6.3.2. A law for aliases

If we include identity itself among the predicates and functors for which identity obeys laws of congruence, it will follow that identity is an equivalence relation provided that it is reflexive. For any reflexive relation that is a congruence for itself will be symmetric and transitive also. Moreover, it is enough to say that identity is a reflexive relation and a congruence for all one-place predicates, simple and complex (i.e., including abstracts), to show that identity is a congruence for all predicates and functors whatsoever. This provides one compact way of capturing the laws for identity; but, when implementing these laws in derivation rules, it will be most convenient to group them together in a different way. We will have two principles, one of which will be the law asserting that identity is a congruence for all simple predicates. The other will be a law that groups reflexivity, symmetry, and transitivity together with the law of congruence for functors.

We arrived at the laws of symmetry and transitivity by understanding an equation $\tau = \upsilon$ to say that the two terms τ and υ are co-aliases, and we might have arrived at the law of reflexivity in the same way since in the usage of *alias* we have adopted here each term is a co-alias of itself. We can extend these same ideas more generally by speaking of terms τ and υ as being ***co-aliases given*** a set Δ of equations or as being ***made co-aliases by*** Δ. Our intention is that this relation capture the conditions under which equations are entailed by other equations.

Clearly a set Δ of equations will imply that an equation $\tau = \upsilon$ is true if either an equation between τ and υ (in either order) appears in Δ or one term can be reached from the other via a series of terms, each of which is linked to the next (in either order) by an equation in Δ. For example, if the equations shown in Figure 6.3.2-1 are in a set Δ, the terms a and e are co-aliases given Δ, as are any other pair of terms appearing in the list. Moreover, we may count each term as an alias for itself given any set of equations.

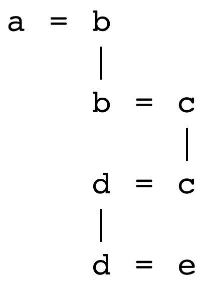


Fig. 6.3.2-1. A chain of equations making terms a and e co-aliases.

Although we have not yet stipulated all the conditions under which we will count terms as co-aliases, we have said enough to summarize the laws of reflexivity, symmetry, and transitivity—and more besides—by stating a

> ***Law for aliases***: $\Gamma \Rightarrow \tau = \upsilon$ if τ and υ are co-aliases given the set of equations in Γ.

Like the law for a premise as a conclusion and a number of other principles we have used, the law for aliases gives sufficient but not necessary conditions for an entailment to hold, so it is stated with *if* rather than *if and only if*. To see why an equation can be a valid conclusion without its component terms being made co-aliases by the premises, note that, while an equation will be entailed by a set of equations only if it equates terms made co-aliases by that set, an equation can be entailed by a set of sentences without being entailed by the equations in the set. (For example, $t = u$ is entailed by the premises $A \rightarrow t = u$ and A, and that set contains no equations at all, only a conditional and an unanalyzed sentence.)

Linking a pair of terms by a chain of equations is not the only way a set might imply that they have the same extension. Recall the law of congruence for an *n*-place functor f

$$\tau_1 = \upsilon_1, \ldots, \tau_n = \upsilon_n \Rightarrow f\tau_1 \ldots \tau_n = f\upsilon_1 \ldots \upsilon_n$$

This tells us that the terms $f\tau_1 \ldots \tau_n$ and $f\upsilon_1 \ldots \upsilon_n$ must have the same extension whenever their corresponding components (i.e., τ_1 and υ_1, τ_2 and υ_2, and so on) do. To incorporate this principle into the law for aliases, we will want to say that two applications of a given functor are made co-aliases whenever their corresponding components are made co-aliases, and we will want to allow this sort of connection between terms to figure as a link in a chain by which further terms are made co-aliases.

Putting all this together, we can give a fuller definition of the idea of co-aliases as follows:

The ***co-aliases given*** a set Δ of equations include pairs of terms of all of the following kinds:

(i) a term paired with itself;
(ii) a pair of terms equated (in either order) by a member of Δ;
(iii) a pair of terms connected by a chain of terms linked as co-aliases given Δ;

(iv) a pair of applications of the same functor whose corresponding components are co-aliases given Δ.

Notice that the third and fourth classes are described in terms of the relation we are defining. A definition like this can be thought as a series of instructions for building the extension of the relation it defines. We first put in all the pairs covered by instructions (i) and (ii). Then we gradually add more pairs as we are directed to by instructions (iii) and (iv), replacing the phrase "co-aliases given Δ" by "pairs already in the extension." A pair of terms then count as co-aliases given Δ if and only if they are added at some stage in this process. And, since this process of building an extension for a two-place predicate can be described without using the term *co-alias*, we really have explained the meaning of that term.

In the simple examples we will usually consider, it will be easy to see which terms are co-aliases given a set of equations. But it may help in understanding the idea to think of the sort of "calculation" we might perform to apply the definition in a more complex example. When checking to see whether a pair of terms τ and υ are co-aliases given a set Δ of equations, let us collect all terms appearing as components in τ, υ, and Δ. Figure 6.3-2 shows these terms for a case where the set Δ consists of the equations a = b, fb = c, fb = fc, d = gca, and g(fa)b = e and we are checking to see whether the terms a and fd are co-aliases.

