7.5.3. Derivations for universals

The implementation of the laws for universal quantifiers is fairly straightforward if we use derivations only in a positive way—i.e., use them only to show that entailments hold. Their use to show that entailments fail will be postponed until 7.6. We do need one general elaboration of our system of derivations that we will use to manage general arguments. The portion of a derivation that constitutes a general argument will be marked by a scope line that is *flagged* by the parametric term on which we generalize (as shown in Figure 7.5.3-1). This flagging declares that the term is parametric. Indeed, we will require that a term flagging a scope line appear only to its right, so the scope line will mark the scope of the term's use. This is more than is necessary to stay in accord with the laws for universals as conclusions. They require only that the term not appear in either the goal or the active resources of the gap that the vertical line spans, but we will never run short of terms and the stronger requirement is far easier to check.

In either form, the requirement is designed to insure that the parameter maintains no ties to the outside of the general argument so that, within the argument, it might refer to anything at all. For this reason, we will speak of a scope line flagged by a term as a *veil of ignorance*; the portion of the derivation marked off by the scope line proceeds without any information about the specific identity of the parameter a.
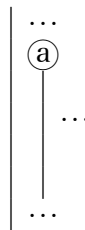

Fig. 7.5.3-1. A veil of ignorance flagged by the parameter a.

Now, let us look at the rules—first those for unrestricted quantifiers. Figure 7.5.3-2 shows the exploitation rule, which we will call *Universal Instantiation* (UI). It can be used to add any instance of the universal as a further resource, notating the universal to indicate the term for which an instance was added.
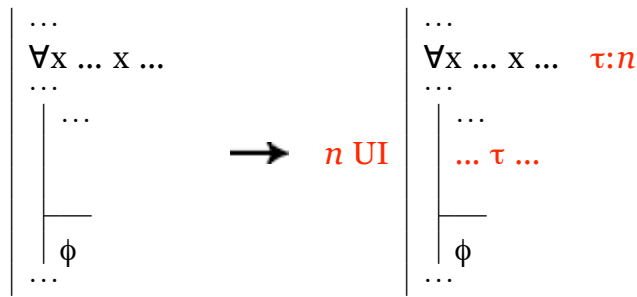
Fig. 7.5.3-2. Developing a derivation at stage n by exploiting an unrestricted universal for a term τ.

Although we record the use of this rule alongside the universal, the universal resource is not rendered completely inactive. The rule provides only a partial exploitation, extracting the content of the universal only for the single term τ. Since a universal does not bring with it any definite set of instances, it will never be rendered completely inactive, no matter how often this rule is used. Still, one use of the rule does exploit the universal for one term, and we record this by noting both the stage number and the term for which the universal has been exploited.

This information is used (much in the way we have used marking by stage numbers) to judge when a universal is **active for** a given term. To be active for a given term in a gap, a universal must be available in the gap and must not have been exploited for the term in the course of narrowing the gap. Specifically, an available universal is **inactive for** τ in a gap if it is marked by a pair τ:n and all scope lines to the left of some resource or goal entered at stage n continue unbroken to the left of the gap. Although an available universal is always active, it may not be active for all terms; and a term for which we apply the exploitation rule above should be one for which the universal is still active.

As we will see in 7.7.4 , the use of this rule may be limited to terms appearing in the available resources and goals the gap. These are the same terms from which we form alias sets and it will be enough to exploit a universal for at least one term from each alias set. But, occasionally, no terms will appear in the initial premises and conclusion and none will be introduced by other rules. When this is so, the exploitation rule above may be used to introduce a new unanalyzed term into the derivation.

At the other extreme, use of this rule in case of generalizations containing functors may introduce new terms into the derivation, leading to new uses of the rule. For example, instantiating ∀x P(fx) for the term a will give us P(fa), which contains the term fa, and we may use this term also to instantiate the generalization, giving us P(f(fa)),

which contains the term f(fa)—and so on. As we will see in 7.6, this is one aspect of a general feature of the deductive logic for generalizations that will sometimes keep a derivation from ever reaching an end. That is not our concern now, but the possibility of going on forever in the application of rules shows that we can no longer wait to apply rules fully before checking to see if a gap closes. And, because a large number of applications of instantiation may be possible, it is wise to select from among the terms with which we might instantiate a generalization those that seem most likely to help us close a gap.

The following two derivations illustrate these points.

