3.3. Negations as premises

3.3.0. Overview

A second group of rules for negation interchanges the roles of an affirmative sentence and its negation.

3.3.1. Indirect proof

The basic principles for negation describe its role as a premise only in *reductio* arguments but a *reductio* is always available as an argument of last resort.

3.3.2. Using lemmas to complete *reductio*s

The role negative resources play will be to contradict other sentences; since what they contradict must often be introduced as a lemma, a use of lemmas is built into the rule for exploiting negative resources.

3.3.3. More examples

These new rules permit some new approaches to entailments that could be established using the last section's rule; but they also support some further entailments.

Glen Helman 03 Aug 2010

3.3.1. Indirect proof

The last section pursued consequences of the law for negation as a conclusion. The rules of this section will implement the other basic law for negation, the law for it as a premise:

$$\Gamma, \neg\, \varphi \vDash \bot \text{ if and only if } \Gamma \vDash \varphi$$

This says that a negation is $\neg\, \varphi$ inconsistent with a set Γ if and only if the sentence φ is entailed by that set.

There are several lessons we can learn from this law. First, the only-if-statement tells us that negative conclusions are not the only ones that can be established by way of *reductio* arguments, for it says that an entailment $\Gamma \vDash \varphi$ must hold if the *reductio* $\Gamma, \neg\, \varphi \vDash \bot$ holds. Further, the if-statement tells us in part that such an approach is safe, that the *reductio* is valid whenever the argument we wish to support by it is valid. But if-statement tells us more. Notice that φ is just the sort of resource that would enable us to complete a *reductio* that has $\neg\, \varphi$ as a premise. The if-claim above tells us that, if a *reductio* with $\neg\, \varphi$ as a premise can be completed at all, we would be able to validly conclude φ as a lemma—and that we could do so without using $\neg\, \varphi$ itself as a premise. This further lesson will provide the basis for exploiting negative resources. However, its full application depends on the broader use of *reductio* arguments supported by the other two lessons, and that is what we will consider first.

Here is an example of this broader use of *reductio*s. If we take No one is home to be the negation $\neg$ someone is home, the law for negation as a premise says we can rest the validity of the left-hand argument below on the validity of the right-hand argument.

If no one was out, the car was in the driveway	If no one was out, the car was in the driveway
The car wasn't in the driveway	The car wasn't in the driveway
	No one was out
Someone was out	$\bot$

The right-hand argument depends in part on the logical properties of if; but, as far as negation is concerned, it depends on only the fact that a sentence and its negation are mutually exclusive.

The fact that they are mutually exclusive also supports the entailment $\neg\, \neg\, \varphi, \neg\, \varphi \vDash \bot$. If we apply the law for negation as a premise to this entailment, we get the principle $\neg\, \neg\, \varphi \vDash \varphi$. Moreover, the latter principle can be combined with the law for negation as a conclusion to establish the law for negation as a premise. So the further logical properties of negation that are captured by the law for negation as a premise can be summarized in the principle that a double negation entails the corresponding positive claim.

This principle is one that was rejected by Brouwer in his intuitionistic mathematics. And one of his chief reasons for rejecting it was that it would allow us to draw a conclusion of the form Something has the property P when the corresponding claim Nothing has the property P was inconsistent with our premises, and that is just the sort of thing that was done in the example above. His concern with this is that it would enable us to conclude Something has the property P in cases where we were unable, even in principle, to provide an actual example of a thing with that property P. Brouwer did not object to such an argument in ordinary reasoning about the physical world (like the example above); but he held that, in reasoning concerning infinite mathematical structures, we were not reasoning about an independently existing realm of objects but instead about procedures for constructing abstract objects and that we had no business claiming the existence of such objects without having procedures enabling us to construct them. Brouwer's concerns may not lead you to question the law for negation as a premise; but they highlight the indirectness of supporting a positive conclusion by an argument concerning its denial. This aspect of these arguments is reflected in a common term for them, *indirect proofs*.