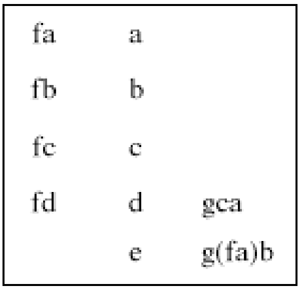
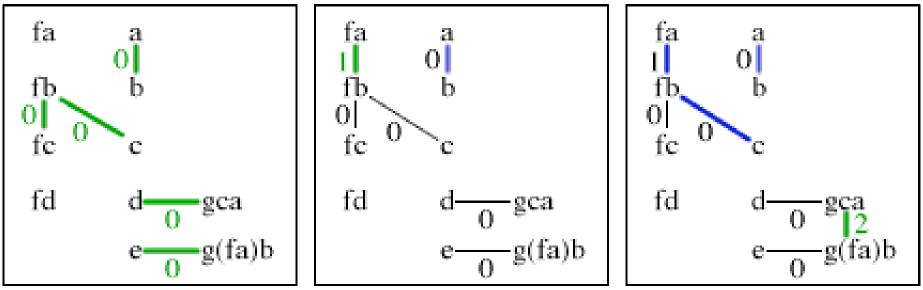


Fig. 6.3.2-2. A work space for finding co-aliases.

Notice that we include fa because it is a component g(fa)b; however, there is no need to include fe or other more complex terms that could be formed from this vocabulary. (The arrangement of the terms is not significant; the one used here is designed simply to make later steps easier to depict.)

Now let us accumulate links between co-aliases. We will represent them as lines between terms. At the initial stage (which we will label 0), we put in links corresponding to equations in the set Δ. We can follow instruction (iii) after this and after each succeeding stage by considering terms to be co-aliases when they are linked either directly or by a chain, so there is no need to draw additional lines. At each of the stages from 1 on, we will consider all functors appearing among the terms and add any links we are directed to by instruction (iv); this will usually require new lines. We may need to do this several times over, but if we add no new links at any stage we can stop because there will be nothing to add thereafter. And, with a finite number of terms, this must happen at some point because there are only a finite number of links we might add.

Figure 6.3.2-3 shows such a process for the example of Figure 6.3.2-2, using labels on links to record the order in which they are entered. The new links at each stage are emphasized along with any older links that lead to the new entry. At stage 1, we check the applications of the functors f and g to see whether we can add any links by instruction (iv). Since a and b were already linked at stage 0, we add a link between fa and fb. We add no other links between the applications of f because d is not linked to a, b, or c. One pair of corresponding terms from gca and g(fa)b (viz., a and b) were connected at stage 0 but the other pair were not, so the link between the two applications of g is entered only at stage 2 after c and fa have also been connected (by the link between fa and fb we enter at stage 1). Even at stage 2 the group including d is not linked to either the groups in which a, b, and c appear, so there are no further links between applications of f and the process is complete. The terms a and fd we were checking do not prove to be co-aliases at the end, but many other pairs of terms were shown to be co-aliases.



Stage 0 Stage 1 Stage 2

Fig. 6.3.2-3. Terms classified as co-aliases in a series of stages.

The links connect the terms in groups shown in Figure 6.3.2-4. The members of any group are co-aliases of one another but not of any other terms.

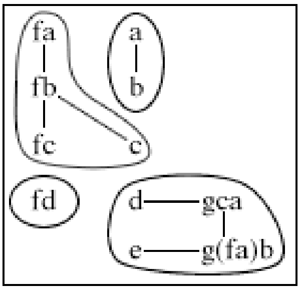


Fig. 6.3.2-4. Linked terms grouped in alias sets.

Some terms, like fd in the diagram, may be groups unto themselves; but, because they are co-aliases of themselves, we can still say that any pair made from such a group is a pair of co-aliases. If the terms had been written down more randomly, the links between them might have crossed and the groups of connected terms would no longer stand out; but they would still be there, and any diagram can be disentangled so that they appear. (This is a distinguishing feature of equivalence relations; any such relation divides a range of values into non-overlapping ***equivalence classes***.) We will refer to each such group of connected terms as an ***alias set***.

Now we are ready to justify our law of aliases, which claims that $\Gamma \Rightarrow \tau = \upsilon$ whenever τ and υ are co-aliases given the equations in Γ. We can do this by showing how this law summarizes earlier ones. Each of the instructions (i)-(iv) for building connections between terms implements one or more of laws of entailment:

	instruction	*law(s)*
i.	enter all terms appearing as components in τ, υ, and the set Δ of equations appearing in Γ	law of reflexivity (since entering the term establishes a link with itself)
ii.	link each pair of terms equated (in either order) by a member of Δ	law for a premise as a conclusion and the law of symmetry (since a link amounts to an equation in both directions)
iii.	count as linked any pair of terms connected by a chain of links	law of transitivity
iv.	link any pair of applications of the same functor whose corresponding components are linked	law of congruence for functors

We combine laws in (ii) and also through carrying out the instructions in a series of stages. This combination of laws can be justified by the law for lemmas because we can think of the process of adding links as a process of adding further equations as lemmas.

Glen Helman 22 Oct 2004