Fretish Semantics

version Oct 2019

References

- Louise E. Moser, P. M. Melliar-Smith, Y. S. Ramakrishna, George Kutty, Laura K. Dillon: **The Real-Time Graphical Interval Logic Toolset.** CAV 1996: 446-449

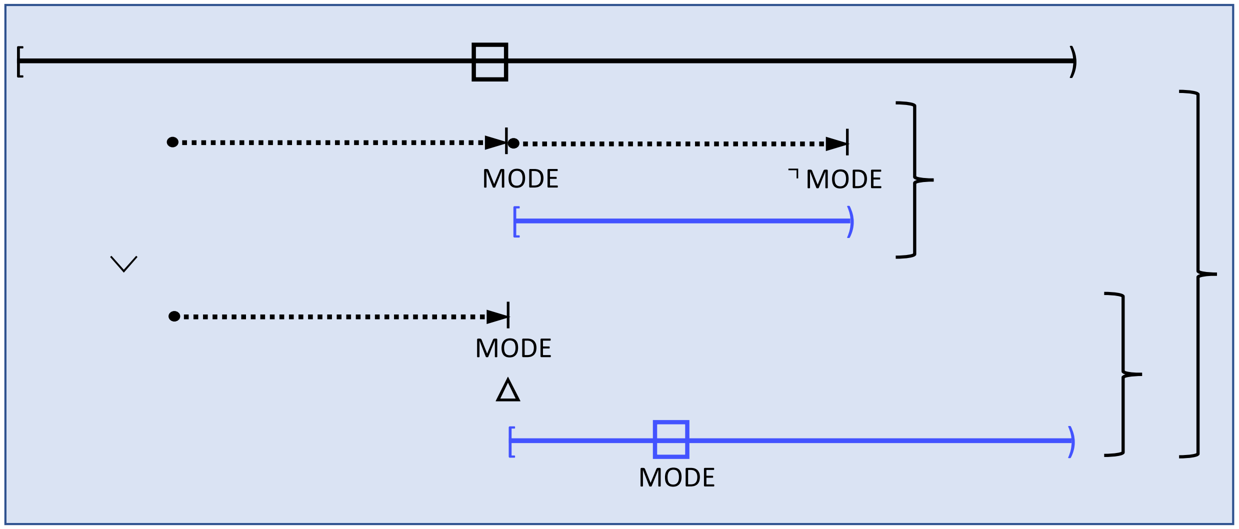
- Laura K. Dillon, George Kutty, Louise E. Moser, P. M. Melliar-Smith, Y. S. Ramakrishna: **A Graphical Interval Logic for Specifying Concurrent Systems.** ACM Trans. Softw. Eng. Methodol. 3(2): 131-165 (1994)

scope semantics – infinite state

In the following slides we define the intervals that are relevant for the scopes described. The formulas for the rest are applied then to each of these intervals with conjunction – but we present them separately. Defined intervals are highlighted in blue.

in

current – equivalent to (after MODE until ˥ MODE)



Notes

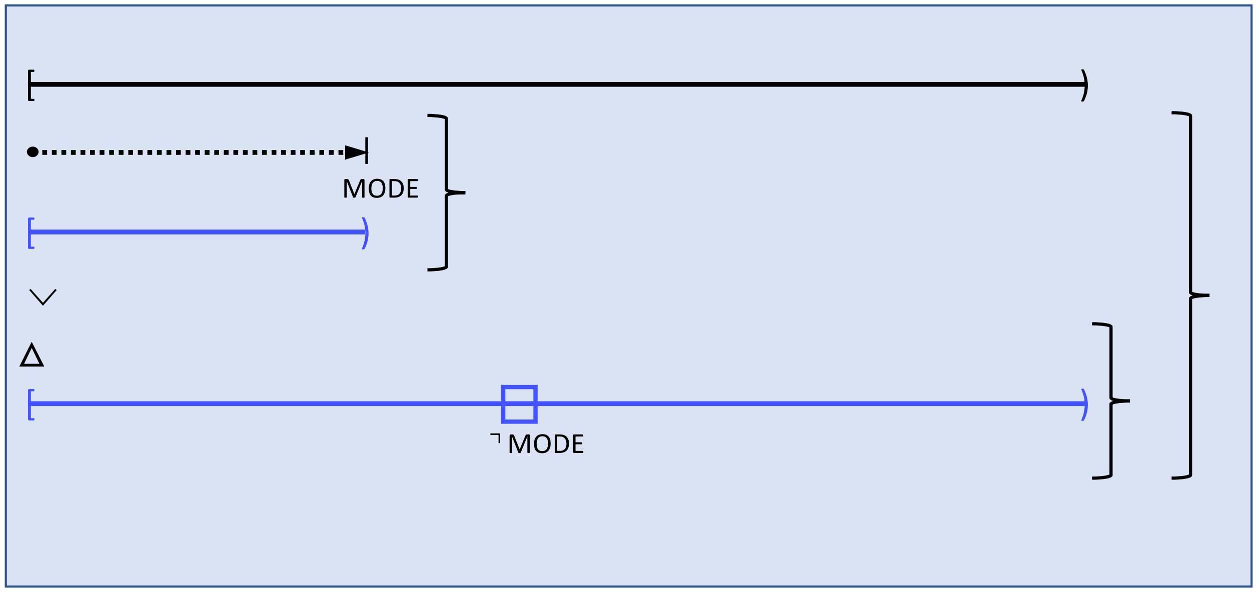
- using box here to stress the invariant (like Kansas State patterns) although in original box is not needed for always.
- Interval not defined if MODE never occurs in first interval formula
- Second interval formula adds the possibility to extend for ever when mode does not end and is only applicable to the after until semantics

- Options:
  - in: {afteruntil, between}

alternative – ignores interval that starts but does not end
equivalent to (between MODE and ˥ MODE)
To get this we simply remove the second part of the disjunction

before

current – if mode never occurs we still check.



