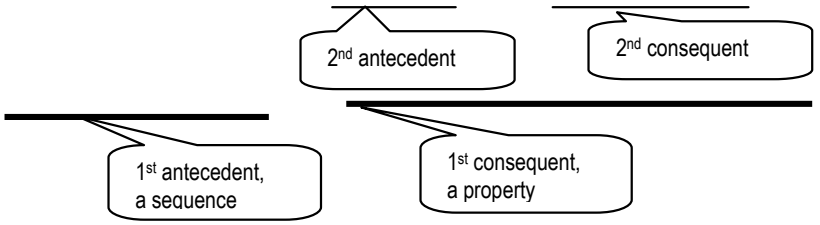
This rule is important because unexpected result may occur. This is also particularly true when multiple threads update a common local variable as each thread carries its own set of local variables. Those issues are explained, with examples, in Section 2.7.9.

The **first_match()** DOES NOT NECESSARILY GUARANTEE EXCLUSIVITY IN FIRST MATCHES

🔔 **Guideline:** Avoid unnecessary successful start of attempts by ensuring that the start of the triggering condition is unique. Use the **$changed, $rose**, or **$fell** functions when appropriate If an antecedent sequence can spawn multiple threads, and it is desired to ensure a unique first match, use the **first_match** operator. This guideline also applies to circumstances where the consequent of the first antecedent is a property that itself has an antecedent sequence with multiple matches. For example,

```
a ## 1 b[*1:2] ##1 c |=> ##[0:$] d |->  e [*2:3] ##1 f;   // 👎👎
```



For uniqueness, use the **first_match** and edge detect function such as: // 👍👍

```
first_match($rose(a) ## 1 b[*1:2] ##1 c) |=>
     first_match(##[0:$] d) |-> e [*2:3] ##1 f;
```

Also, see guidelines for the *goto* repetition operator in Section 2.3.4.

2.5.2 End point of sequences, .triggered

📖 **Rule:** *[1] The end point of a sequence is reached whenever the ending clock tick of a match of the sequence is reached, regardless of the starting clock tick of the match. The reaching of the end point can be tested by using the method* **triggered**. *The syntax of the* **triggered** *method is as follows:* `sequence_instance.`**triggered**, *where* **triggered** *is a method on a sequence. The result of its operation is true or false. When method* **triggered** *is evaluated in an expression, it tests whether its operand sequence has reached its end point at that particular point in time. The result of triggered does not depend upon the starting point of the match of its operand sequence.*

Thus, an end point of a sequence is a Boolean expression (True/False) that represents the evaluation of a thread of a sequence at its last cycle. Every time a thread of sequence completes it represents an endpoint for that thread. Thus, a multi-threaded sequence can have multiple endpoints. The value of sequence method (**triggered**) would be true at these endpoints. Applying a **first_match** method on a `sequence_instance.`**triggered** does not make sense, and is equivalent to `sequence_instance` (see guidelines below for an example).

An end point can be used as a seed to test other assertions, as a reset / accept / reject of a property, or as a trigger to a procedure. For example, with the requirement that once a fetch from main memory is completed (end point here) data must be sent to the slave unit via a handshaking mechanism. This can be expressed as follows, ignoring the data values.

```
default clocking cb_clk @ (posedge clk);  endclocking
sequence get_main_mem;  // clocking event needed because of end point
     first_match(@(posedge clk) rd ##[1:10] valid); endsequence
ap_mem2slave: assert property( // sequence infers clocking event rom the context
      get_main_mem.triggered |=> bus_req ##1 ack ##2 done);
```

Rule: An assertion sequence can infer a clock in other contexts, such as from a **default clocking** or from a procedural clocking event if the sequence is used inside a procedure (e.g. **always_ff**). Thus, sequence declarations need not have the clocking event specified in the declarations. In addition, the triggered and matched sequence methods could be used on instances of sequences

Guideline: Until tools support this new feature, sequence declarations that will be used with a **.triggered** or **.matched** methods should have the clocking event specified inside the declarations. Thus, the above sequence should be declared as:

```
sequence get_main_mem;  @ (posedge clk)first_match(rd ##[1:10] valid); endsequence
```

Rule: An end point of a sequence is obtained through the use of the **.triggered** method.[27] If an assertion directive has a sequence with a **.triggered** method applied to it, then the end point of the sequence is computed as follows:

1. At every clocking event an evaluation start of that sequence is performed.
   a. If at the start of the evaluation there is no match (i.e., first element of sequence is zero) then the sequence is discarded from consideration for an end point, since there is no possibility of a match.
   b. However, if at the start of the evaluation the first term is a match, and if the sequence is multi-threaded (see 2.3.2), then each thread of the sequence is separately evaluated until it reaches its ending clock tick; thus, there could be multiple end points.
   c. If at any cycle during the evaluation of a thread there is no match then that thread is discarded from consideration for an end point.
2. If at the ending of a thread of a sequence there is a match, then that thread is considered as an end point (i.e., method returns true).
3. If at the ending of a thread of a sequence there is no match, then that thread is not considered as an end point (i.e., method returns false)..
4. If at the clocking event there is no end point from any thread, then **.triggered** returns false.
5. If at the clocking event there is an end point from any thread, then **.triggered** returns true.

Guideline: To avoid unexpected results when end points are used in an antecedent or in a **wait** statement (see 2.5.4.3), apply the **first_match** function in the declaration of the sequence that is multi-threaded. This is because a **.triggered** or **.matched** function on a sequence that is multi-threaded will produce multiple points, each of which is generated at the termination of each thread with a Boolean evaluation of true or false. Do not use the **first_match** method on a sequence that has the **triggered** method applied to it. For example:

```
//1) end point of a first match of a sequence (ch2/2.5/s1c_ep.sv, s1c_ep.jpg)  ✓
    sequence q_ab; @ (posedge clk) a ##[1:3] b; endsequence
    sequence q_abFM; @ (posedge clk) first_match(a ##[1:3] b); endsequence
    ap_q_abFM_triggered: assert property(@ (posedge clk)  q_abFM.triggered |=> 1); ✓

// 2) first match of end points of a sequence  👎👎
   ap_FMq_ab_triggered: assert property(@ (posedge clk)
        first_match(q_ab.triggered ) |=> 1);    // 👎👎 first_match of end points is meaningless
        //              (q_ab.triggered ) |=> 1)  // SAME AS ABOVE
```

DO NOT USE THIS

[27] The usage of **ended** function is deprecated in IEEE 1800-2009 because the **triggered** meets the requirements of both the IEEE 1800'2005 **ended** and **triggered**