Section 20

Stefen Boyd

Since the only thing that can be randomized is a class, this whole section should be a subsection of 11. At least it should be just after 11. If added to section 11, some of the sections may need to be grouped as this would be a large subsection in contrast with lots of tiny ones.

It seems natural to group this with section 11 since the bnf for 11 will include most all of the rand stuff since it's part of the way classes are declared.

General

Stefen Boyd

If we don't already have an issue open for the random algorithm, we should have one. Granted, constraint solving may not be able to be specified, but given no constraints, the same random members of a class should have the same sequence between different implementations. The random algorithm used for all randomization w/o constraints should be identified (randomize(), $urandom, $srandom, $urandom_range)

Response by David: This whole discussion was already covered in the previous review of this chapter. The decision we made then was that since we would not be standardizing the constraint solver there is no point in standardizing the random functions.

Stefen Boyd

All bnf for the constraints should go into one spot (in 20.4). Unless it's a problem, I'll have Stu make the necessary changes when I give him the BNF syntax boxes.

Stefen Boyd

Several references to "scalar" that should be "singular"

Response by David: See CH-116.

Section 20.5

Stefen Boyd

Sections 20.5 through 20.12 should all be 20.4.1 through 20.4.x since they relate to constraints. 20.16 through 20.18 should also be moved to subsections of 20.4.

Section 20.6

Stefen Boyd

Inheritance: The last bullet doesn't belong here. It should go in 20.13.2.

Section 20.7

Stefen Boyd

Paragraph beginning with "*value_range_list*" ends with statement that "The bound to the left of the colon MUST be less..." What if it isn't? Is that an error or do you get the empty list? Is this only a compile check? Is it possible to get variable values in ranges that would violate at runtime?

Section 20.13.2

Stefen Boyd

pre/post_randomize(): Super is being checked in if statement to see if it's true. Seems that we need consistency between object handles and the handle type. I like this shortcut for "super!=null" (that is what it means isn't it?), but if we can do it here, we should be able to do it with the handle data type.

Section 20.15.1

Stefen Boyd

All the rest of the testbench donation avoided creating reserved words "on" and "off" but here they are again with $constraint_mode and $rand_mode. Someone (Arturo) needs to come up with alternate syntax soon (I need it for the bnf).

It would appear that these should simply be methods for use on either an object or it's members. Especially since they can't be used with multiple objects at the same time. How about rand_on() rand_off, constraint_on(), constraint_off()?

Section 20.16.1

Stefen Boyd

All the rest of the testbench donation avoided creating reserved words "on" and "off" but here they are again with $constraint_mode and $rand_mode. Someone (Arturo) needs to come up with alternate syntax soon (I need it for the bnf).

It would appear that these should simply be methods for use on either an object or it's members. Especially since they can't be used with multiple objects at the same time. How about rand_on() rand_off, constraint_on(), constraint_off()?