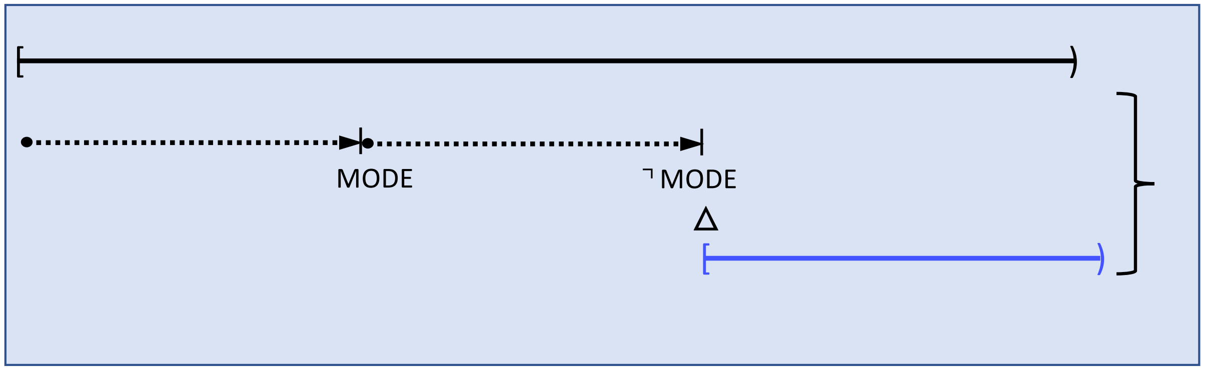
Notes

- If mode happens at beginning of execution then interval is not defined and all properties are trivially true
- In our implementation we use weak to capture the second part of the disjunction
- Options:
  - Same as for in because before also sets an interval from FTP to particular point

alternative – we simply remove the second part of the disjunction

after

current – if we never exit the mode then no requirement is imposed



Notes

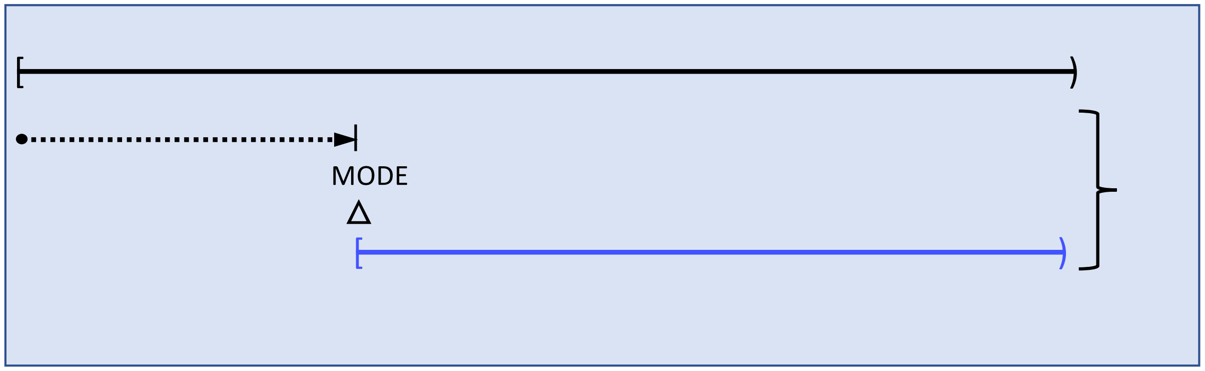
- If mode never happens all properties are trivially true
- Options:
  - Here we always have an after until semantics – no other option is meaningful

only scopes negate the main formulas (see defs later)

- only in = same as in ¬MODE
- All only modes have a special way of negating the formula that follows – we discuss this when we present the timing-satisfaction

