3.39. Indirect Deduction Strategy

We promised that Indirect Deduction (ID) could establish the validity of arguments that the deductive rules alone cannot – offering the following as a simple example.

1. $\sim P$

$\therefore \sim(P \wedge Q)$

With ID in hand it's easy to establish the validity of this argument.

1.	$\sim P$	
		Get $\sim(P \wedge Q)$ (ID)
2.	$\sim\sim(P \wedge Q)$	AID
3.	$(P \wedge Q)$	2, $\sim$–
4.	P	3, $\wedge$–
5.	$\sim P$	1, R
6.	$\sim(P \wedge Q)$	2, 4, 5, ID

And it's simple enough to make good on an earlier claim: that any argument deducible *without* using ID can also be deduced via indirect deduction. For we can always wrap an ID box around a completed deduction.

The following boring little deduction, for instance, reaches the conclusion, "Q," without appeal to indirect deduction.

1. $(P \vee Q)$	Premise
2. $\sim P$	Premise
	Get: Q
3. Q	1, 2, $\vee$–

But all these steps can be preserved in an indirect deduction – simply adding an ID box, and its accompany AID.[1]

1.	$(P \vee Q)$	
2.	$\sim P$	
		Get Q
3.	$\sim Q$	AID
4.	Q	1, 2, $\vee$–
5.	Q	3, 4, ID

So we lose no deductive power by pursuing every deduction through ID. And since (as in our earlier example) some arguments can be deduced *only* through ID, our strategy will be to use ID for each deduction – unless there is some obvious, simple way of reaching the conclusion through the rules alone.

Indirect Deduction Strategy: automatically use ID (unless there is an obvious way of reaching the conclusion without ID).

Now it is already a feature of our deductive system that if an argument in the formal language is valid, ID and the seven rules can demonstrate this – deducing, within the ID box, some sentence and its negation. But from the two sentences we can deduce any further sentences we please. For as noted elsewhere, an inconsistent set of sentences entails any (and every) sentence.[2]

That means that whenever one sentence and its negation can be deduced, **every** sentence and its negation can be. (Indeed, this explosive entailment is part of what makes inconsistency so unacceptable.)

[1] Note that our justification of Line 5 cites only *two* line numbers (3 and 4), rather than the *three* lines we usually cite, because in this deduction “~Q” does double duty: as the AID, and also as half of the opposite pair.

[2] In Section 3.16, “*Features of Validity*”

In that last deduction, for instance, we needn't have rested content with "Q" and "~Q".

1.	$(P \vee Q)$	
2.	$\sim P$	
		Get Q
3.	$\sim Q$	AID
4.	Q	1, 2, $\vee-$
5.	$(Q \vee X)$	4, $\vee+$
6.	X	3, 5, $\vee-$
7.	$(Q \vee \sim X)$	4, $\vee+$
8.	$\sim X$	3, 7, $\vee-$
9.	Q	3, 6, 8, ID

So in an ID we always have a choice of which pair of sentences to use. But of course it's just unnecessary work to choose a pair, such as {X, ~X}, built from a sentence letter appearing *nowhere* in the premises. In general, it saves time and labor to choose sentences which use a sentence letter appearing in the premises.

> ***Indirect Deduction Strategy***: try to choose a pair of sentences that feature a sentence letter already appearing in previous lines.

We can extend that point further: not only should our pair of sentences feature a sentence letter appearing earlier in the deduction, but when possible we should **use a sentence we already have** one member of this pair. For then we only need to come up with the other half of that pair – effectively cutting the ID work in half.

> ***Indirect Deduction Strategy***: try to use a sentence you already have as half of the pair of opposed sentences, and then get the other half.

The following argument is a simple example (though one deducible *only* through ID).

1. We’re having ice cream	1. P
2. We’re not having *both* ice cream *and* cake.	2. $\sim(P \wedge Q)$
∴ We’re not having cake.	∴ $\sim Q$

Once we’ve assumed the negation of the conclusion, we have one application of ~– before running dry on Elim rules. In this situation our strategy is to use the Intro rules set up a sentence – either the missing ingredient for an Elim rule, or the very sentence on the “Get” line.

