Event detection and queue behaviour:

Internal and external events are all posted to the event queue, and processed in time order. Events are detected as follows:

For each state machine:
  For each active transition (outgoing transitions from active state):
    For each transition discontinuous equation:
      Ensure positive zero crossing (z > last(z)):
        Calculate the zeros for z, evaluate the state condition for all zero crossings. Post an internal zero crossing event for all zero crossings where the state condition evaluates true.

        If no zero crossings were detected, but z is positive by some tolerance (related to nudge tolerance), post an internal non-zero crossing event if the state condition evaluates true.

After collecting all events from all state machines, sort the event queue in chronological order. Remove later duplicate events from the same machine (later time zero crossings).

The event queue is processed before each model executive step. If any event falls between t_last and t_current, the first event (earliest time) is processed. After processing an event, internal events remaining on the queue are tested for validity (retest state condition). If they are no longer valid, they are removed from the queue.

Notes:

Non-zero crossing events must be detected to maintain stability, and the timing inaccuracy is extremely small. Models will misbehave is state events are missed. The most common cause of missing events is events that happen so close together that the time nudge of the first causes the zero crossing of the second to be missed.

All valid internal events are left on the queue to reduce the number of non-zero crossing events, and improve processing accuracy.

Event processing:

1. Time is reset to event time, and all variables and derivatives are interpolated at event time.
2. If the event is a parameter event:
    a. If the parameter change is within some tolerance of the previous value, a reinitialize on step failure flag is set, but NLA is not invoked.
    b. If the parameter change is significant (outside tolerance), NLA is invoked.

3. If the event is a state event, the event is triggered. If the event triggers a state transition (general case unless blocked by a guard condition), the transitioning model will generate an OnTransition() callback.

Transition handling / consistent initialization:

If a state transition occurred, the equation map is updated per the outgoing and incoming equations.

The NLA structure is retrieved from the model. This is the structure where all x and xdot, and y are treated as variables (mix of jacobian and mass matrix). The g equations (discontinuity equations) and z variables (discontinuity variables) are excluded.

MSS is run if output assignment is NULL, or high index problem (further work needed to condition MSS on high index problems, and reuse previous MSS results).

Information is updated from MSS VAL and EAL in the variable and equation maps to support calculation of derivative equations in NLA solver. Note that the derived equations could be evaluated directly by the DAE solver provided MSS augmented equations and variables could be added to solver (maintain square problem).

If a state transition occurred, transition equations (g* and r[]) are added to both equation map, and the NLA structure.

The BLT algorithm is run on the NLA structure. → more in next section

The NLA solver is run to get consistent initial values → more in next section

Residuals for g equations are recalculated using consistent variable results, and z variables are updated to eliminate residual error.

BLT for consistent initialization:

The problem is made square as follows:

For index 1 problems, single edge continuity equations are added for all x (VAL[iVar] >= 0) variables.

For high index problems (augmented vars and equations equations exist), unassigned variables are determined (next step), and the continuity equations are added for these variables (similar to low index case)

Unassigned variables (high index problem only) are determined as follows:

1) Determine which variables are x's (from VAL)
2) Search the assignments for those variables that are not x's, and note the referenced equations
3) Determine which equations were never referenced.
4) Build an edge list including only those equations found in 3, and only those variables found in 1
5) Do an output assignment, stopping when all equations are assigned.
6) Determine which x's didn't get assignments

Continuity equation (single edge equation – constant) assignments in the output assignment are cleared so they can be reassigned.

Output assignment is attempted using previous output assignment. If this fails (generally never happens), the entire output assignment is cleared, and redone from scratch (expensive task because it is recursive). There is tremendous performance advantage to reusing previous output assignments.

If equations were added (g* or r[]), an edge list is built using only those equations and variables referenced by those equations. An output assignment is performed on this addition subset. The subset output assignment is merged into the larger output assignment (existing assignments are overwritten with the subset assignments). Discarded equations are removed from the edge list.

The BLT algorithm is invoked, and various supporting mapping activities are executed.

NLA variable initialization

Initialize dg/dt maximum and minimum bounds for nudge based on tolerances

Solve all blocks. When the g* equation is evaluated, the residual is nudged. dg/dt is calculated using chain rule. delta(g) is calculated based on dg/dt and range limits. if dg/dt < 0, delta(G) is subtracted from the residual, otherwise, delta(g) is added to the residual.

Ensure dg/dt used for g* during block solving has the same sign after solving. If not, the g* must have been evaluated before a reset equation block that would affect the direction of event polishing. In cases of dg/dt sign change, NLA solver is run again. Two passes should be sufficient.

Time is nudged as well based on delta(g) / dg/dt to compensate for nudge in variable(s) affected by the g* nudge.