Proxel-based Simulation Algorithm
Proxel == $P = (m, \tau, t, p)$

- m == the discrete state (marking of the Petri net)
- τ == age intensity vector of the enabled or race age transitions
- t == simulation time
- p == the probability of that combination

Other algorithm elements

- generated proxels are temporarily stored in a queue Q
- $P.x$ == element x of proxel P
- $\pi_m(t)$ == total probability to be in state (marking) m at time t
- $tmax$ == maximum simulation time
- Δ == discrete simulation step size
- T == a transition in the Petri net
  - $T.id$ is the index of the τ vector, that contains the age intensity information for the transition

1: $Q \leftarrow \oslash$

2: addproxel$(m_0, \vec{0}, 0, 1)$

3: WHILE $Q \neq \oslash$

4: $P \leftarrow$ getproxel()

5: $\pi_{P.m}(P.t) \leftarrow \pi_{P.m}(P.t) + P.p$

6: IF $(P.t < tmax)$

7: addproxel($P.m$,update($P.\tau, P.m, \oslash$),$P.t + 1$, $P.p * (1 - \Delta * \sum h_T(\tau))$

8: $\forall T$: IF(enabled$(P.m, T)$)

9: addproxel(succ$(P.m, T)$, update$(P.\tau, P.m, T)$, $P.t$+1, $P.p$*Δ*$h_T(\tau)$)

The following functions are used in the algorithm:

succ(m, T) returns the marking reached from marking m when firing transition T.

enabled(m, T) returns TRUE if transition T is enabled in marking m.

addproxel(P) inserts proxel P into queue Q.

getproxel() deletes a proxel from Q and returns its value.

update(τ, m, T) updates the enabling time vector τ when transition T fires, or is advanced, in marking m.

memory(T) returns memory policy of transition T (i.e. ENABLE or AGE).

0.0.1 Explanation

First we will comment the algorithm line by line:

Line 1: The proxel queue Q is initialized to the empty set.

Line 2: The initial proxel P_0 representing the initial state of the model is inserted into the queue, possibly splitting it into phases for activated phase-type transitions.

Line 3: Loop until the proxel queue is empty.

Line 4: Get the next proxel P from the queue.

Line 5: Add the probability of the current proxel $P.p$ to the solution.

Line 6: Continue only if maximum simulation time $tmax$ has not yet been reached.

Line 7: Add a new proxel, representing the case, that the SPN remains in the marking $P.m$.

Line 8: Consider all transitions T that can fire in the marking of the current proxel $P.m$.

Line 9: Add a new proxel to the queue that represents the next marking of the Petri net after the firing of T.

0.0.2 Memory policies

The function update() considers all $j = 1...n_\tau$ supplementary variables, which are active in this marking, and modifies their values according to certain rules. The function update() modifies the j-th element of the vector τ of the transition T, according to the behavior of the j-th transition as follows:

update(τ, m, T)

FOR $j = 1$ TO n_τ DO

IF ($j = T.id$) $\tau_j \leftarrow 0$

ELSEIF (enabled(m, T_j)) AND enabled(succ(m, T),T_j)) $\tau_j \leftarrow \tau_j + 1$

ELSEIF (memory(T_j) = AGE) $\tau_j \leftarrow \tau_j$

ELSE $\tau_j \leftarrow 0$