do this in one step by specifying the replacements this way: `:x > c :y > d`. But the fastest way is in two steps, with Fitch supplying the right substitution at each step by default.) One more application of → **Elim** gives us ¬Hires(c, d), which was the goal sentence for this subproof. This completes the proof — see Proof Dangerfield's Argument.

Leibniz's argument

The philosopher Leibniz (1646-1716) wrote, "I define a good man as one who loves all men." He also proposed to "deduce all the theorems of equity and justice" from this and a few other basic definitions. One such theorem might be that *there is a man who loves all good men*. We will construct a proof in Fitch to show that this theorem is, indeed, a FO-consequence of Leibniz's definition of a good man.

> [By 'man' here Leibniz clearly meant 'person'; so we'll treat the domain of discourse as *persons*, which will simplify the translation process.]

We'll treat the definition as a universally quantified biconditional:

$$\forall x\ (\mathrm{Good}(x) \leftrightarrow \forall y\ \mathrm{Loves}(x, y))$$

The translation into FOL of the theorem to be proved is straightforward:

$$\exists y\ \forall x\ (\mathrm{Good}(x) \rightarrow \mathrm{Loves}(y, x))$$

Open the file Leibniz's Argument (on the Supplementary Exercises page) and try to construct the proof. For a hint on how to start, open Proof Leibniz Start. As you can see, the strategy is **proof by cases**, with the two cases being (1) there are some good persons, and (2) there are not any good persons. Since the disjunction of (1) and (2) is an instance of excluded middle, we can obtain it by **Taut Con**.

The problem now is to show that in either case, we can deduce the conclusion. The best way to do this is to sketch an informal proof. Then you should be able to model your Fitch proof on this.

Informal Proof:

Case (1): Suppose there is at least one good person, *b*, for example. Then, according to the definition, *b* loves all persons. Since *b* loves all persons, *b* clearly loves all the good ones, too. So someone loves all good persons.

Case (2): Suppose, on the other hand, that there are no good persons. That means that any universal generalization you make about *all good persons* will be vacuously true. For example, *all good persons are loved by d*. (Here, *d* can be anyone you like, and need not be an arbitrarily chosen person.) That is, *d* loves all good persons. Hence, someone loves all good persons.

Now the problem is to come up with a Fitch implementation of this strategy. We will do this in stages.

Stage 1

Case (1) has an ∃ sentence as its assumption line, so we use existential instantiation strategy: assume Good(b), which is an instance of ∃x Good(x), deduce the desired conclusion, and apply **∃ Elim**.

Case (2) is more complicated. Since we are attempting to prove an ∃∀ sentence, the strategy is to come up with an instance of the ∃ sentence, and then use **∃ Intro**. But the instance in question will be a general conditional sentence, namely, ∀x (Good(x) → Loves(d, x)). So the strategy here is to use general conditional proof: start a new subproof assuming Good(c), deduce Loves(d, c), and apply **∀ Intro**. Note that to employ this strategy, one must introduce c as a boxed constant on the assumption line. But d does not have to be a boxed constant—it can be any name you like.

To see where the proof stands at this stage, open Proof Leibniz Stage 1.

Stage 2

Case (1): We have assumed Good(b), and wish to prove that *b* loves all good persons. So, we use **∀ Intro** strategy: assume that arbitrary person *a* is good, and prove that *b* loves *a*.

Case (2) has ¬∃x Good(x) as its assumption line, and our subproof within that assumes Good(c). From these we should be able to get a contradiction. Then we can use **⊥ Elim** to get Loves(d, c).

To see the proof at this stage, look at Proof Leibniz Stage 2.

Stage 3

Case (1): We apply ∀ **Elim**, obtaining Good(b) ↔ ∀y Loves(b, y). We can then detach the right side of this biconditional and apply ∀ **Elim** again, obtaining Loves(b, a). We then end the subproof and apply ∀ **Intro**.

Case (2): From our assumption Good(c) we can obtain ∃x Good(x) by ∃ **Intro**. This gives us the contradiction we were looking for.

Putting all the pieces together, we get the completed proof, which you can see by opening Proof Leibniz's Argument.