Although we will employ indirect proofs, we will need them for only a limited range of conclusions. We have other ways of planning for a goal that is a conjunction or a negation, and we can simply close a gap whose goal is ⊤. We will not adopt any rule to plan for the goal ⊥ of a *reductio argument*. At the moment, that leaves only unanalyzed components; and, until the last chapter (where we consider the logical properties of something), those are the only goals for which we will use indirect proofs. We have often closed gaps whose goals are atomic, so we know that indirect proof is not always necessary even for such goals, but it will serve us as a last resort.

In chapter 6, we will begin to analyze sentences into components that are not sentences, and we will still use indirect proof for goals that are analyzed in that way. In anticipation of this, we will use the term *atomic* for the kind of goals to which we will apply indirect proof; and we will refer to other sentences as *non-atomic*. Until chapter 6, any sentence we analyze will be a compound formed by applying a connective to one or more sentences, so, for the time being, the atomic sentences will be the unanalyzed sentences. ⊤ and ⊥ count as non-atomic since identifying them as Tautology and Absurdity counts as an analysis of their logical form. As a result, for the time being, the atomic sentences will be simple letters, and all other sentences will be non-atomic.

In tree-form and sequent proofs, this new form of argument will look like RAA or negation as a conclusion except that the positions of φ and ¬ φ will be reversed. The same is true of the rule implementing indirect proofs in derivations, but we will choose a name that reflects its rather different role and call it *Indirect Proof* (IP). It takes the following form:

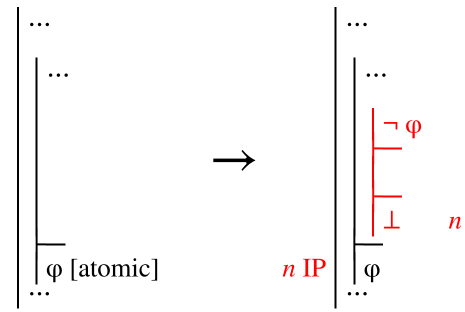

Fig. 3.3.1-1. Developing a derivation by planning for an atomic sentence at stage *n*.

Here is an example, which is related to the argument at the beginning of 3.2.2.

	$\neg((A \wedge B) \wedge \neg C)$	(4)
	A	(2)
	B	(2)
	$\neg C$	(3)
2 Adj	$A \wedge B$	X,(3)
3 Adj	$(A \wedge B) \wedge \neg C$	X,(4)
	●	
4 Nc	$\bot$	1
1 IP	C	

This example adds to the premise Ann and Bill were not both home without the car being in the driveway further premises telling us that each of Ann and Bill was home, and we conclude that the car was in the driveway. Although the initial premises and conclusion differ from those of the argument in 3.2.2, the *reductio* argument that is set up at stage 1 here has the same resources as the *reductio* set up at stage 3 in the derivation for the argument of 3.2.2 that was given at the end of 3.2.3.

The rule IP is easily seen to be strict and safe, but we need to be more careful in assessing the decisiveness of a system using it. We will consider this question most fully in 3.4.1, but we can see the issue in outline now. Since IP introduces a sentence with one more connective than the goal it plans for, it does not reduce the quantity we have used to assess the progressiveness of other rules. But IP can be seen to be progressive nonetheless if we look progressiveness in a slightly different way. We will treat both atomic sentences and their negations as equally basic when they are resources: neither sort of resource will be exploited. And, as was noted above, we will treat ⊥ as the basic form of goal, the only one without a corresponding planning rule. Thus IP leaves us with a goal that requires no planning, and it introduces no resources that need to be exploited further. This suggests a way of looking at the distance of a proximate argument from a dead end that would allow us to say that IP reduces this quantity. This way of looking at distance from a dead end is a departure from counting connectives but only a small departure. We may count connectives except for the case of those sentences that are never exploited or planned for—i.e., atomic and negated atomic sentences as resources and ⊥ as a goal—and these sentences will be given a lower degree than all other sentences, no matter how few connectives those sentences contain.

