This proposal corrects an incomplete fix for item 1381 regarding when an evaluation attempt takes place under inferred enabling condition from an always block.

P1800-2008-draft1, 17.13.5, p. 289 (IEEE Std 1800™-2005)

Replace

Another inference made from the context is the enabling condition for a property. Such derivation takes place when a property is placed in an **`if..else`** block or a **`case`** block. The enabling condition assumed from the context is used as the antecedent of the property. A concurrent assertion embedded in procedural code specifies that a new evaluation attempt of the underlying property_spec begins at every occurrence of the inferred clocking event.

```
property r3;
      @(posedge mclk)(q != d);
endproperty
always @(posedge mclk) begin
      if (a) begin
            q <= d1;
            r3_p: assert property (r3);
      end
end
```

The above example is equivalent to the following:

```
property r3;
      @(posedge mclk)a |-> (q != d);
endproperty
r3_p: assert property (r3);
always @(posedge mclk) begin
      if (a) begin
            q <= d1;
      end
end
```

Similarly, the enabling condition is also inferred from **`case`** statements.

```
property r4;
      @(posedge mclk)(q != d);
endproperty
always @(posedge mclk) begin
      case (a)
            1: begin q <= d1;
                  r4_p: assert property (r4);
            end
            default: q1 <= d1;
      endcase
end
```

The above example is equivalent to the following:

```
property r4;
```

```
      @(posedge mclk)(a==1) |-> (q != d);
endproperty
r4_p: assert property (r4);
always @(posedge mclk) begin
      case (a)
            1: begin q <= d1;
            end
            default: q1 <= d1;
      endcase
end
```

With

Another inference made from the context is the enabling condition for a verification statement. Such derivation shall take place when a verification statement is placed in an **`if..else`** block or a **`case`** block. The enabling condition assumed from the context is used to modify the `property_expr` of the verification statement as follows:

- If the verification statement is assume property or assert property then the enabling condition is used as the antecedent of an overlapping implication of which the consequent is the `property_expr` stated in the verification statement.

- If the verification statement is cover property then the enabling condition is used as the antecedent of a negated overlapping implication of which the consequent is the negated `property_expr` stated in the verification statement. The double negation used in the property expression represents the dual property to the overlapping implication and it is often called "followed by".

The resulting new `property_expr` then replaces the original `property_expr` in the verification statement. A concurrent assertion embedded in procedural code in an **`if...else`** block or a **`case`** block specifies that a new evaluation attempt of the underlying `property_spec` begins at every occurrence of the (inferred) clocking event.

For example,

```
property r3;
      @(posedge mclk)((q != d) ##1 ack);
endproperty
always @(posedge mclk) begin
      if (a) begin
            q <= d1;
            r3_p: assert property (r3);
            r3_c: cover property (r3);
      end
end
```

The above example is equivalent to the following:

```
property r3;
      @(posedge mclk)((q != d) ##1 ack);
endproperty

r3_p: assert property (@(posedge mclk) a |-> r3);
r3_c: cover property (@(posedge mclk) not (a |-> not r3));
```

```
always @(posedge mclk) begin
      if (a) begin
            q <= d1;
      end
end
```

Similarly, the enabling condition is also inferred from case statements.

```
logic [1:0] a;
property r4;
      @(posedge mclk)((q != d) ##1 ack);
endproperty
always @(posedge mclk) begin
      case (a)
            2'b01: begin q <= d1;
                  r4_p: assert property (r4);
                  r4_c: cover property (r4);
            end
            default: q1 <= d1;
      endcase
end
```

The above example is equivalent to the following:

```
logic [1:0] a;
property r4;
      @(posedge mclk)((q != d) ##1 ack);
endproperty

r4_p: assert property (@(posedge mclk)(a===2'b01) |-> r4);
r4_c: cover property (@(posedge mclk) not((a===2'b01) |-> not r4));

always @(posedge mclk) begin
      case (a)
            1: begin q <= d1;
            end
            default: q1 <= d1;
      endcase
end
```