2.2. Proofs: analyzing entailment

2.2.0. Overview

We can get some insight into deductive logic by looking at basic principles of entailment, but more will come by looking at how these principles may be combined in proofs.

2.2.1. Proofs as trees
The simplest way of combining deductive principles takes the shape of a tree in which premises, premises from which these premises are concluded, and so on, grow and branch from the final conclusion.

2.2.2. Argument trees
The search for a proof can take the form of a tree connecting arguments whose validity is tied.

2.2.3. Derivations
Although writing a proof as a tree can make its structure very explicit, we will mainly use a compact notation that more closely matches the patterns that are used when deductive reasoning is put into words.

2.2.4. Rules for derivations
In the context of derivations, principles of entailment take the form of rules that direct the search for a proof.

2.2.5. An example
All derivations that involve conjunction alone share many features; we will look closely at one typical example.

2.2.6. Two perspectives on derivations
Derivations have aspects that reflect both conclusion trees and argument trees; the latter aspect will prove especially important.

2.2.7. More rules
Tautology and absurdity provide a first example of derivation rules for logical forms other than conjunction.

2.2.8. Resources
In order to plot a course in constructing a proof for a given conclusion, we need to keep track of not only the premises but also the conclusions that have already been reached.

Glen Helman 01 Aug 2013

2.2.1. Proofs as trees

Our study of entailments involving conjunction will rest on the principles discussed in 2.1.1. These are shown below, in symbolic form on the left and in English on the right:

$\varphi \wedge \psi \vDash \varphi$	both φ and $\psi \vDash \varphi$
$\varphi \wedge \psi \vDash \psi$	both φ and $\psi \vDash \psi$
$\varphi, \psi \vDash \varphi \wedge \psi$	$\varphi, \psi \vDash$ both φ and ψ.

We will refer to the first two of these patterns as *extraction* (*left* and *right* extraction to distinguish them) and to the third simply as *conjunction*. To establish particular cases of entailment, we will want to put together instances of these general patterns and, eventually, instances of other patterns, too. We will look at three different sorts of notation for doing this. It is the third, to be introduced in the next subsection, that we will employ in the end; but two others, which we will consider in the this subsection are useful for understanding the one we will actually use.

What may be the most direct notation for combining principles of entailment employs something like the two-dimensional form we have used for arguments, with the conclusion below the premises and marked off from them by a horizontal line. In order to make the premises of a multi-premised argument available to serve as conclusions of further argument, we will spread them out horizontally. In this style of notation, the basic patterns for conjunction take the following forms (where abbreviations of their names are used as labels):

$$\text{Ext}\ \frac{\varphi \wedge \psi}{\varphi} \qquad \text{Ext}\ \frac{\varphi \wedge \psi}{\psi} \qquad \text{Cnj}\ \frac{\varphi \quad \psi}{\varphi \wedge \psi}$$

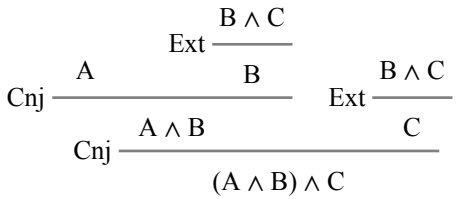
To these we will add the one way of drawing conclusions available for all sentences:

$$\text{QED}\ \frac{\varphi}{\varphi}$$

While this adds no new conclusion, it will play a role in joining compound arguments that were constructed separately. The label for this rule reflects that role. It abbreviates the Latin quod erat demonstrandum (which might be translated as what was to be proven), a phrase that is traditionally used when a planned conclusion is reached.

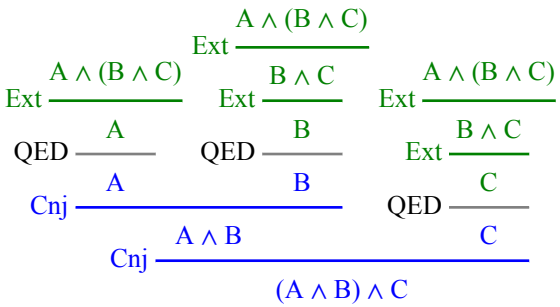
Arguments exhibiting these patterns can be linked by treating the premises of one argument as conclusions of other arguments. For example, the follow-

ing shows that (A ∧ B) ∧ C is a valid conclusion from the two premises A and B ∧ C:

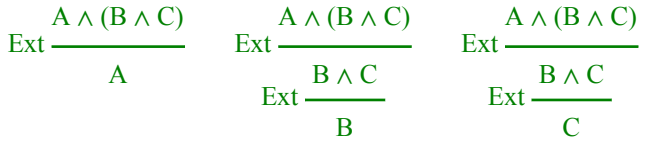


The ability to put entailments together in this way rests on the general laws of entailment discussed in 1.4.7. The law for premises enables us to begin; it shows that the premises A and B ∧ C entail the tips of the branches of this *conclusion tree*. Repeated uses of the chain law then enable us to add conclusions drawn using the principles for conjunction, and we work our way down the tree showing that the original set of premises entails each intermediate conclusion and, eventually, (A ∧ B) ∧ C. For example, just before the end, we know that our original premises entail each of the premises of the final conclusion —i.e., that A, B ∧ C ⊨ A ∧ B and A, B ∧ C ⊨ C. The chain law then enables us to combine these entailments with the fact that A ∧ B, C ⊨ (A ∧ B) ∧ C (a case of Conjunction) to show that A, B ∧ C ⊨ (A ∧ B) ∧ C.

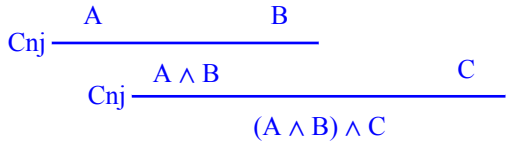
The simplicity of these conclusion trees makes them useful for studying the general properties of proofs; however, we will employ a somewhat different way of representing proofs. One reason is that conclusion tree do not provide much support for discovering proofs. There is a small modification can help in that respect and that will lead us in the direction of the representation we will use. If we employ QED in the midst of a tree, we can distinguish separate components. That's done here in the following proof showing that A ∧ (B ∧ C) ⊨ (A ∧ B) ∧ C:



Here we have three conclusion trees (mere unbranched saplings) drawing conclusions from the initial premise by Ext



glued to the tips of a tree deriving the desired conclusion by use of Cnj



Notice that the conclusions of the trees using Ext are all unanalyzed sentences, as are the premises of the tree using Cnj.

The net effect is that the premises simplify as you move down one of trees of using Ext from tip to root while there is simplification in the tree using Cnj when you move up from root to tip. Assuming that this can always be managed, it suggests a systematic way of finding a conclusion tree for given premises and conclusion: derive simpler premises from the initial ones using Ext and plan to get the final conclusion from simpler ones using Cnj.