1.	P	
2.	$\sim(P \wedge Q)$	
		Get: $\sim Q$ (ID)
3.	$\sim\sim Q$	AID
4.	Q	3, ~–

Setting up an Elim rule is no good: since we have here no conjunction, disjunction, or double-negation to break down, there is no occasion to use an Elim rule.

And before attempting to build the sentence on the “Get” line, we must keep in mind that in an ID we reach that sentence indirectly – by *first* assembling an pair of opposed sentences within the ID box. Indeed, we can’t close that box, and thereby reach the sentence on the “Get” line, until we have some such pair of sentences.

Setting out in search of a sentence and its negation, our latest bit of strategy comes to bear: instead of building up both sentences, we try to use a sentence we already have as half of the desired pair. We then only need to build its opposing sentence.

1.	P	
2.	~(P ∧ Q)	
		Get: ~Q (ID)
3.	~~Q	AID
4.	Q	3, ~–

Both “Q” and “~~Q” call for “~Q” as the other half of the pair; but no Intro rule builds a single negation. That likewise rules out “P,” which takes “~P” as the other half of the pair.

“~(P ∧ Q)” is more promising, however, as the other half of the pair is then the conjunction “(P ∧ Q)” – and there *is* an Intro rule for building a conjunction.

Applying ∧+ to Lines 1 and 4 (then bringing down Line 2, through Repetition) completes the ID.

1.	P	
2.	~(P ∧ Q)	
		~~Get~~: ~Q (ID)
3.	~~Q	AID
4.	Q	3, ~–
5.	(P ∧ Q)	1, 4, ∧+
6.	~(P ∧ Q)	2, R
7.	~Q	3, 5, 6, ID

Note that in constructing the missing half of the opposed pair of sentences, we've just found a new application for a familiar bit of strategy: using the Intro rules to break a deductive logjam. For within an ID, finding an opposed pair of sentences clearly advances the deduction.

And ID itself provides a surprising further way of reaching a sentence which breaks such a logjam. The following little (valid) argument illustrates how.

1. Either we're having both ice cream and cake, or we're having ice cream without having cake.	1. $((P \wedge Q) \vee (P \wedge \sim Q))$
∴ We're having ice cream.	∴ P

Once past the AID, the deduction hits a wall, as there are no opportunities to apply any Elim rule.

1.	$((P \wedge Q) \vee (P \wedge \sim Q))$	
		Get: P (ID)
2.	$\sim P$	AID

Our default strategy in such a case is again to 'set up' a sentence which gets the deduction moving again. But here using a sentence we already have, as half of an opposed pair, doesn't look like an option. For both Lines 1 and 2, we lack the ingredients to construct the opposing sentence – "$\sim((P \wedge Q) \vee (P \wedge \sim Q))$" and "$\sim\sim P$," respectively.

Setting up an Elim rule looks more promising, however, since Line 1 is a disjunction. To apply ∨– we then need either "$\sim(P \wedge Q)$" or "$\sim(P \wedge \sim Q)$". But since neither is the sort of sentence an Intro rule could build, the chances of advancing the deduction again seem dim.

All hope is not lost, however. Consider: if we had Lines 1 and 2 as premises, and wanted to deduce "$\sim(P \wedge Q)$" or "$\sim(P \wedge \sim Q)$" from them, we'd know *exactly* what to do: we'd write a "Get" line for the desired sentence, and start an ID.

Why not do the same here? True, we have so far only written a “Get” line immediately after the premises of a deduction. But there is no official restriction on where one can be introduced. So we declare this further un-jamming strategy: **if no other moves are open in a deduction, write a “Get” line for the desired sentence**.

> ***Indirect Deduction Strategy***: if the deduction needs a particular sentence to advance, and no application of the rules can obtain that sentence, write a “Get” line for the sentence.

Of course we use ID on that “Get” line. For otherwise we would have only the rules and the existing lines available to us; and that much, as we’ve already seen, leaves us at a deductive standstill.