Although IP introduces a resource that needs no exploitation, this is not to say that applying IP will eliminate the need for further exploitations; indeed, since negated compounds will be exploited only in *reductio* arguments, we will often be in a position to exploit such resources only after we have used IP. The rule it can put us in the position to use is the one we will consider next.

3.3.2. Using lemmas to complete *reductio*s

Now that we have IP, we are in a position to provide a proof for any argument whose validity depends only on the properties of ⊤, ⊥, conjunction, and negation. However, to do this using only the rules we have so far, we would often need to use LFR—or, in simpler cases, Adj—to make use of negative resources. This poses no problems when we construct derivations for valid arguments, but it makes it difficult to show that an argument is not valid. LFR does not itself exploit resources, so negated compounds remain as active resources until a gap is closed. In order to count an open gap as having reached a dead end, we would need some description of the conditions under which LFR had been used often enough. Such a description could certainly be given; and, in the last two chapters, we will need to take an analogous approach in the case of one of the rules for quantifiers. But, in the case of negation, it is possible to keep track of the use of resources by way of a genuine exploitation rule, which will eliminate the need to use LFR and Adj.

The basis for this approach is one of the lessons drawn from the law of negation as a premise: if a *reductio* that has ¬ φ as a premise is valid, then φ must be a valid conclusion from the premises other than ¬ φ. That is, Γ, ¬ φ ⊨ ⊥ only if Γ ⊨ φ. Now φ is just the lemma we need in order to use the premise ¬ φ to reach the goal ⊥ and complete the *reductio*. The fact that our goal is ⊥ tells us that it is safe to set φ as a goal, and the law for negation as a premise tells us that it is safe to drop ¬ φ from the active resources of the gap in which we establish the lemma φ. That is, we can use a negation ¬ φ to complete any *reductio* argument, so we can exploit a negated compound whenever our goal is ⊥.

We will call the rule that implements these ideas *Completing a Reductio* (CR).

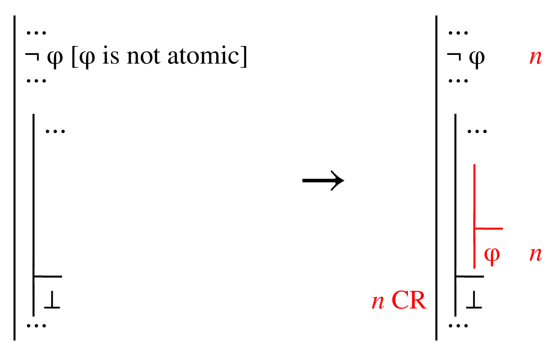

Fig. 3.3.2-1. Developing a derivation by exploiting a negated compound at stage *n*.

The motivation for CR lies in its use to exploit the negations of non-atomic sentences, since we can arrange things so that the negations of atomic sentences remain active forever. In fact, we must limit the use of CR to the negations of non-atomic sentences. It is sound and safe in the case of negations of atomic sentences, but it would not be progressive in that case because it would allow us to go around in circles. Both IP and CR carry us between gaps whose proximate arguments have the forms Γ, ¬ φ / ⊥ and Γ / φ; but they carry us in opposite directions, so, if there is any overlap in the sentences φ to which they apply, a derivation could move back and forth between the two arguments forever. We block such circles by limiting IP to cases where φ is atomic and limiting CR to cases where φ is non-atomic.

One way of understanding the role of CR is to compare it with a use of LFR, where the recourse to lemma is more explicit. Below are two derivations for the argument that was used as an illustration in the last subsection. The one on the left uses CR and the one on the right uses LFR:

	¬ ((A ∧ B) ∧ ¬ C)	2
	A	(3)
	B	(3)
	¬ C	(4)
3 Adj	A ∧ B	X,(4)
4 Adj	(A ∧ B) ∧ ¬ C	X,(5)
	●	
5 QED	(A ∧ B) ∧ ¬ C	2
2 CR	⊥	1
1 IP	C	