That we can always construct a conclusion tree in this way follows from the two principles below. Although they are related to the validity of Ext and Cnj, they take a different form from principles stating the validity of particular patterns of argument. Instead they describe general conditions under which any arguments involving conjunction are valid.

LAW FOR CONJUNCTION AS A PREMISE. Γ, φ ∧ ψ ⊨ χ if and only if Γ, φ, ψ ⊨ χ

LAW FOR CONJUNCTION AS A CONCLUSION. Γ ⊨ φ ∧ ψ if and only if both Γ ⊨ φ and Γ ⊨ ψ

These principles can be seen to hold by the comparing the sort of possible worlds each side of the if and only if rules out. To separate the premises of Γ, φ ∧ ψ / χ from its conclusion, we would need to do exactly what we'd need to do to separate the premises of Γ, φ, ψ / χ from its conclusion. And to separate the premises of Γ / φ ∧ ψ from its conclusion, we'd need to do exactly what we'd need to do the show the right side of the second law false—that is, separate premises from conclusion for at least one of Γ / φ and Γ / ψ.

The if parts of these principles reflect the validity of arguments of the forms Ext and Cnj, respectively, together with the chain law. For example, the if part of the first says that any valid conclusion from φ and ψ, perhaps along with other premises Γ, will follow from φ ∧ ψ along with those other premises, so we are not going astray as we move down a tree of premises. The only if parts

tell us that the validity of the arguments on their left sides can always be established in the way licensed by the if part. For example, the only if part of the second tells us that, if a conjunction is a valid conclusion, then the premises needed to reach it by Cnj are bound to be valid conclusions also; so it should be possible to establish what we need to in order to apply Cnj. In sum, these two principles together show that the sort of mixed tree of premises and conclusions shown above will always suffice in constructing a proof.

Glen Helman 01 Aug 2013

2.2.2. Argument trees

In the simple form in which we have considered it, the approach to finding proofs described at the end of the last subsection will really work only with conjunction. Among the logical forms we will consider, it is only conjunction whose content is the cumulative content of its components, so in particular, it is the only one whose role as a premise can be simply taken over by its components. However, in the case of other logical forms, we will still have principles like the laws for conjunction as a premise and as a conclusion. The difference will be that the laws will not concern solely premises or conclusions. A law describing the role of a logical form as a premise, for instance, may refer to the role of a component as a conclusion. Proofs involving these other logical forms also force significant modifications to conclusion trees. Such modifications are possible, and we will consider them briefly in later chapters, but the most natural way of implementing the laws for these other logical forms is a different sort of tree which permits consideration of both premises and conclusion at the same time—in short, a tree composed of arguments rather than single sentences.

We will generally find such a tree—an *argument tree*—growing on its side, with the arguments written vertically.

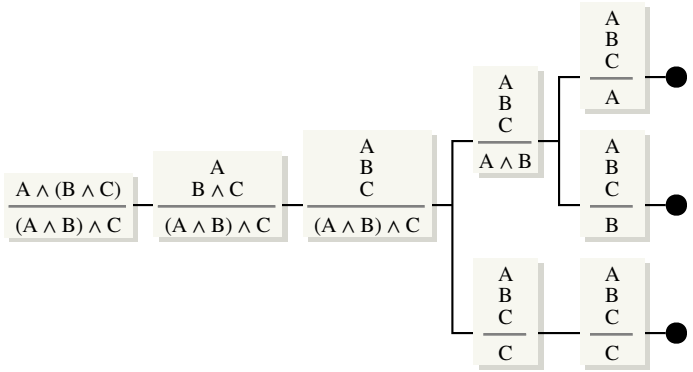
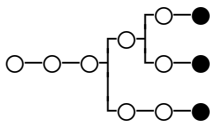


Fig. 2.2.2-1. An argument tree growing from left to right, using the law for conjunction as premise twice, followed by two uses of the law for conjunction as conclusion, and ending with three arguments whose validity follows from the law for premises.

Moving left to right, we first use the law for conjunction as a premise to analyze premises, replacing them by simpler ones that whose cumulative content is the same. We then analyze the conclusion using the other law. This leads us to replace an argument by two whose conclusions are the conjuncts of the conclusion we analyzed. Finally, at the tips of the branches, we have reached argu-

ments that are valid by the law of premises, and that is indicated by a bullet marking the end of this part of the search for a proof. If every path leading from an argument ends in a bullet, we know that the argument is valid because the paths have lead us to arguments whose validity is tied to its validity by the laws for conjunction as a premise and as a conclusion.

Since an argument tree leads us to see that an argument is valid, it counts as a proof in its own right. But the arguments whose validity is tied by the two laws for conjunction are connected by the use of the patterns of argument Ext and Cnj, so an argument tree points us to a way of using those patterns to construct a conclusion tree. The figure below illustrates this for the argument tree above. The buttons controlling it take the form of a schematic version of that tree with open circles representing arguments for which a proof is not yet provided.



$A \wedge (B \wedge C)$

?

$(A \wedge B) \wedge C$

Fig. 2.2.2-2. A conclusion tree discovered using an argument tree. Run a pointer over (or click on) parts of the schematic argument tree at the top to show stages in the discovery of a conclusion tree. (The full argument tree is the one illustrated earlier.)

The question marks that appear before the last stage each indicate a conclusion that has not yet been connected with the premises or the conclusions derived from them.

Like conclusion trees, argument trees illustrate important features of deductive reasoning. But, also like conclusion trees, they are cumbersome to write since they take up space and involve rewriting sentences several times. In the next subsection, we will look at a more compact notation for proofs. Of course, any notation for writing out proofs is more than we need to settle questions of entailment involving only conjunction. But the complications introduced by the logical forms we will consider in later chapters make it useful to have some system of notation; and it will help in understanding this notation to introduce it now since the simplicity of conjunction makes it easier to see what is going on. We will write out conclusion trees and argument trees only rarely from this point on, but our more compact notation will have ties to both of them, and it will useful to look at them from time to time since they exhibit quite clearly some features of the compact notation that are disguised by its compactness.

Glen Helman 01 Aug 2013

2.2.3. Derivations

Both conclusion trees and argument trees spread out in two dimensions. The more compact notation that we will actually use will be more linear, though still somewhat two-dimensional. We will gain compactness by listing premises and conclusions in a more-or-less vertical way and by minimizing the repetition of premises that are used draw a number of conclusions. We will still need a tree structure to keep track of the premises relevant any given point, but this will involve rather stunted trees. These will grow left to right like argument trees; and, indeed, this notation consists most essentially of argument trees that have been squashed from branch tips to root.

