Date: Tue, 26 Mar 91 12:02 GMT
From: Brian <UDAA000@HAZEL.CC.KCL.AC.UK>
To: SC22WG11@AWITUW01
Subject: distinct datatypes

This is my two quid's worth. (Things cost more in the UK. I leave it to you to judge if they are worth more.)

It will surprise no-one to learn that I am distinctly uneasy about the tone of some of the recent debate about which types are distinct and which are just minor variants. It sounds to me like insidious encroachment by the minimalist tendency! I have always been an outright maximalist, for CLID to have the richest possible variety of types at the linguistic level. Never mind that some can be simulated in terms of others, by adding constraints or removing or adding operators; that is not a matter for CLID. In my Sigplan 2-valued dataypes paper I termed the gratuitous identification of BOOLEAN with subset of INTEGER as 'abstract weak typing'; saying you can simulate STATE with ENUMERATED or the other way round, or LIST with ARRAY or the other way round, I similarly term 'abstract representationalism'. I don't want any part of it.

I agree with Ed Greengrass that LIST is distinct from ARRAY, though being able to insert into lists is only part of it, as we shall see. I am going to go further and argue that CHARACTER-STRING and BIT-STRING are also distinct from either AND from each other.

LISTS AND ARRAYS

Another (I think the most crucial) distinction between LIST and ARRAY is that it is of the essence that list elements may be atoms BUT MAY ALSO THEMSELVES BE LISTS. Structurally a list is a tree. Arrays are not like that at all.

However an array is essentially multidimensional; vectors are just a special case. A list is, however, at the top level is essentially one- dimensional.

Incidentally, though I think lists are not inherently homogeneous, as arrays are, the base-type definitions in CLID are sufficient that this does not matter. There is sufficient justification for some applications for homogeneous lists that it is best to leave that to the base type.

CHARACTER-STRING

Now let's look at character strings. At the element level they are array like because they are homogeneous - single characters. Substrings are like array partitions, and indeed some languages use indexing to identify them. However, they are one-dimensional only; assemblages of character strings could be arrays, lists or tables but lists are probably the most common and ´natural´ - look at book contents or indexes and you will see lists of character strings with some entries being sublists. Nothing array-like there. Character strings also have insert and concatenate, again list-like.

You can regard CHARACTER-STRING as an array/list hybrid if you like, but it is not enough one or the other to identify it firmly as either. Making an arbitrary choice (at the 'abstract representational') level may be convenient at language design or implementation level but I don't think CLID should pronounce one or the other. I think the hybrid is sufficiently distintive to warrant a separate identity of its own.

BIT-STRING

Again these are atomic only, and you have substrings, and though insert and concatenate are hardly of much importance superficially they do appear to be 'like' character strings, hence presumably their name.

However, their operations of SHIFT-LEFT, SHIFT-RIGHT, and MASK are not character string like or even, apart from the last, bit-like. Most important of all, however, is that THEY ARE INHERENTLY MULTIDIMENSIONAL and the concept of lists of bit-strings makes little sense to me in the way that lists of character strings do.

So I contend that this again is distinct from arrays, lists and (character) strings and suggest that a better name for it is BIT-MAP. I do not just mean a two dimensional bit map in printed or plotted output, but three-dimensional ones (sculptures, holographic images), four-dimensional (generated images of klein bottles, space-time diagrams) and however much higher you want to go.

I hope all maximalists will rally to the call: but these are NOT fabricated distinctions, they are inherent, so I hope others will also accept the logic of the case I have advanced.

Brian Meek