(This isn't really issue 211; it's a problem that came up while looking at issue 211. However, it must be resolved before issue 211 can be.)

Background:

1. Component values in a structure constructor can be specified either positionally or by keyword. Specifying them by keyword has advantages in clarity, but it is approximately twice as verbose in cases where the values are expressed simply.

2. Extended types offer present a different choice in constructors. The "nested" form specifies values for the parent component and any additional components in the type definition. The nested form is always available in the module where the type is defined. The "flattened" form replaces the value for the parent component with the values that would be used to construct such a value in a constructor for the parent type. The flattened form is available if and only if a constructor for the parent type is available in this scoping unit. If the parent type is itself extended, you potentially have the same choice between a nested and a flattened specification of those components. This gives rise to various partially flattened forms and a fully flattened form that goes all the way back to the base type.

3. In paper 00-346, Malcolm provides an example of an ambiguity that sometimes exists between the fully flattened form and one or more of the nested forms. For the ambiguity to exist, the following must be true:

    a. The fully flattened form must be available.

    b. The fully flattened form of one of the ancestor types (typically, the base type) must consist solely of a pointer that can point to that type.

    c. The constructor must be expressed positionally.

    d. The supplied value must be a valid target for such a pointer.

    Malcolm provides a persuasive argument that in such cases, the likely intended meaning is the fully flattened form. His proposal to eliminate the ambiguity is to disallow the position notation except for the fully flattened form.

In other words, to eliminate an ambiguity that exists only under specialized circumstances, Malcolm proposes to disallow the most convenient notation except in the form that is least likely to be available for an extended type! This strikes me as being "too big a hammer" for this problem. I suggest a more limited solution. Ideally, I would simply say that the ambiguous case is interpreted as being the fully flattened form, but that would still make some of the unambiguous forms difficult to parse because you wouldn't know how to interpret the first value until you examined the later values in the constructor. Instead, I suggest that if the fully flattened form is available, the first component in that form is a pointer, the first component value is specified positionally, and the first component value is a valid target, then the constructor is interpreted as being in flattened form.

Under my proposal, if a user wishes to write a nested form, he can either switch to the keyword notation as in Malcolm's proposal, or he can make the first component specification an expression that is not a valid target.

I have not yet worked out the edits for my proposal, but I felt I should at least present the gist of it before the committee adopts Malcolm's "big hammer" approach.

•