Compactness is not all we will gain with this notation. It it will approximate the ways proofs are normally stated in language. Indeed, although we will not emphasize this aspect of it, the notation for proofs could be thought of as a notation for analyzing the form of proofs presented in English that is in some respects analogous to our symbolic notation for analyzing the logical forms of sentences.

The system to be developed here falls into a broad class often referred to as *natural deduction systems* because they replicate, to some extent, natural patterns of reasoning. Such systems were first set out in full in the 1930s by G. Gentzen and also by S. Jaskowski, but some of the key ideas can be found already in the Stoic philosopher Chrysippus (who lived in the 3rd century BCE). The notation we will be using is an adaptation of notation introduced by F. B. Fitch but our approach to these systems will be influenced heavily by the "semantic tableaux" of E. Beth. (Their ideas now go back about 60 years.)

This system, which we will call a *system of derivations*, will employ a perspective on proofs that we adopted in the last section whenever we considered ways of restating claims of entailment. If we ask whether an entailment holds, we find ourselves faced with the task of reaching the conclusion from the premises (or showing that it cannot be reached). Let us think of the conclusion as our *goal* and of the premises as the *resources* we have available in trying to reach that goal. Until we reach the goal, it lies on the other side of a *gap* that it is our aim to close.

We begin in the state shown in Figure 2.2.3-1, with a single gap between the premises and conclusion the argument whose validity we are trying to establish.

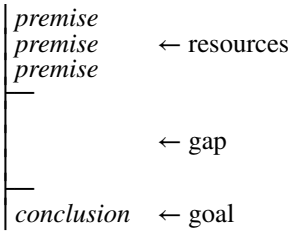


Fig. 2.2.3-1. The initial state of a derivation.

The premises of the argument (if it has any) are written above a horizontal line, and the conclusion is written below a second line. The space in between the horizontal lines marks the gap and will be filled in with additional resources and new goals as the derivation develops. (The vertical line on the left will be discussed later.)

We will approach the problem of closing the initial gap (or showing that it cannot be closed) step by step. At each step, either we will plan the way a goal may be reached or we will exploit resources, usually by drawing one or more conclusions from them. In making a step of either sort, we will restate our problem with different goals or resources, and we will say that, by this restatement, we are *developing* the derivation. When it is seen from this perspective, the problem of closing a gap is a problem of connecting available premises with desired conclusions. In developing a derivation, we work forward from premises and backward from conclusions in hopes of making this connection. Although conclusion trees will be only in the background of these derivations, the process of filling a gap from both ends to the middle that was displayed in Figure 2.2.2-2 is closely related to this sort of development.

This process may lead us to divide a gap in two. In the case of conjunction, this will happen when we plan to reach a goal $\varphi \wedge \psi$ by first concluding φ and ψ individually, for we will then set each of φ and ψ as a preliminary goal and each will have a gap before it. This development of our initial problem by restating it and perhaps dividing it into subproblems will be expressed in a sort of tree structure just as it is in argument trees. However, a derivation will be written as a more or less vertical list of sentences. The subgoals that we plan to reach in order to go on to a further goal will be written one above the other, each preceded by space for further growth, and conclusions we reach by exploiting resources will be written in at the top of a gap. In order to indicate the tree structure of problems and subproblems within this vertical list of sentences, we will need to mark up the derivation in various ways.

We will employ two main devices for doing this. One is the numbering of stages and sentences added at those stages. The other device is a system of vertical lines like the line at the left in Figure 2.2.3-1. These lines will be called

scope lines, and they will serve us in a number of ways. First of all, new scope lines will be introduced as we analyze goals, with a separate scope line serving to mark the portion of the derivation devoted to each subgoal. The scope line will indicate the portion of derivation where a given subgoal is the goal we are aiming at, and it is in this sense that the scope line marks scope of the subgoal. As scope lines accumulate, they will be nested, some to the right of others, in a way that indicates the tree structure of the proofs. In later chapters, proofs will sometimes involve assumptions beyond the initial premises, and scope lines will then also serve to mark the portions of a proof in which these assumptions are operative—that is, they will serve to mark the scope of assumptions as well as goals. Later still, the scope lines will be labeled to indicate vocabulary that has a special role in the portion of a derivation marked by the scope line.

At any stage in the development of a derivation, each gap will have certain *active resources*. These are resources available for use in the gap that have not already been exploited in developing it. These resources can be used to form the premises of an argument, using the gap's goal as its conclusion. We will call this argument the *proximate argument* to distinguish it from the *ultimate argument* for which the derivation is being constructed. Our aim in a developing a gap will always be to see whether the goal of the gap is entailed by its active resources, whether its proximate argument is valid. And this means that the situation depicted in Figure 2.2.3-1, which is explicit at the beginning of the derivation for the case of the ultimate argument, will be replicated, although less explicitly, for proximate arguments throughout the development of a derivation. This is the way derivations are associated with argument trees: the proximate arguments of the gaps of a derivation form an argument tree, and the development of the derivation is really the development of this tree. The argument tree and conclusion tree associated with a derivation may not be immediately apparent, but their presence will be felt in the bookkeeping required by the system of scope lines and stage numbers. This bookkeeping is the price paid for using less space and less repetition in writing.

2.2.4. Rules for derivations

One way of developing a gap is to restate our problem so that one of its resources can be dropped from consideration, perhaps adding others of equivalent power but simpler form. We will call this process *exploitation*, and one example is provided by the way we implement the law for conjunction as a premise. That principle tells us that anything we can conclude from premises that include a conjunction can still be concluded if we replace the conjunction by its two components. In derivations, we will apply this idea by adding, as further resources, both of the conclusions that can reached from a conjunction by Ext. By adding both conclusions, we eliminate any further need to consider the conjunction we are exploiting; but, since both conclusions may not be needed to reach our ultimate goal, a derivation may contain conclusions that are never used later.

The derivation rule Extraction thus takes the form shown in Figure 2.2.4-1.

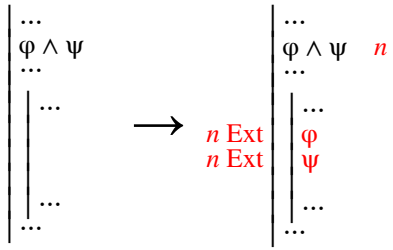


Fig. 2.2.4-1. Developing a derivation by exploiting a conjunction at stage *n*.

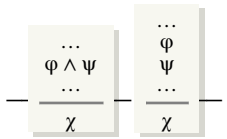
On the left, the gap is shown nested inside scope lines (two are shown but there may be just one or more than two). A conjunction is displayed at the top to show that it is among resources available for use in this gap. It is shown to the right of one of the scope lines running to the left of the gap but not to the right of the other. The requirement this illustrates is that a resource being exploited need not be inside all the scope lines to the left of the gap but cannot be inside any extra ones; that is, *a resource cannot be exploited after the end of any line to its left*.