	∀x ∀y ∀z Rxyz	a:1		∀x Fx	a:1
				∀x ¬ Fx	a:2
1 UI	∀y ∀z Rayz	b:2, c:4			
2 UI	∀z Rabz	c:3	1 UI	Fa	(3)
3 UI	Rabc	(6)	2 UI	¬ Fa	(3)
4 UI	∀z Racz	c:5		•	
5 UI	Racc	(6)			
6 Adj	Rabc ∧ Racc	(7)	3 Nc	⊥	
	•				
7 QED	Rabc ∧ Racc				

The first derivation keeps the use of UI to a minimum. Only the main quantifier is removed with each use, so three uses are required to reach the bare predication Rabc. Only two more are needed to reach Racc but three would have been required to reach a predication, such as Rccc, which did not have the term a in the first place after R. On the other hand, a full use of instantiation for the terms appearing in the conclusion would have lead to 3 + 3×3 + 3×3×3 = 39 uses of UI (i.e., three to exploit the premise for a, b, and c, three each to exploit the three resources that result, and three each to the nine resources added in that way). A derivation is not damaged by extra uses of UI any more than it is damaged by using Ext to add conjuncts that are not needed later. But, while adding all conjuncts as resources whenever a conjunction was exploited presented no practical problem, using UI in all ways possible would often lead to unmanageably large derivations.

The premises and conclusion of the second derivation above contain no terms at all, so there would be no way of beginning it if we did not instantiate one of them for a new term. This is the only sort of case in which instances need be added for terms new to the gap being developed. The fact that we do so at all reflects the assumption built into our system that there is at least one reference value. The derivation above shows one consequence of this assumption—namely, that *Everything is finished* and *Everything is unfinished* are inconsistent.

Clearly, if there is anything at all, then these two sentences cannot both be true. On the other hand, if we were to drop the assumption that there is something, both sentences could be true. For generalizations are false only when they have counterexamples; and, in a world in which there was nothing, there would be nothing to serve as a counterexample to either *Everything is finished* or *Everything is unfinished*.

Finally, let us look at the planning rule for universal goals. It is known as **Universal Generalization** (UG) and is shown in 7.5.3-3.
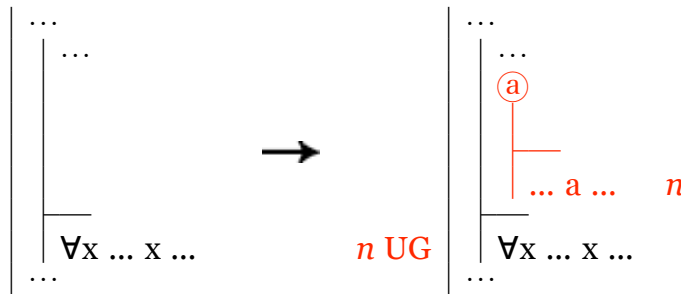

Fig. 7.5.3-3. Developing a derivation at stage *n* by planning for an unrestricted universal; the parameter a may be any unanalyzed term that is new to the derivation.

We try to reach our goal by a general argument, so we choose as our parameter an unanalyzed term a that is new to the derivation. An instance of ∀x θx for the term a is the goal of the general argument, and further development of the gap lies on the other side of a veil of ignorance concerning that parameter.

The short derivation shown below illustrates these two rules.
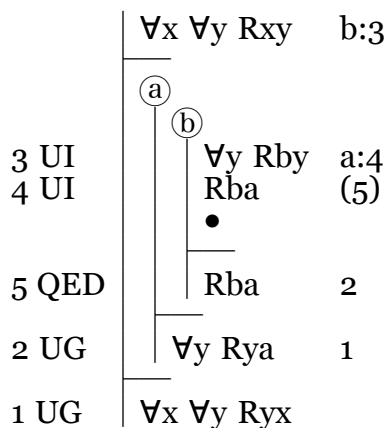
At the initial stage here, there is no vocabulary from which a term may be formed—and UI should be used to introduce new terms only as a last resort—so we apply the planning rule to the universal conclusion. Whenever we apply this rule we must introduce a new term as a

parameter so, when the rule is applied a second time at stage 2, a second new term is introduced. There is now plenty of vocabulary for use with the exploitation rule. It would have been possible to exploit the initial premise twice to add ∀y Ray as well as ∀y Rby to the resources, and we might have exploited each of these resources twice as well. But the two uses of the exploitation rule that are shown above are enough to derive the resources that enables us to close the gap.