only before

current – if mode never occurs we ignore

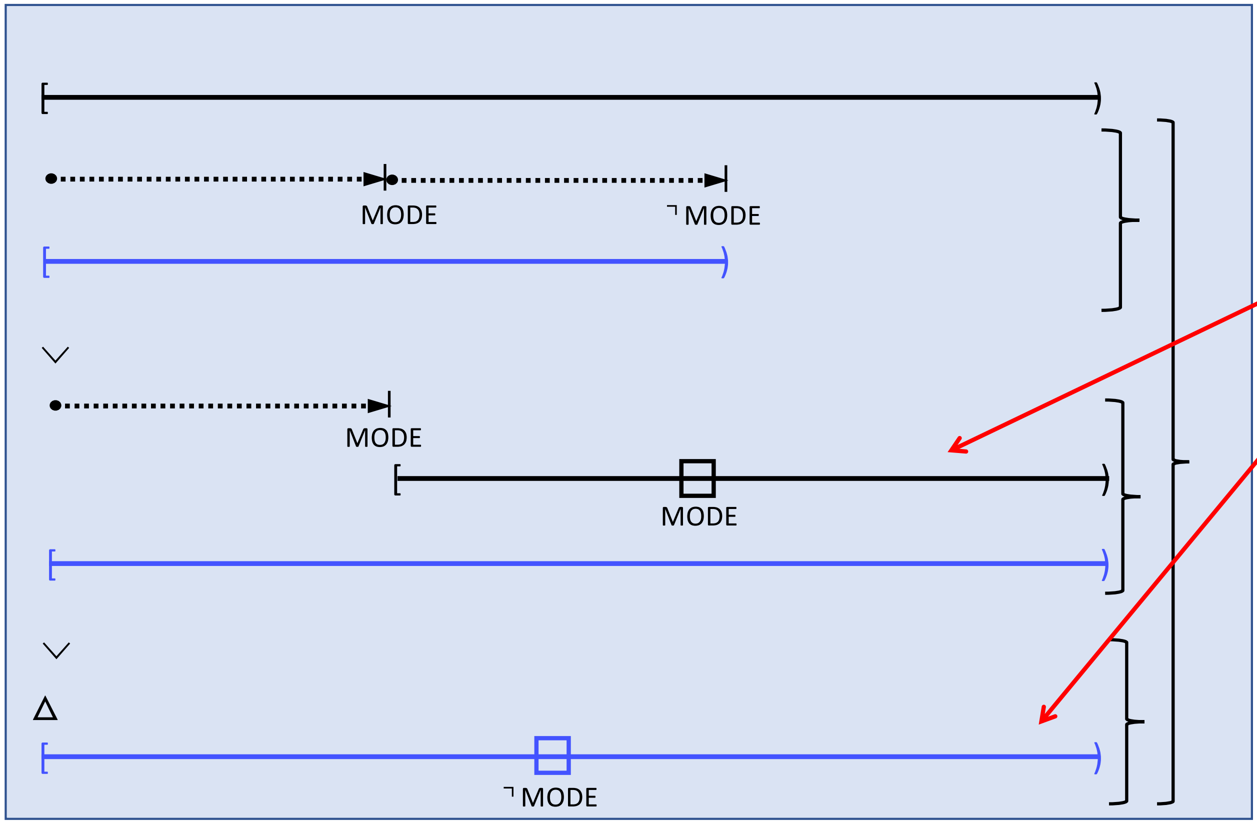


Notes

This makes sense for all cases; we should not differentiate alternative.

only after

current – if mode never exits we still check.



alternative – we simply remove the third part of the disjunction

Notes

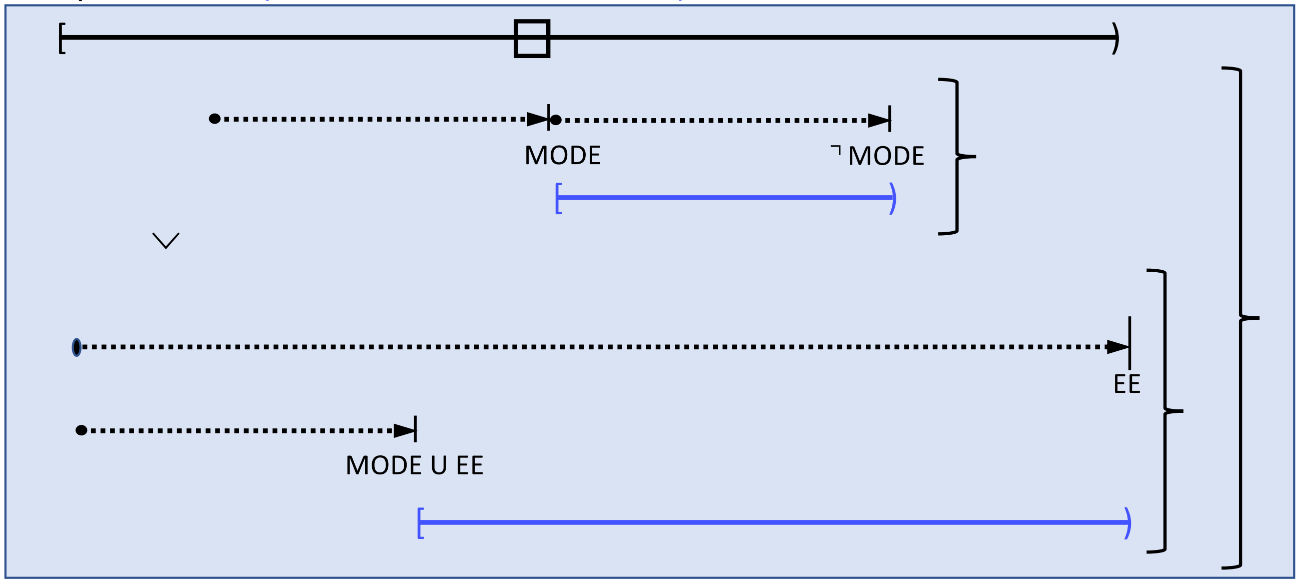
- If mode never exits we check
- If mode never happens we check (consistent with semantics of before – must similarly use “weak” in SALT)
- There is no special case (except third) for “(not mode) happens at FTP” because we only care if it follows a mode and this can never occur at FTP (this refers to our implementation) .

scope semantics –finite state

In the following slides we define the intervals that are relevant for the scopes described. The formulas for the rest are applied then to each of these intervals with conjunction – but we present them separately. Defined intervals are highlighted in blue.