	¬ ((A ∧ B) ∧ ¬ C)	(6)
	A	(3)
	B	(3)
	¬ C	(4)
3 Adj	A ∧ B	X,(4)
4 Adj	(A ∧ B) ∧ ¬ C	X,(5)
	●	
5 QED	(A ∧ B) ∧ ¬ C	2
	(A ∧ B) ∧ ¬ C	(6)
	●	
6 Nc	⊥	2
2 LFR	⊥	1
1 IP	C	

Notice that the gap resulting from CR on the left is identical to, and filled in the same way as, the first of the two gaps introduced by LFR on the right. We know in advance that the second of these gaps will close because the denial of its supposition is one or our active resources. Indeed the point of choosing (A ∧ B) ∧ ¬ C as the lemma in LFA is to combine it with the resource ¬ ((A ∧ B) ∧ ¬ C) to reach ⊥ and complete the *reductio*. That is, LFA on the right is part of a plan to use the first premise. What is new in CR is the claim that this resource need not be used further in developing the derivation and may be dropped from its active resources. And this makes CR clearly progressive in a way that LFR is not.

3.3.3. More examples

Here is an English argument whose derivation exhibits all of the rules for negation:

Ann's proposal wasn't unfunded without
Bill's and Carol's each being funded
Bill's proposal was not funded

Ann's proposal was funded

And here is the derivation:

	¬ (¬ A ∧ ¬ (B ∧ C))	2
	¬ B	(7)
	¬ A	(4)
	●	
4 QED	¬ A	3
	B ∧ C	6
6 Ext	B	(7)
6 Ext	C	
	●	
7 Nc	⊥	5
5 RAA	¬ (B ∧ C)	3
3 Cnj	¬ A ∧ ¬ (B ∧ C)	2
2 CR	⊥	1
1 IP	A	

The rules of this section are used at the first two stages, and the rules of 3.2 are in the course of reaching the goal introduced by CR. One alternative approach would be to introduce ¬ (B ∧ C) as a lemma at the second stage using LFR. Combined with a use of Adj to add ¬ A ∧ ¬ (B ∧ C) as a resource, it would produce a simpler derivation but one that requires foresight to discover.

In the absence of the rules of this section, the exercise **2d** of 3.2.x required use of LFR. Here are two derivations for the argument of that exercise which use CR instead but differ in the choice of the premise to be exploited by this rule.

	¬ (A ∧ B)	3
	¬ (C ∧ ¬ B)	(8)
	A ∧ C	2
2 Ext	A	(5)
2 Ext	C	(7)
	●	
5 QED	A	4
	¬ B	(7)
7 Adj	C ∧ ¬ B	X,(8)
	●	
8 Nc	⊥	6
6 IP	B	4
4 Cnj	A ∧ B	3
3 CR	⊥	1
1 RAA	¬ (A ∧ C)	

	¬ (A ∧ B)	(8)
	¬ (C ∧ ¬ B)	3
	A ∧ C	2
2 Ext	A	(7)
2 Ext	C	(5)
	●	
5 QED	C	4
	B	(7)
7 Adj	A ∧ B	X,(8)
	●	
8 Nc	⊥	6
6 RAA	¬ B	4
4 Cnj	C ∧ ¬ B	3
3 CR	⊥	1
1 RAA	¬ (A ∧ C)	

These derivations have the same number of stages as the answer in 3.2.xa for **2d**, but their scope lines are nested one deeper. Each of the arguments completing the gaps set up by LFR in the earlier derivation appears in one of these derivations, but we arrive at these arguments in a different way.

It is possible to dispense with Adj in the derivations above and exploit both premises by CR. This leads to a derivation with two more stages and scope lines that are nested more deeply. What we get in return for that increased complexity is direction in how to complete the derivation. In effect, all the thinking required to identify appropriate lemmas is done on paper. We will look at this third approach to the example in 3.5, where we consider how the rules guide the search for derivations.