The right side of the figure illustrates the results of exploiting the conjunction. When we exploit it, we add its components as new resources at the top of the gap. If either component of the conjunction should happen to be already among the active resources of the gap, it would not be necessary to add this component again; but there is nothing wrong with doing so, and examples in the text will generally add it. (Although this practice may make the derivation slightly less compact, it makes it possible to focus solely on the parts of the derivation that are immediately relevant to the rule—i.e., the ones displayed in

the diagram above.)

The number *n* of the new stage in the development of the derivation is written to the right of the conjunction to show that it has been exploited at this stage, and the stage number is also shown, along with the label Ext, to the left of each of the two lines that are added.

Once the conjunction has been exploited, it is no longer an active resource for this gap though it could be active in other gaps (we will see later how to tell). So the effect of this rule on proximate arguments is the sort of segment of an argument tree shown at the right.



The numbers in a derivation are also one of the devices derivations use to encode the structure of conclusion trees: they mark the relation between premises and conclusion that conclusion trees marked the horizontal lines between premises and conclusions. In English argumentation, words and phrases like therefore, hence, and it follows that indicate the same sorts of connections in a less explicit way.

Exploiting resources like this is one way to narrow a gap. Another way to narrow a gap is to restate the problem it represents so that the goal we seek to reach is replaced by one or more simpler goals. We will call this process *goal planning*. The law for conjunction as a conclusion tells us how we may plan for a goal that is a conjunction. Such a goal is entailed by our active resources if and only if each of its components is entailed. So the project of reaching a conjunction φ ∧ ψ from given resources comes to the same thing as completing two projects—namely, reaching each of the components φ and ψ from those same resources. This sort of goal planning thus uses Cnj and takes the form shown in Figure 2.2.4-2.

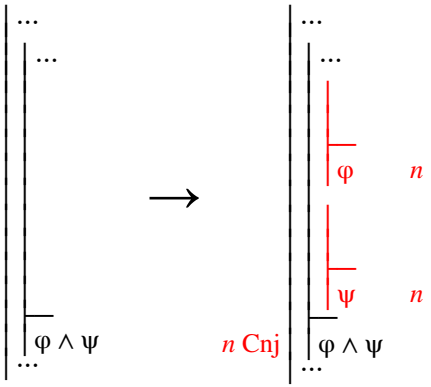
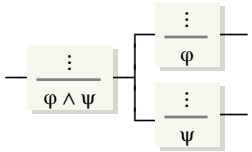


Fig. 2.2.4-2. Developing a derivation by planning for a conjunction at stage *n*.

On the left, no assumptions are made about the resources, but the goal is shown as a conjunction. On the right, we have introduced two new gaps, each with one of the conjunction's components as its goal. The two new goals bring with them two scope lines and are marked off by horizontal lines (as was the initial conclusion) to show that they represent the new material that led to the use of new scope lines. At the right of each of the new goals is a number showing the stage at which it was added. The same number appears to the left of the goal along with the label Cnj. This rule is applied in the sort of case where argument trees branch,



and this branching appears in derivations in the division of one gap into two, each with its own goal.

While in the case of Ext, numbers appeared at the left of the resources that were added and at the right of the resource being exploited, numbers here appear on the right of the new goals and at the left of the old one. This is because the new goals added by Cnj are introduced as premises from which the old goal may be concluded while the resources added by Ext are added as conclusions drawn from the resource that is exploited. Still, in both cases the numbers mark a connection between premises and conclusions. The numbers also show for both rules how an element of the derivation has been superceded by new additions. But, in the case of Cnj, this information is also provided by the added gaps: a gap will always have exactly one goal, and that goal will appear immediately below it.

The new gaps introduced in planning for a conjunction initially have the same active resources as the original gap. As resources are exploited in narrowing one of the gaps, these resources will become inactive for that gap; but they will remain active for the other gap until they are exploited there. When a derivation contains more than one gap, the question of where resources are active becomes important, and something will be said about it before too long. But, when we are dealing with conjunction alone, it is possible to exploit the initial resources completely before we plan for goals. As a result, a general discussion of active and inactive resources can be postponed until we have considered an actual example of a derivation.

What we cannot postpone is an account of how a gap may be closed. If the goal of a gap appears also among its resources, the law for premises tells us that the goal is entailed by these resources. That means we have succeeded in making a connection between our resources and that goal, and the gap may be

closed. The rule we use to do this is shown in Figure 2.2.4-3 below.

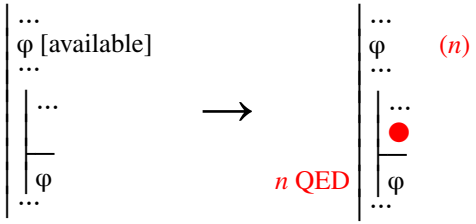
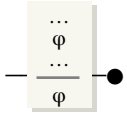


Fig. 2.2.4-3. Closing a gap by locating its goal among its resources.

The phrase abbreviated by the label for this rule (see 2.2.1) reflects its function: we can close a gap when the goal we have sought can be found among our resources. This corresponds to the end of a branch in argument trees,



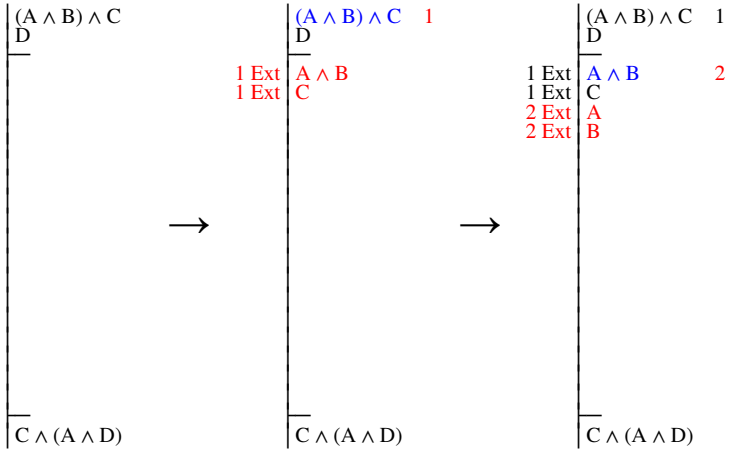
and we use the same symbol ● (a *filled circle*).