in

current – considers interval that starts but ends at EE (execution ended)
EE flags the time point after the last time point of the execution
equivalent to (after MODE until ¬MODE)



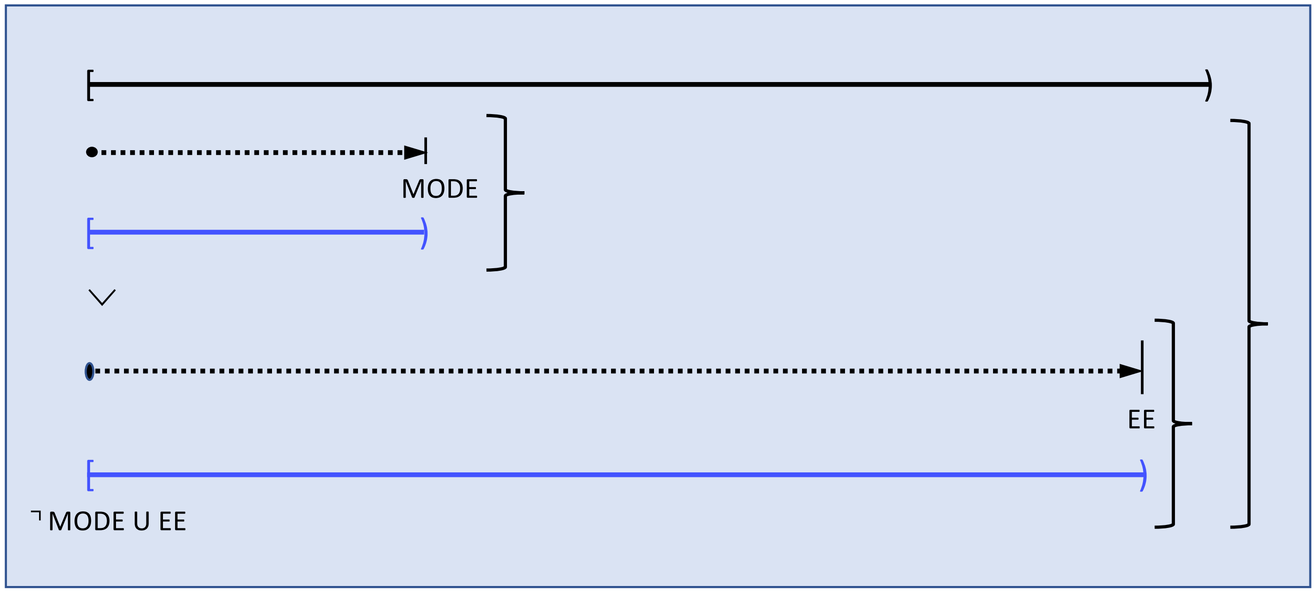
Notes

- using box here to stress the invariant (like Kansas State patterns) although in original box is not needed for always.
- Interval not defined if MODE never occurs in first interval formula
- Second interval formula adds the possibility to extend for ever when mode does not end and is only applicable to the after until semantics

alternative – ignores interval that starts but does not end by EE
equivalent to (between MODE and ¬MODE)
To get this we simply remove the second part of the disjunction

before

current – if mode never occurs we still check.



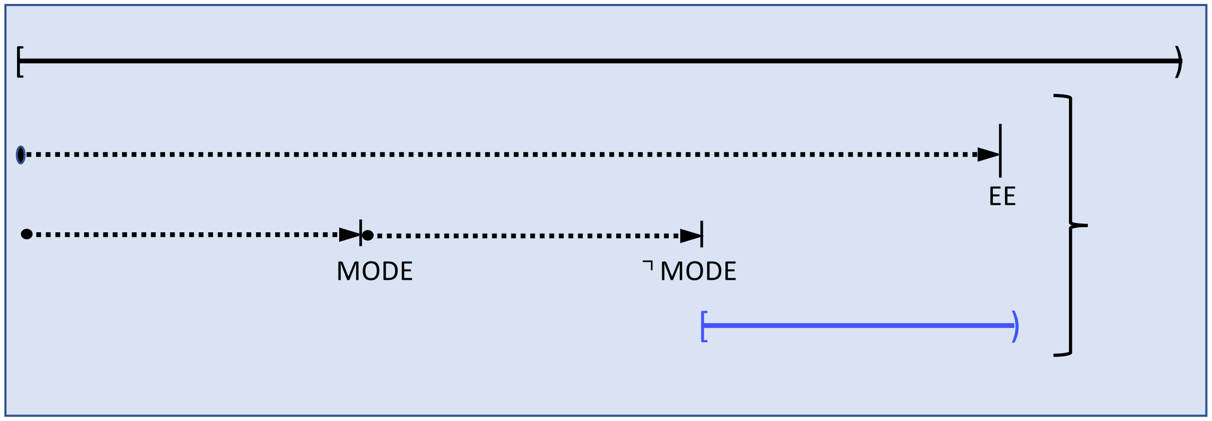
Notes

- If mode happens at beginning of execution then interval is not defined and all properties are trivially true
- In our implementation we use weak to capture the second part of the disjunction