3.3.s. Summary

1 The law for negation as a premise tells us two things about entailment. It tells us first that a conclusion is valid if and only if the denial of that conclusion can be reduced to absurdity given the premises. This is the principle of indirect proof; it is closely tied to the entailment ¬ ¬ φ ⊨ φ (and is subject to the same concerns as is that entailment). We have no need for this principle except in the case of unanalyzed components, which we will begin to call atomic sentences. And, for reasons noted later, we need to limit the use of the rule Indirect Proof (IP) to such conclusions.

2 Another lesson we can draw from the law for negation as a premise is that a *reductio* argument with a negative premise ¬ φ is valid if and only if the sentence φ is entailed by whatever other premises there are. This tells us that φ can be safely introduced as a lemma even if we drop ¬ φ from our active resources. The rule implementing this idea, Completing a *Reductio* (CR) serves as our rule for exploiting negative resources. It applies only to *reductio* arguments but the availability of IP insures that any gap will eventually turn into a gap in a *reductio* argument (unless it closes before that point). Since CR, by dropping a resource ¬ φ and adding a goal φ has an effect opposite to that of IP, we must apply them to different sentences φ to avoid going in circles. So, just as IP is limited to atomic sentences, CR is limited to negations of non-atomic sentences.

3 The rule CR can lead us to set as goals any lemmas we need in order to use negations in completing *reductio* arguments. It therefore eliminates any need for LFR. The rule Adj is also no longer needed (though still sometimes useful) since the rules CR and Cnj will lead us to identify and prove any lemma that Adj would introduce. Indeed, derivations for arguments involving conjunction can now be constructed by simply letting the rules guide us.

3.3.x. Exercise questions

Use derivations to establish each of the claims of entailment shown below. You can maximize your practice in the use of CR by avoiding LFR and using Adj only in cases where the goal is a conjunction.

1. ¬ (A ∧ ¬ B), A ⊨ B

2. J ∧ ¬ (J ∧ ¬ C) ⊨ J ∧ C (see exercise **1j** of 3.1.x)

3. ¬ (¬ (A ∧ B) ∧ C), ¬ A ⊨ ¬ C

4. ¬ (A ∧ ¬ (B ∧ C)) ⊨ ¬ (A ∧ ¬ B)

5. ¬ (A ∧ ¬ B), ¬ (B ∧ ¬ C) ⊨ ¬ (A ∧ ¬ C)

6. ¬ (A ∧ ¬ B), ¬ (A ∧ ¬ C) ⊨ ¬ (A ∧ ¬ (B ∧ C))

For more exercises, use the exercise machine.

Glen Helman 03 Aug 2010

3.3.xa. Exercise answers

1.

```
         │¬ (A ∧ ¬ B)     2
         │A               (3)
         ├──
         ││¬ B            (3)
         │├──
3 Adj    │││A ∧ ¬ B       X,(4)
         │││●
         ││├──
4 QED    │││A ∧ ¬ B       2
         │├──
2 CR     ││⊥              1
         ├──
1 IP     │B
```

2.

```
         │J ∧ ¬ (J ∧ ¬ C)   1
         ├──
1 Ext    │J                 (3),(6)
1 Ext    │¬ (J ∧ ¬ C)       5
         ││●
         │├──
3 QED    ││J                2
         │││¬ C             (6)
         ││├──
6 Adj    ││││J ∧ ¬ C        X,(7)
         ││││●
         │││├──
7 QED    ││││J ∧ ¬ C        5
         ││├──
5 CR     │││⊥               4
         │├──
4 IP     ││C                2
         ├──
2 Cnj    │J ∧ C
```

3.

```
          │ ¬ (¬ (A ∧ B) ∧ C)      2
          │ ¬ A                    (7)
          ├───
          │ │ C                    (4)
          │ ├───
          │ │ │ │ │ A ∧ B          6
          │ │ │ │ ├───
6 Ext     │ │ │ │ │ A              (7)
6 Ext     │ │ │ │ │ B
          │ │ │ │ │ ●
          │ │ │ │ ├───
7 Nc      │ │ │ │ │ ⊥            5
          │ │ │ ├───
5 RAA     │ │ │ │ ¬ (A ∧ B)        3
          │ │ │
          │ │ │ │ ●
          │ │ │ ├───
4 QED     │ │ │ │ C                3
          │ │ ├───
3 Cnj     │ │ │ ¬ (A ∧ B) ∧ C      2
          │ ├───
2 CR      │ │ ⊥                    1
          ├───
1 RAA     │ ¬ C
```

