Original Approximate Algorithm

The first attempt at the approximation

- Differing dimensions can be processed separately.
- Potential for parallel processing over dimension and within a geometric cell.
- Reactions (rules) are spatially local and may be fired out of order at the cost of accuracy.

Original Approximate Spatially Embedded Hybrid Parameterized SSA/ODE Algorithm

while $t \leq t_{max}$ **do**
  foreach *dimension* $d \in \{D_{max}, D_{max}-1, \ldots, 0\}$ **do**
    using function φ *map rule instances to the geocells of the expanded cell complex*;
    ParFor expanded geocell $c_i \in ExpandedCellComplex(d)$ **do**
      run Exact Hybrid Parameterized SSA/ODE algorithm for Δt in c_i;
  $t += \Delta t$;

Algorithm 2

- Good for larger simulations.
- Requires more details to fully address the problem.

(Medwedeff and Mjolsness, 2023)

Approximate Algorithm

An improvement on the original

- Serial version is algorithm used in DGGML.
- Key additions:
    - Match data structure
    - Incremental update
    - Synchronization
    - Rule recomputation
- Has potential to be scaled to large problems.

Improved Approximate Spatially Embedded Hybrid Parameterized SSA/ODE Algorithm

initialize the match data structure with all rule instances;
while $t_{global} \leq t_{max}$ **do**
  foreach *dimension* $d \in \{D_{max}, D_{max} - 1, \ldots, 0\}$ **do**
    using function φ *map rule instances to the geocells of the expanded cell complex*;
    ParFor expanded geocell $c_i \in ExpandedCellComplex(d)$ **do**
      $t_{local} = t_{global}$;
      factor $\rho_r([x_p],[y_q]) = \rho_r([x_p]) * P([y_q] \mid [x_p])$;
      while $t_{local} \leq t_{global} + \Delta t_{local}$ **do**
        initialize SSA propensities as $\rho_r([x_p])$;
        initialize $\rho^{(total)} := \sum_r \rho_r([x_p])$;
        initialize $\tau := 0$;
        draw effective waiting time τ_{max} from $\exp(-\tau_{max})$;
        while $\tau < \tau_{max}$ *and* $t_{local} \leq t_{global} + \Delta t_{local}$ **do**
          solve ODE system, plus an extra ODE updating τ;
          $\frac{d\tau}{dt_{local}} = \rho^{(total)}(t_{local})$;
        draw rule instance r from distribution $\rho_r([x_p])/\rho^{(total)}$;
        draw $[y_q]$ from $P([y_q] \mid [x_p])$ and *execute* rule instance r;
        incrementally update match data structure;
    synchronize and remove invalid rule instances from data structure of matches;
  recompute rule level matches (as needed);
  $t += \Delta t_{global}$;

Algorithm 3