alternative – we simply remove the second part of the disjunction

after

current – if we never exit the mode then no requirement is imposed



Notes

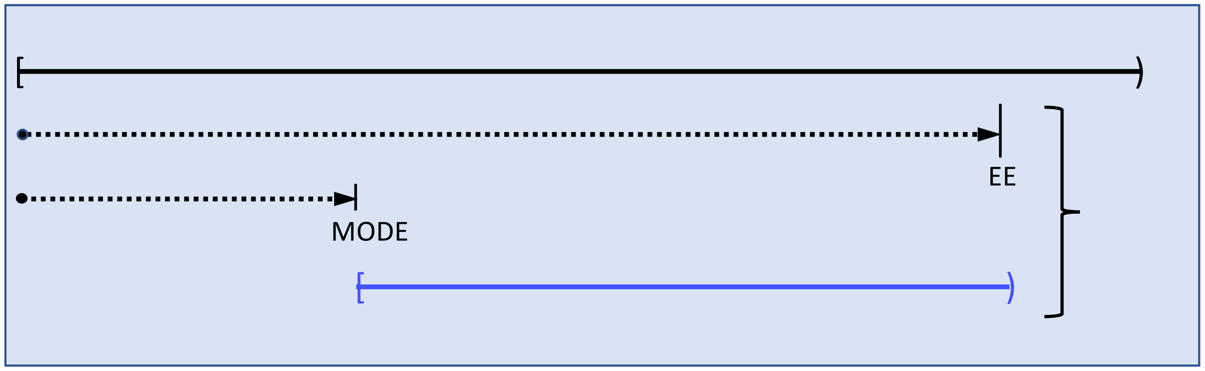
- If mode never happens all properties are trivially true

only scopes negate the main formulas (see defs later)

- only in = same as in ⌝MODE for scope but then (special) negate the formula – more about this later

only before

current – if mode never occurs we ignore

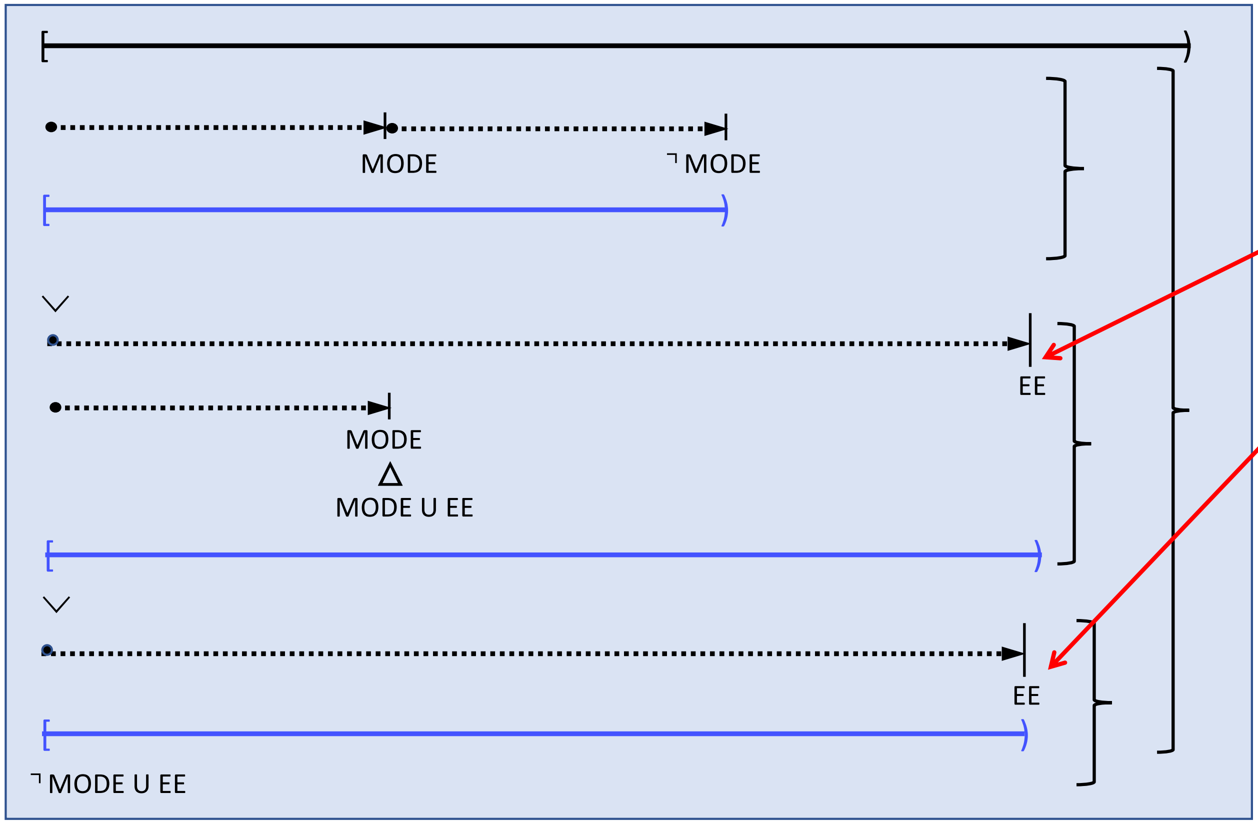


Notes

This makes sense for all cases, we should not differentiate alternative.

only after

current – if mode never exits we still check.



alternative – we simply remove the third part of the disjunction

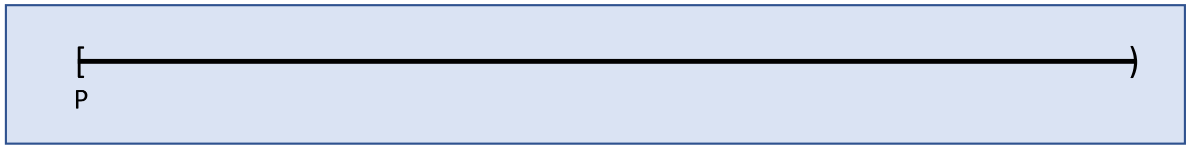
Notes

- If mode never exits we check
- If mode never happens we check (consistent with semantics of before – must similarly use "weak" in SALT)
- There is no special case (except third) for "(not mode) happens at FTP" because we only care if it follows a mode and this can never occur at FTP (this refers to our implementation) .