4.

```
          │ ¬ (A ∧ ¬ (B ∧ C))      3
          ├───
          │ │ A ∧ ¬ B              2
          │ ├───
2 Ext     │ │ A                    (5)
2 Ext     │ │ ¬ B                  (8)
          │ │
          │ │ │ │ ●
          │ │ │ ├───
5 QED     │ │ │ │ A                4
          │ │ │
          │ │ │ │ │ B ∧ C          7
          │ │ │ │ ├───
7 Ext     │ │ │ │ │ B              (8)
7 Ext     │ │ │ │ │ C
          │ │ │ │ │ ●
          │ │ │ │ ├───
8 Nc      │ │ │ │ │ ⊥            6
          │ │ │ ├───
6 RAA     │ │ │ │ ¬ (B ∧ C)        4
          │ │ ├───
4 Cnj     │ │ │ A ∧ ¬ (B ∧ C)      3
          │ ├───
3 CR      │ │ ⊥                    1
          ├───
1 RAA     │ ¬ (A ∧ ¬ B)
```

5.

```
          │ ¬ (A ∧ ¬ B)            3
          │ ¬ (B ∧ ¬ C)            7
          ├───
          │ │ A ∧ ¬ C              2
          │ ├───
2 Ext     │ │ A                    (5)
2 Ext     │ │ ¬ C                  (8)
          │ │
          │ │ │ │ ●
          │ │ │ ├───
5 QED     │ │ │ │ A                4
          │ │ │
          │ │ │ │ │ B            (8)
          │ │ │ │ ├───
8 Adj     │ │ │ │ │ │ B ∧ ¬ C    X,(9)
          │ │ │ │ │ │ ●
          │ │ │ │ │ ├───
9 QED     │ │ │ │ │ │ B ∧ ¬ C    7
          │ │ │ │ ├───
7 CR      │ │ │ │ │ ⊥            6
          │ │ │ ├───
6 RAA     │ │ │ │ ¬ B            4
          │ │ ├───
4 Cnj     │ │ │ A ∧ ¬ B          3
          │ ├───
3 CR      │ │ ⊥                  1
          ├───
1 RAA     │ ¬ (A ∧ ¬ C)
```

6.

	│ ¬ (A ∧ ¬ B)	3
	│ ¬ (A ∧ ¬ C)	7
	│ │ A ∧ ¬ (B ∧ C)	2
2 Ext	│ │ A	(5),(9)
2 Ext	│ │ ¬ (B ∧ C)	10
	│ │ │ │ ●	
5 QED	│ │ │ │ A	4
	│ │ │ │ │ B	(11)
	│ │ │ │ │ │ │ ●	
9 QED	│ │ │ │ │ │ │ A	8
	│ │ │ │ │ │ │ │ C	(11)
11 Adj	│ │ │ │ │ │ │ │ │ B ∧ C	X,(12)
	│ │ │ │ │ │ │ │ │ ●	
12 QED	│ │ │ │ │ │ │ │ │ B ∧ C	10
10 CR	│ │ │ │ │ │ │ │ ⊥	9
9 RAA	│ │ │ │ │ │ │ ¬ C	8
8 Cnj	│ │ │ │ │ │ A ∧ ¬ C	7
7 CR	│ │ │ │ │ ⊥	6
6 RAA	│ │ │ │ ¬ B	4
4 Cnj	│ │ │ A ∧ ¬ B	3
3 CR	│ │ ⊥	1
1 RAA	│ ¬ (A ∧ ¬ (B ∧ C))	

Choosing ¬ (B ∧ C) as the resource to exploit by CR at stage 3 would lead to a somewhat shorter and simpler derivation.

Glen Helman 03 Aug 2010