The stage number appears to the left of the goal (along with the label) since the goal is the conclusion, and it appears to the right of the resource since the resource is the premise. The latter number is enclosed in parentheses to indicate that the premise is not here being exploited. Since the gap is closed, the question whether a resource is active or not becomes moot; but this sort of notation will be used later in other cases where resources are used without being replaced by simpler resources of equivalent content, and QED shares with these rules the feature that the resources to which it is applied do not need to be active. Although the symbol ● occupies a line, it is not the conclusion or premise of the argument and is not given a stage number. Think of an analogy with written language: the symbol marks the end of a series of stages in the way a period marks the end of a series of words without being a word itself.

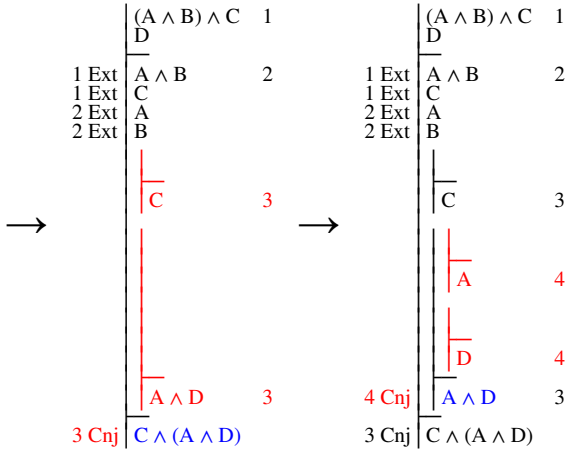
Glen Helman 01 Aug 2013

2.2.5. An example

Now, let us look at an example using these rules. The development is shown stage by stage below. At each stage, new material is shown in red. Resources that are exploited or goals that are planned for are shown in blue. At each of the stages 1 and 2, a resource is exploited. The added resources are conclusions drawn from the exploited resource, so the number of the stage is written at the left of the resources that are added and at the right of the one that is exploited.

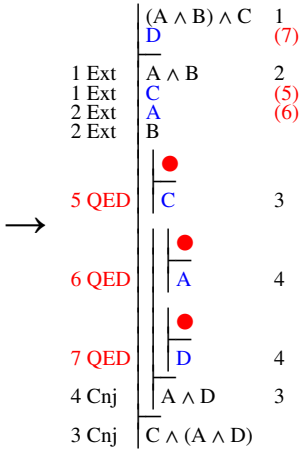


In stages 3 and 4, we plan for goals. The goals we add in each case are premises from which we plan to conclude the goal we are planning for. The stage number therefore appears at the right of the new goals and to the left of the old one.

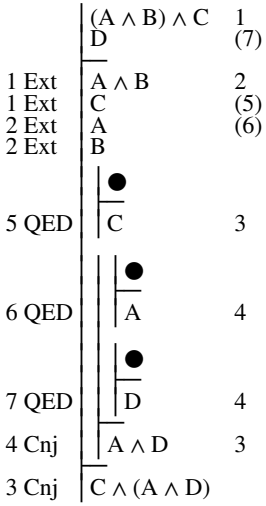


In the last three stages we close gaps. Although these are different stages,

they are independent of one another and could have been done in any order, so all three are shown together. No sentences are added and the stage numbers merely mark the connection between resources that serve as premises and the goals that are concluded from them (both shown in blue).



If your browser has JavaScript enabled, the diagram below can be used to display each stage in the development of the derivation we have been considering.



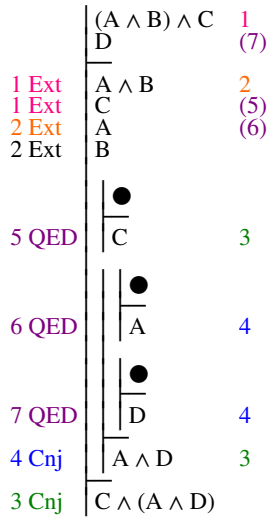
When this sort of animation is not available, the stage numbers in a completed derivation can be used to reconstruct its history.

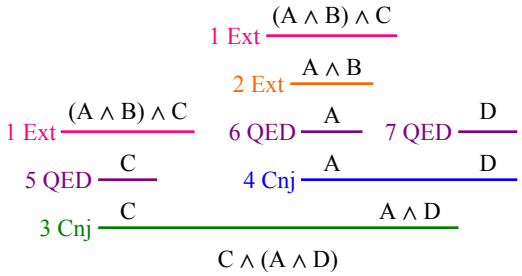
Glen Helman 01 Aug 2013

2.2.6. Two perspectives on derivations

The locations of the stage numbers appearing in a derivation reflect the patterns of argument on which the derivation rules are based. The label for a rule always appears to the left of the conclusion of such an argument, and the number of the stage at which the rule was applied appears not only next to the label but also to the right of the premises of the argument. A conclusion tree can be reconstructed from the derivation by beginning with the final conclusion and working backward to the premises from which it was concluded, the premises from which those were concluded, and so on.

If we apply this idea to the example of the last section (which is reproduced below), we get the conclusion tree following it.



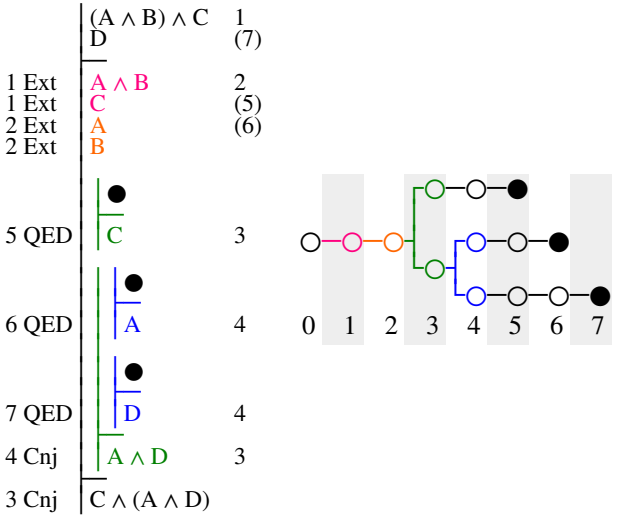


The sentence B concluded by Ext at the second stage of the derivation does not appear in the conclusion tree because it is not used as a premise for any later conclusions, something that is marked in the derivation by the fact that it has no stage number to its right.