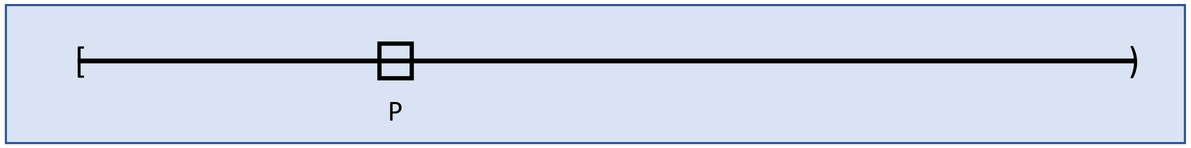
timing semantics – same for finite and infinite

The context that we refer to here are the blue intervals defined by the scope previously, i.e., the semantics here refers to each interval in blue

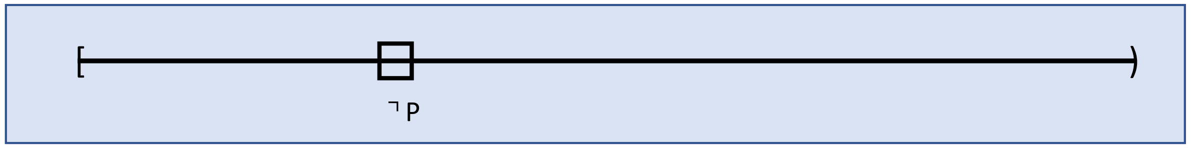
basic timing – no conditions, no real time



