REPLACE section 7.17

7.17 Set membership

SystemVerilog supports singular value sets and set membership operators.

The syntax for the set membership operators is:

```
inside_expression ::=                                          // from Annex A.8.3
        expression inside { range_list }
      | expression insidez { expression_list }

range_list ::= value_range { , value_range }

value_range ::=
        expression
      | [ expression : expression ]
      | [ expression : $ ]
      | [ $ : expression ]

expression_list :: = expression { , expression }
```

Syntax 7-2—inside expression syntax (excerpt from Annex A)

The *expression* on the left-hand side of the **inside** or **insidez** operator is any singular expression.

The set-membership *range_list* on the right-hand side of the **inside** or **insidez** operator is a comma-separated list of expressions or ranges. If an expression in the list is an aggregate array, its elements are traversed by descending into the array until reaching a singular value.

A range is specified in ascending order with a low and high bound enclosed by square braces `[ ]`, and separated by a colon (`:`), as in `[low_bound:high_bound]`. A bound specified by **$** shall represent the lowest or highest value for the type of the expression on the left-hand side. When specifying a range, the expressions must be of a singular type for which the relational operators are defined. If the bound to the left of the colon is greater than the bound to the right, the range is empty and contains no values. Values can be repeated, so values and value ranges can overlap.

Both the **inside** and the **insidez** operators evaluate to a self-determined 1-bit value. The **inside** operator uses the equality (==) operator to perform the comparison. It evaluates to 1'b1 if the expression on the left-hand-side is contained in the *range_list* set (i.e., the expression is equal to *at least one* of the values in the set), and to 1'b0 if it is not contained in the set (i.e.the expression is not equal to all of the values in the set), otherwise it returns 1'bx.

The **insidez** operator performs the same as the **inside** operator, except that expressions are restricted to integral values, does not allow ranges with X and Z values, and uses the wildcard equality (=?=) operator to perform the comparison.

For example:

```
int a, b, c
if ( a inside {b, c} ) ...
bit ba = a inside { [16:23], [32:47] };

int x, y, array[10];
if( (x + y) inside { [1:5], array, 10 } ) ...
```

```
bit [7:0] bus, bus_list[$];
if( bus insidez { 0, 1, , bus_list, 6'b111?? } ) ...
```

REMOVE from section 17.9

— `$inset (<expression>, <expression> {, <expression> } )` returns true if the first expression is equal to at least one of the subsequent expression arguments.
— `$insetz (<expression>,<expression> {, <expression> } )` returns true if the first expression is equal to at least other expression argument. The comparison is performed using casez semantics, so 'z' or '?' bits are treated as don't-cares.

REMOVE from section 22.7 (and from BNF box 22-5)

— `$inset` returns true if the first expression is equal to at least one of the subsequent expression arguments.
— `$insetz` returns true if the first expression is equal to at least one other expression argument. Comparison is performed using **`casez`** semantics, so `z` or `?` bits are treated as don't-cares.

REPLACE BNF in A.8.3

inside_expression ::=
expression **inside** { range_list }
| expression **insidez** { expression_list }

range_list ::= value_range { , value_range }

value_range ::=
expression
| [expression : expression]
| [expression : $]
| [$: expression]

expression_list :: = expression { , expression }