CS-Note* 84

Bill J. Ellis
October 29, 2004

Praxis trip III: Notes

1 Base types matter

As in source code, assertions must respect types. Two integer types can only be compared if they share the same base type. This extends to attributes of these types also. Casting must be used to allow values of different base type to be compared.

2 Loop termination

Implementation of the loop termination heuristic stuttered in the search for a suitability general algorithm. This is not required. The simple exit condition (exit on inequality or equality) need only be considered. This special case can be implemented with a less general but more attainable algorithm.

3 Set based analysis

This is seen as the upper limits of our program analysis. It would be good to explore this as an example of how diverse and sophisticated the program analysis can become. The table utilities (find) example is a good candidate for exploring this.

4 For loops are weird

4.1 Is unchanged

In Ada, and thus SPARK, for loop exit conditions are evaluated only on entry. Thus any variables in a for loop range are translated to special 'entry' variables inside the loop. These can be referred to in annotations by postfixing the variable with a percent symbol (`X%`). Where `X` does not change in the loop (which is nearly always the case) it is possible to add the assertion `X% = X`. This can be used to relate any assertions involving `X` to the input entry values. Note that the percent qualifier only apples to the top level variables.

4.2 Type on entry

Type can be added for all values at the entry to a for loop. These will refer to the percent tagged variables, and thus is only applicable to variables referred to in the for loop range. The Examiner seems less happy to generate this information - due to subtle complexities of the semantics of the subrange facility (so subtle they are not known, but probably rooted in for loop ranges being allowed to be null).

4.3 Type (bounds) in loop

Type (bounds) can be added for all variables in a for loop as normal. The Examiner is very good at adding type for these variables. Note that this will essentially repeat the type on entry if the variable is both in the for loop range and is unchanged within the loop. I.e. `X% = X` can be used as a rewrite `X% => X` on the type of `X%` to reveal the same property that would be added for type in loop. Thus this should not be added in these cases.

4.4 Inside range

A for loop variable is inside its range inside the for loop. However, the bounds will refer to the initial (percent) values. To exploit this property it is important that the above properties are added, relating the percent entry values with the actual loop values.

*Notes in this series are for ϵ-baked ideas, for $1 \geq \epsilon \geq 0$. Only exceptionally should they be cited or distributed outwith the NuSPADE project (EPSRC GR/R24081).