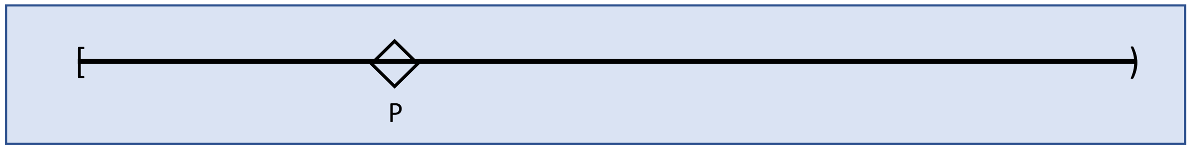
immediately P



always P

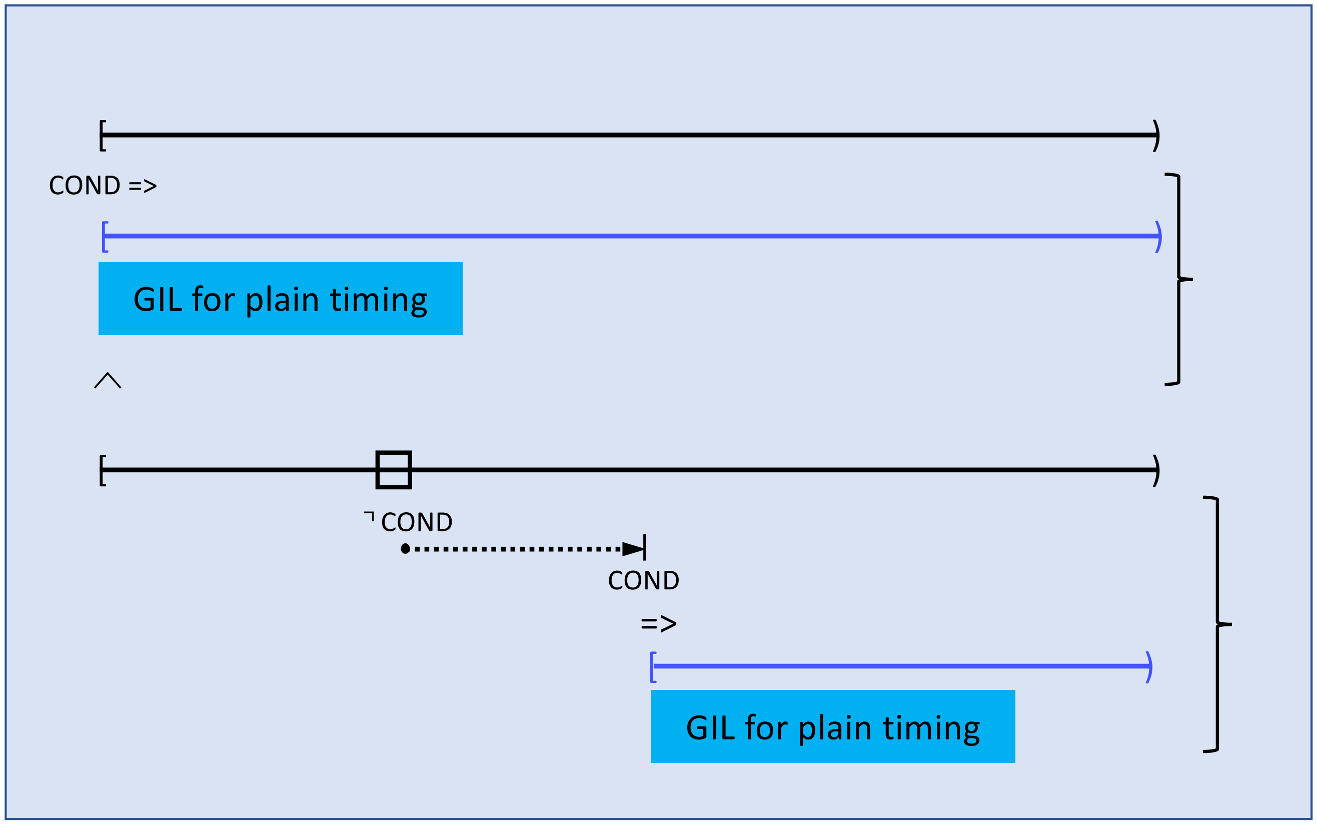


never P

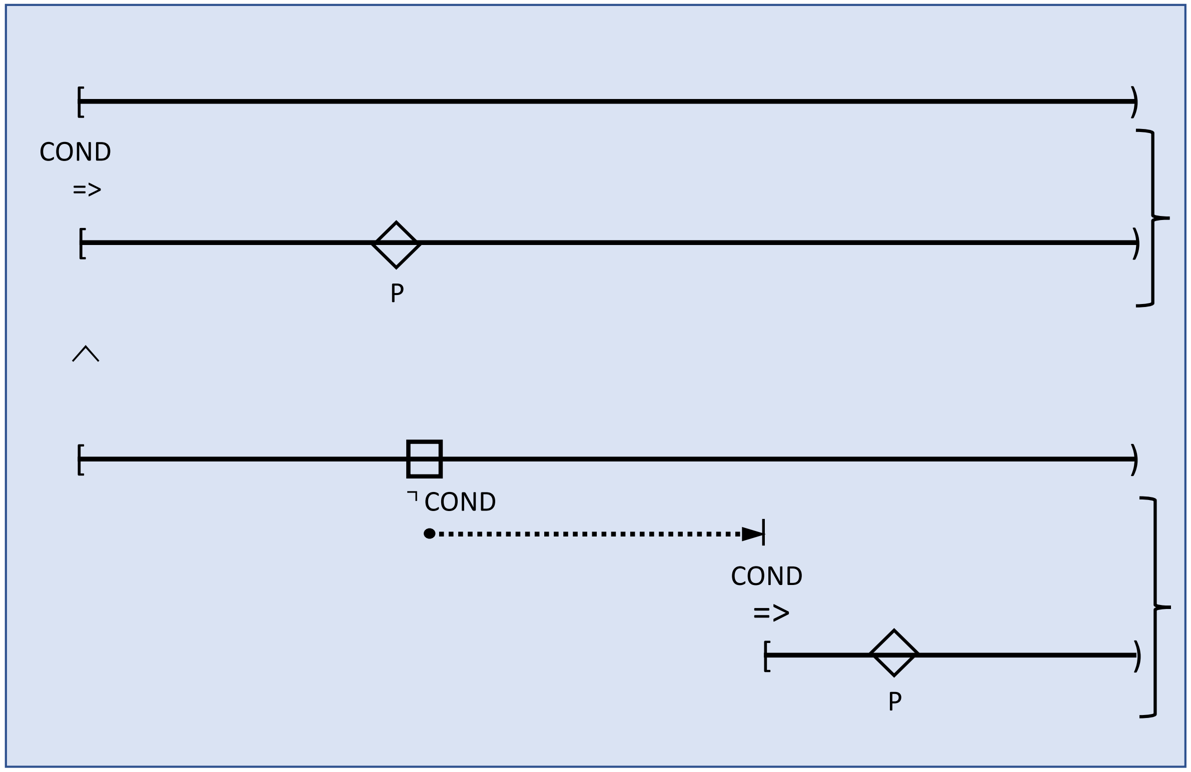


eventually P

adding conditions – pattern



adding conditions – example: eventually P



basic timing – no real time, negations (all cases)

- not (immediately P) = immediately (not P)
- not (always P) = eventually (not P)
- not (never P) = eventually P
- not (eventually P) = always (not P)