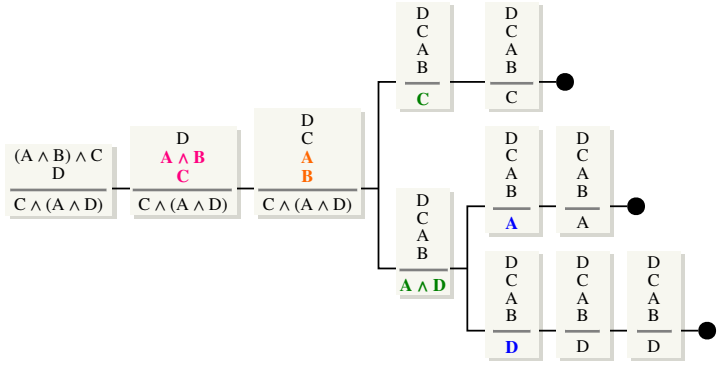
Looked at in this way, a derivation could be thought of as the result of disas-

sembling a conclusion tree and stacking the pieces up vertically. When reassembling the tree, we paid no attention to the horizontal organization provided by scope lines. The order of the stage numbers played no role either: they could just as well have been arbitrary codes used to mark corresponding parts of the tree so they could be fit together again. Indeed, even the vertical order of the lines of the derivation did not matter. Matching numbers on the left with numbers on the right is all that was necessary to reassemble the tree, and pieces could have been given to us in an unorganized heap. However, all these features of derivations, which are not needed to reconstruct a conclusion tree, do matter for another, and more important, way of looking at derivations, one in which a derivation is associated with a argument tree.

To see this association, first use the stage numbers, scope lines, and the vertical ordering of lines to determine the way the gaps of the derivation develop over time, beginning with the intial gap, eventually dividing, and finally closing. That is shown on the right in the diagram below, where the stages are arrayed left to right and gaps are indicated by circles, with a filled circle used to indicate closure and an *empty circle* used to indicate a gap that is open. Colors are used to emphasize where and when changes occur.



We used similar notation in Figure 2.2.2-2 to represent the skeleton of an argument tree, and we get an argument tree if we flesh out the skeleton above by including the proximate arguments of the gaps that the empty circles represent, something that is shown below.



Colored sentences are new additions as the tree grows, and premises are added at the end of the list to match the order of active resources in the derivation above. Comparison with it should give you some sense of the way in the which a derivation amounts to a squashed argument tree: repeated premises and conclusions coincide and new ones are folded in towards the middle.

Glen Helman 01 Aug 2013

2.2.7. More rules

A couple of the principles for ⊤ and ⊥—in particular, with the laws for ⊤ as a conclusion and ⊥ as a premise—have a role to play in derivations. Like the laws for conjunction, these laws have associated patterns of valid argument:

$$\text{ENV}\ \frac{}{\top} \qquad \text{EFQ}\ \frac{\bot}{\varphi}$$

The label for the second, EFQ, abbreviates the Latin ex falso quodlibet (which might be translated as from the false, whatever), a traditional way of stating the law for ⊥ as a premise, and the label for the first, ENV, abbreviates ex nihilo verum (from nothing, the true), which gives a corresponding statement of the law for ⊤ as a conclusion.

The two other laws for ⊤ and ⊥ do not have associated patterns of argument and will not be associated with steps in proofs. The law for ⊤ as a premise does not point to a pattern of argument whose conclusion could replace ⊤, for it tells us that ⊤ may simply be dropped from the premises. In fact, it will be just as easy to retain ⊤ as an active resource but ignore it. And that will make our handling of ⊤ more like our handling of ⊥. For we cannot apply the principle for ⊥ as an alternative unless we begin with multiple alternatives or end with none, so it is not a principle of entailment at all and provides no way of replacing ⊥ as a conclusion. This does not mean that ⊥ as a conclusion is insignificant in the way ⊤ is insignificant as a premise, but the role of ⊥ as conclusion is to mark an entailment as a claim of inconsistency, and such claims will be established by applying principles to their premises rather than to their conclusion. (However, we will eventually have some rules for exploiting resources that we will apply only when the goal is ⊥.)

The principles ENV and EFQ figure in derivations as rules for closing gaps. In the case of the first, it is enough for a gap to be closed that it have ⊤ as its goal. No resource is involved, and the stage number appears only as an annotation to the goal.

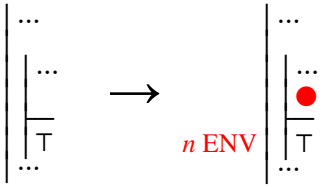


Fig. 2.2.7-1. Closing a gap that has ⊤ as its goal.

The rule EFQ takes a form much like QED.

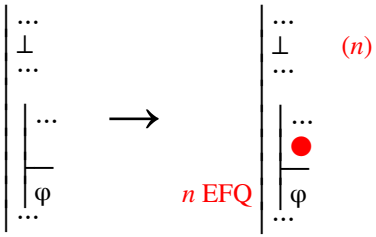


Fig. 2.2.7-2. Closing a gap that has ⊥ among its resources.

The difference is that having ⊥ as a resource enables us to close a gap no matter what its goal is. (If the goal also was ⊥, either EFQ or QED could be used.)

Here are examples of the use of these rules:

	A	(5)
	B	(3)
3 QED	B	2
4 ENV	⊤	2
2 Cnj	B ∧ ⊤	1
5 QED	A	1
1 Cnj	(B ∧ ⊤) ∧ A	

	A ∧ (⊥ ∧ B)	2
2 Ext	A	
2 Ext	⊥ ∧ B	3
3 Ext	⊥	(4),(5)
3 Ext	B	
4 EFQ	C	1
5 EFQ	D	1
1 Cnj	C ∧ D	

Notice that, while every stage number of the second derivation appears somewhere among the annotations on its right-hand side, the same is not true of the first derivation because stage 4 is missing. Of course, that's because stage 4 is when we used ENV, and ENV is a valid argument without premises. Since stage numbers appearing in annotations on the right-hand side of a derivation mark the use of a line as a premise and ENV is the only form of argument we have seen so far that has no premises, a use of the rule ENV should be the only reason for a stage number to appear on the left of a derivation but not on the right.

You can use this idea as a way of checking for errors, and there are some further generalizations like this that you can use as checks. We will have no rules without conclusions, so every stage number should appear somewhere in the left-hand annotations. And, in a completed derivation whose gaps all close, all sentences other than assumptions (which, for now, are just the initial premises) will be conclusions and thus should have annotations on their

left-hand side. Resources that are never used may appear with no annotations on their right; but, as you are constructing a derivation, it can be very useful to check for the absence of right-hand annotations because this can lead you to notice resources that you have not yet exploited. And, when we go on (in 2.3) to use derivations to show that claims of entailment fail, a check for the absence of right-hand annotations will be the key test of whether we done everything possible to complete a derivation.

Glen Helman 01 Aug 2013

2.2.8. Resources

The ideas of available and active resources have been used at several points already, but they have not yet been explained fully. A resource counts as *available* in a gap if it was entered either as one of the initial premises of the derivation or in the course of developing the gap in question. The system of scope lines can be used to tell which resources are available in a gap: a resource is available if every scope line to its left continues unbroken at the left of the gap.