We also keep in mind a point stressed in the last example: that while our ultimate goal in an ID is of course the sentence on the “Get” line, we reach it only by first deducing, within the ID box, an opposed pair of sentences. So though in our current ID we are indeed trying to get the conclusion “P,” we will do so only by *first* getting this pair of opposites.

For that reason, it would be a waste of time to write a new “Get” line for “P”.

☠ A Fruitless Maneuver ☠

1. $((P \land Q) \lor (P \land {\sim}Q))$

		Get: P (ID)
2.	~P	(AID)
		Get: P (ID)
3.	~P	(AID)

The second ID leaves us searching for an opposite pair all over again, armed with no new sentences (since the new AID on Line 3 is just what we already had on Line 2).

To avoid such an infinite loop, the sentence on the second “Get” line will instead be something not already had – in this case, the missing ingredient to trigger ∨– on Line 1.[3] Once again, we won’t get out of this second ID until we secure both halves of an opposed pair of sentences.

1. ((P ∧ Q) ∨ (P ∧ ~Q))

Get: P (ID)

2. ~P (AID)

Get: ~(P ∧ Q) (ID)

3. ~~(P ∧ Q) (AID)

But automatically applying two Elim rules, ~– and ∧–, yields all the ingredients for an opposite pair: “~P” on Line 2, and “P” on Line 5. Repeating Line 2 leaves two opposed sentences in the ID box.

1. ((P ∧ Q) ∨ (P ∧ ~Q))

~~Get~~: P (ID)

2. ~P (AID)

~~Get~~: ~(P ∧ Q) (ID)

3. ~~(P ∧ Q) (AID)

4. (P ∧ Q) 3, ~–

5. P 4, ∧–

6. ~P 2, R

That allows us to close the second box, and finish that second ID.

[3] Here we choose the negation of the left side of Line 1; but the negation of the right half would work just as well, and be as easy to deduce.

At last our deductive logjam is freed. "~(P ∧ Q)" on Line 7 is the missing ingredient for inflicting ∨– on Line 1 – yielding "(P ∧ ~Q)" on 8.

Automatic application of ∧– to Line 8 yields "P". And with both "~P" (Line 2) and "P" (Line 9) in this box, our ID is complete.

1.	((P ∧ Q) ∨ (P ∧ ~Q))	
		~~Get~~: P (ID)
2.	~P	(AID)
		~~Get~~: ~(P ∧ Q) (ID)
3.	~~(P ∧ Q)	(AID)
4.	(P ∧ Q)	3, ~–
5.	P	4, ∧–
6.	~P	2, R
7.	~(P ∧ Q)	3, 5, 6, ID
8.	(P ∧ ~Q)	1, 7, ∨–
9.	P	8, ∧–
10.	P	2, 9, ID

Such "iterated ID" – repeating the ID-building procedure, on an ID within a larger ID – thus ranks alongside the Intro rules as a tool for 'setting up' the sentence needed to complete a deduction or apply an Elim rule.

What we're doing here, of course, is using Indirect Deduction **recursively**. Just as recursive construction rules embedded, e.g., a smaller negation within a larger one, so here we build a smaller ID as part of a larger ID. And just as we could cycle on the construction rules any finite number of times – yielding larger and larger formal sentences – so in what follows we will allow embedding of any (finite) numbers of IDs within IDs, yielding IDs of arbitrarily larger proportions.

With iterated IDs added to our bag of tricks, we have in hand all the deductive tools needed to demonstrate the validity of any valid argument in our formal language. All that we add, in what follows, are new applications of these deductive tools, and strategies for streamlining their use.

Summary: Indirect Deduction Strategy

- Automatically use ID for each deduction (unless there is an obvious way of reaching the conclusion without ID).
- Try to choose an opposed pair of sentences featuring a sentence letter already appearing on some previous line.
- Try to use a sentence you already have as one of the opposed pair of sentences, and then get the other half.
- If a sentence is needed to advance the deduction and cannot be reached through the deductive rules applied to existing lines, write a “Get” line for that sentence and use ID for the new “Get” line. (ID acts on a par with the Intro rules to ‘set up’ a needed sentence.)