real time – same for finite and infinite

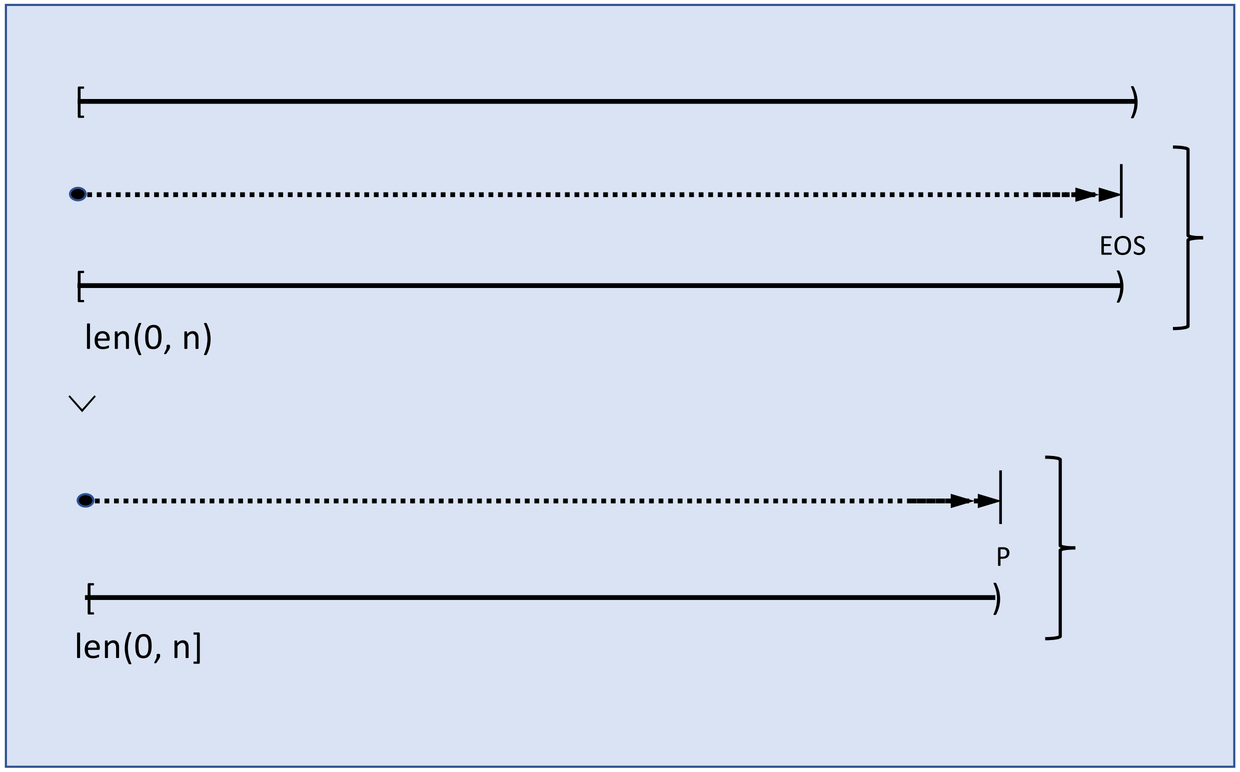
We need to take care of the following plus their negations:

- within n
- for n
- after n

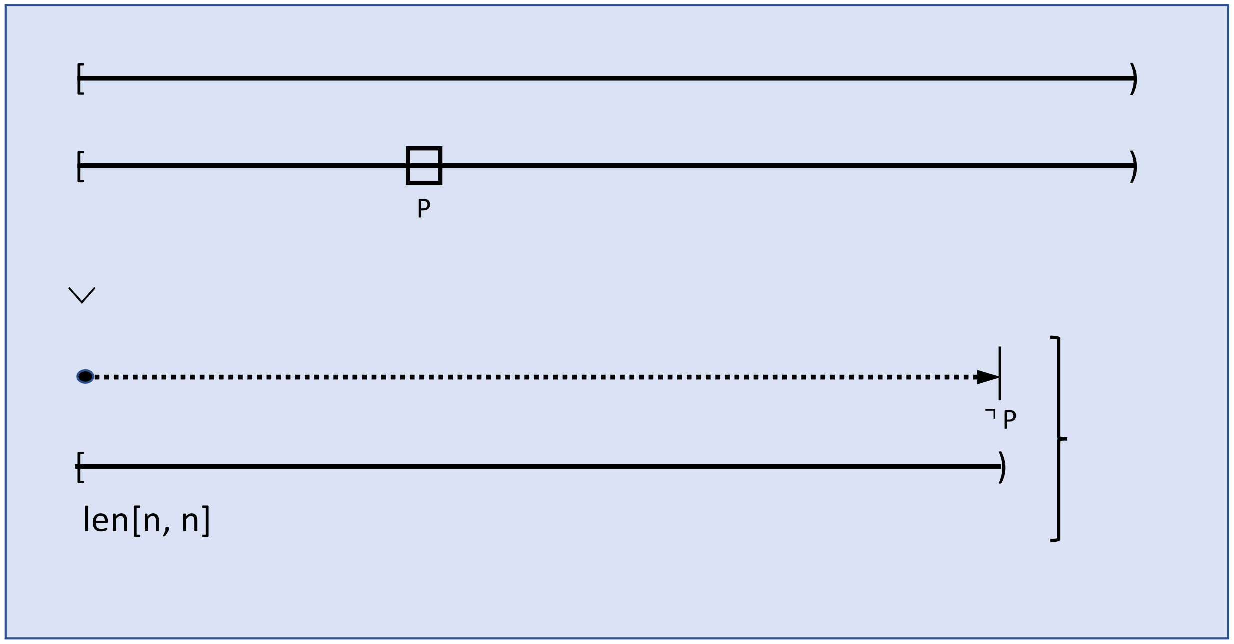
We use EOS (end of scope) to denote the time point right after the end of each scope interval, which coincides with the right point of an interval in the GIL diagrams since these intervals are open to the right: [..).

Conditions simply apply the pattern so no need to write explicitly.

within n P



for n P



Notes

- No double arrow for (not P) happening because P can last for more than n time units

RT timing – negations (all cases)

In general, if an interval is too short, then it is usually not enough evidence of the violation of only.

- not (within n P) = for n (not P)   ✓
- not (for n P) = within n (not P)   ✓
- not (after n P) = (within n P) or (for (n+1) not P) ✓

- EOS is therefore only needed for within, not for, after, not after