One way of thinking about this is shown in Figure 2.2.8-1.

Fig. 2.2.8-1. The boxes indicated by the scope lines of a derivation. If JavaScript is enabled on the browser you are using, moving the cursor over a resource will color the gaps in which it is available green and shade areas where it is unavailable. Moving the cursor over a gap will color resources available in it green and shade areas whose resources are unavailable to it. The resource or gap that the cursor is over will be colored blue and underlined.

You may suppose that each scope line indicates the left side of a box and that a resource is available only to the gaps that are also within the smallest box containing it.

A resource is *active* in a gap if it is available in that gap and has not already been exploited in narrowing it. The easiest way to locate the active resources

of a gap is to scan the available resources and eliminate the inactive ones. To be inactive in any gap, a resource must have been exploited at some stage. If it has, there will be an unparenthesized stage number to its right. A resource may have been exploited only in some gaps and may still remain active in others. To be inactive in a given gap, the resource must have been exploited in narrowing the gap. To see whether this is so, we need to check all resources and goals that were introduced at a stage when the resource was exploited (i.e., at a stage whose number appear unparenthesized to the resource's right). (So far, we have seen goals introduced only in the course of planning for more distant goals, but in later chapters they will be introduced as part of the exploitation of certain resources.) If any such resource or goal is such that the smallest box containing it also contains the gap we are considering, it was introduced in the course of developing the gap. A resource may be exploited more than once, so there may be several stage numbers you will need to check. If any of them was a stage in which the gap you are considering was developed, the resource is no longer among the active resources of the gap. This description of the process may make it sound rather daunting, but in practice you will find that it is usually obvious which resources have been exploited in developing a given gap.

The partially developed derivation shown below has been designed to provide an example of a resource that has been exploited without being exploited in all gaps in which it is available.

	(A ∧ B) ∧ C	1
1 Ext	A ∧ B	3
1 Ext	C	
3 Ext	A	
3 Ext	B	
	A	2
	B	4
	C	4
4 Cnj	B ∧ C	2
2 Cnj	A ∧ (B ∧ C)	

The three steps at the top of the derivation are resources available for each of the derivation's three gaps. The first, (A ∧ B) ∧ C, is inactive in all three gaps. It was exploited at stage 1, and that was the initial stage of development for all the gaps of the derivation. The second resource, A ∧ B, is inactive for the first of the gaps (having been exploited at stage 3 in developing this gap), but it is active for the remaining two gaps since the resources introduced at stage 3 did nothing to narrow these gaps (as is shown by the fact that the gaps are outside the smallest box surrounding the resources with 3 at their left). The third resource C has not been exploited at all (and could not be since it is not a conjunction), so it is active for all three gaps. Since the resource exploited at stage 3 must be exploited again in order to close the second gap, it would have been a little more efficient to exploit this resource before dividing the initial gap in two; but the derivation as shown is perfectly correct (though still unfinished).

You may suppose that a given gap can see only those parts of a derivation that are not boxed off from it—i.e., only those parts all of whose scope lines continue to the left of the gap. If a stage number appears at the left only in parts of the derivation that are invisible to the gap, this stage number is also invisible—even when it appears to the right of resources that are visible.

This idea is illustrated in Figure 2.2.8-2 below where the same derivation is shown from the perspective of each of the three gaps in turn.

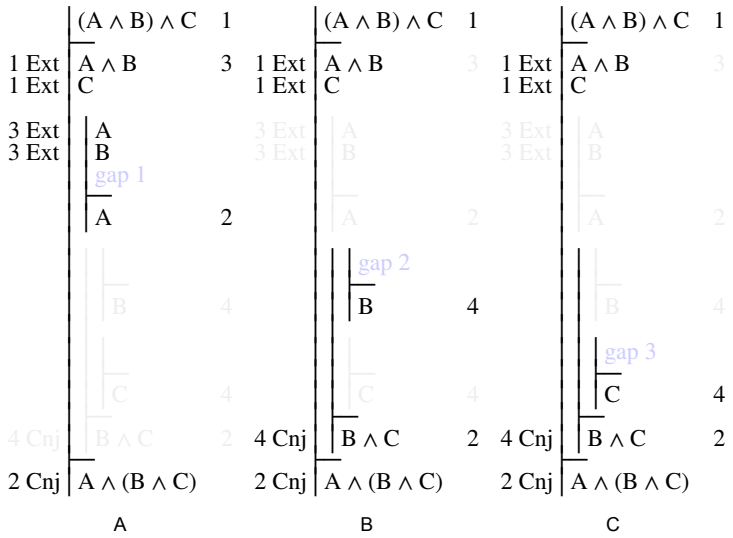


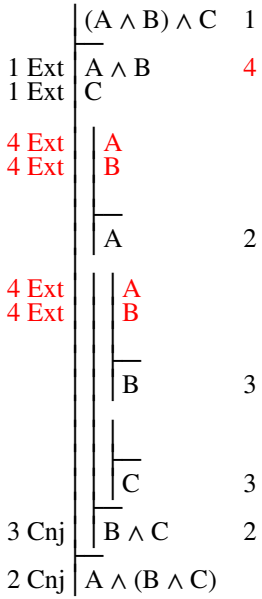
Fig. 2.2.8-2. A derivation from the perspective of each of its three gaps.

Material that is boxed off from a gap is shown in very light gray. Notice that the number 3 at the right of the second line is invisible to the second and third gaps. As we saw earlier, that is because all the development at stage 3 is boxed off from the second and third gaps.

Any derivation can be thought of as the result of superimposing layers like these. There will be one layer for each gap with a gap's layer depicting its perspective on the derivation. This corresponds directly to a feature of argument trees: a gap can see what is on the path from it back to the root of the tree, and

the superimposing layers to make up a derivation corresponds to superimposing paths to make up a tree. When we distinguish the resources available for a gap or determine whether a resource has been used to narrow a gap, we are really considering that gap's layer separately, which is to say we are considering its path to the root apart from paths that have branched off.

When a gap is divided before a resource is exploited to narrow it, it is possible to exploit the resource to narrow several gaps at once. This is shown in the partial derivation below (which has the same initial premises and conclusion as the one we have been considering).



In this derivation, one of the resources has just been exploited at stage 4 to narrow two different gaps. Thereafter, it is inactive in these gaps but still active in the third (where it happens to be unneeded). Some of the resources added at stage 4 will be invisible to each of the first two gaps; but, because other added resources are visible, the number 4 at the right is visible from both these gaps. However, none of the resources added at stage 4 is visible from the third gap, so the number 4 at the right is not visible from it.

