

Note to editor: Shift the numeration of the following subsections accordingly, Expression size system function becomes 22.4, etc.

## 22.3 Type querying system functions

Add syntax box:

```
type_query_function ::= // Not in Annex A
    type_function (expression);
    | type_function (data_type);
type_function ::=
    $isintegral
    | $issequence
    | $isproperty
```

To allow a more efficient implementation of properties and sequences the following type querying functions are provided.

- `$isintegral` returns true if its argument is an integral expression or an integral data type.
- `$issequence` returns true if its argument is a sequence expression, **sequence** type or an integral data type.
- `$isproperty` returns true if its argument is a property expression, **property**, **sequence** type or an integral data type.

All of the above system functions shall have a return type of **bit**. A return value of 1'b1 shall indicate *true*, and a return value of 1'b0 shall indicate *false*.

These functions do not evaluate their arguments, just check their types. The type querying functions may be used in constant expressions, for example, to control conditional generate statements.

Example 1:

```
sequence s ; a ##1 b ; endsequence
```

`$isintegral(s)` returns *false*, while `$issequence(s)` and `$isproperty(s)` return *true*.

Example 2:

```
property p(bit b, property p1);
  generate
    if ($isintegral(p1) begin : simple
      !b || p1;
    end
    else begin : complex
      b |-> p1;
    end
  endgenerate
endproperty
```

Depending on the actual arguments the property expression used in an assertion will change as follows:

```
bit a, b;
sequence s;
  b;
endsequence
```

```
a1: assert property(@clk p(a, s));
```

will expand into

```
a1: assert property(@clk a |-> expr);
```

while

```
a2: assert property(@clk p(a, b));
```

will produce

```
a2: assert property(@clk !a || b);
```

Example 3:

```
property wuntil (p, q);  
  generate  
    if ($isintegral(q)) begin : prop  
      property weak_until(p, q); //non-recursive form when q is integral  
      !q[*1:$] |-> p;  
    endproperty  
  end  
  else begin : prop  
    property weak_until (p, q); // recursive form when q is  
                                // sequence or property  
    q or (p and (1'b1 |>= weak_until (p, q)));  
  endproperty  
  end  
  endgenerate  
  prop.weak_until(p, q);  
endproperty
```