Since we use a similar numerical notation for both resources that are exploited and goals that have been planned for, you might expect that the concepts of availability and activity can be applied to goals as well as resources; and, indeed, they can be. If we were to consider derivations for relative exhaustiveness, we would need to engage in the same sort of accounting for goals that we have been considering for resources. However, in a system of derivations for entailment alone like the one we will actually use, each gap has just one active goal, which appears just below the gap. Goals at earlier stages of a gap's development (i.e., the goals that are not boxed off from the gap) could be described as "available", but they are not available for any sort of use. In particular, although we can consider all available resources when looking for a way of closing a gap, it is only the active goal and not any earlier one that we consider. (Some of the arguments of 2.3.3 could be used to show that considering all "available" goals would not lead us to count an invalid argument as valid, but looking at derivations in this way would make them less like the patterns of ordinary explicit deductive argumentation, which seem to be focused always on a single conclusion.)

Glen Helman 01 Aug 2013

2.2.s. Summary

1 Principles of entailment can be applied in concert by using the graphical idea of a tree—and in more than one way. conclusion trees provide a natural notation for applying the valid patterns of argument Cnj and Ext. Alternatively, we can consider principles of entailment that state conditions for the validity of arguments that have conjunctions as conclusions or as premises.

2 These principles, too, can be combined in a tree as to show an argument is valid. Such a tree—an argument tree—traces the conditions that must hold for the argument at its root to be valid, but it can also be seen as a guide for finding conclusion trees.

3 In fact, we will use a different, more compact notation for combining principles of entailment—a kind of natural deduction system that we will refer as a system of derivations. This notation presents the project of showing that an entailment holds as the task of closing a gap between its conclusion, which serves as a goal, and its premises, which serve as resources. As we narrow the initial gap (and others that result from it), we develop the derivation. Derivations also have a tree structure displayed in a system of vertical scope lines which indicate the resources and goals relevant to various parts of the derivation.

4 The laws of entailment appear as rules for exploiting resources, planning for goals, and closing gaps. There are rules for each of the patterns of argument that figure in conclusion trees. The key rules for conjunction are Extraction (Ext) and Conjunction (Cnj). Quod Erat Demonstrandum (QED) is used to close a gap when its goal is among its resources, and the symbol ● (a filled circle) marks a closed gap.

5 When a derivation is developed, numbers are used along with the labels for rules to record both the order of the development and the connection between the premises and conclusions of the rules.

6 The branching structure of conclusion trees is replicated in derivations by the system of cross-references provided by stage numbers. And the branching structure of argument trees lies in the way gaps develop, something indicated by the order of stage numbers and the arrangement of scope lines. This structure, together with the proximate argument of each gap (formed from its active resources and its goal), forms an argument tree.

7 Principles of entailment for other logical forms will be associated with further rules. Those for $\top$ and $\bot$ are the rules Ex Nihilo Verum (ENV) and Ex Falso Quodlibet (EFQ), which figure in derivations as rules for closing gaps.

8 We keep track of changes in the information contained in goals and resources by using the scope lines of a derivation to tell in which gaps given resources are available and in which gaps available resources are still active.

Glen Helman 01 Aug 2013

2.2.x. Exercise questions

1. Restate the derivation below in two ways: (i) as a conclusion tree, labeling each horizontal line with the number of the stage at which it is entered, and (ii) as its associated argument tree. That is, do with it what is done with the example in 2.2.6 (ignoring the extra decoration, such as colors and dashed lines, that appeared there).

	$(A \wedge C) \wedge B$	1
1 Ext	$A \wedge C$	2
1 Ext	B	(4)
2 Ext	A	
2 Ext	C	(5)
4 QED	B	3
5 QED	C	3
3 Cnj	$B \wedge C$	

2. Use the system of derivations to establish each of the following claims of entailment:

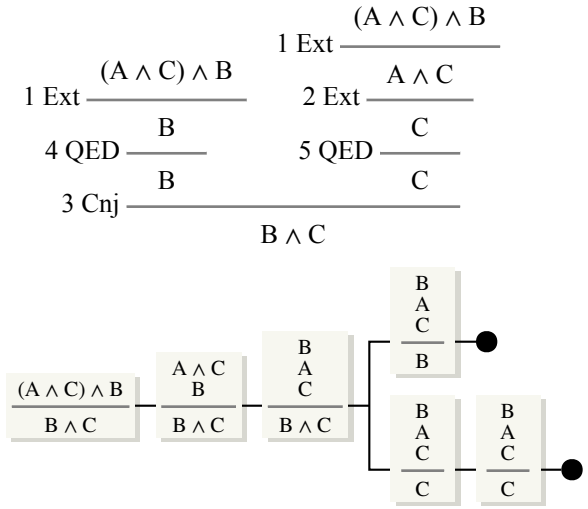
- **a.** $A \wedge B \vDash B \wedge A$
- **b.** $A \vDash A \wedge A$
- **c.** $A \wedge (B \wedge C) \vDash (C \wedge B) \wedge A$
- **d.** $A, B \wedge C, D \vDash (C \wedge (B \wedge A)) \wedge B$
  [The derivation for **d** will have three premises above the initial horizontal line.]
- **e.** $A \wedge (B \wedge C) \vDash (B \wedge A) \wedge (C \wedge A)$

For more exercises, use the exercise machine.

Glen Helman 01 Aug 2013

2.2.xa. Exercise answers

1.



2. **a.**

	$A \wedge B$	1
1 Ext	A	(4)
1 Ext	B	(3)
3 QED	B	2
4 QED	A	2
2 Cnj	$B \wedge A$	

b.

	A	(2),(3)
2 QED	A	1
3 QED	A	1
1 Cnj	$A \wedge A$	

c.

	A ∧ (B ∧ C)	1
1 Ext	A	(7)
1 Ext	B ∧ C	2
2 Ext	B	(6)
2 Ext	C	(5)
5 QED	C	4
6 QED	B	4
4 Cnj	C ∧ B	3
7 QED	A	3
3 Cnj	(C ∧ B) ∧ A	

d.

	A	(7)
	B ∧ C	1
	D	
1	B	(6)
1	C	(5)
5 QED	C	3
6 QED	B	4
7 QED	A	4
4 Cnj	B ∧ A	3
3 Cnj	C ∧ (B ∧ A)	2
	B	2
2 Cnj	(C ∧ (B ∧ A)) ∧ B	2

e.

	A ∧ (B ∧ C)	1
1 Ext	A	(7),(9)
1 Ext	B ∧ C	2
2 Ext	B	(6)
2 Ext	C	(8)
6 QED	B	4
7 QED	A	4
4 Cnj	B ∧ A	3
8 QED	C	5
9 QED	A	5
5 Cnj	C ∧ A	3
3 Cnj	(B ∧ A) ∧ (C ∧ A)	

Glen Helman 01 Aug 2013