Grammar Rules (Extended)

How do we write and interpret them?

- History, dynamic grammars[1]:

$$\{\tau_{\alpha(p)}[x_p]\,|\,p \in L_r\}_* \longrightarrow \{\tau_{\beta(q)}[x_q]\,|\,q \in R_r\}_*$$

with $\rho_r([x_p],[y_q])$

$$\{\tau_{\alpha(p)}[x_p]\,|\,p \in L_r = R_r\}_* \longrightarrow \{\tau_{\beta(q)}[x_q]\,|\,q \in R_r = L_r\}_*$$

solving $\{\dfrac{dx_{p,j}}{dt} = v_{p,j}([x_k])\,|\,p,j\}$.

- Rate function factorization[1]:

$$\rho_r([x_p]) \equiv \int \rho_r([x_p],[y_q])\,\Delta[y_q]$$

$$P([y_q] \mid [x_p]) \equiv \frac{\rho_r([x_p],[y_q])}{\rho_r([x_p])}$$

$$\rho_r([x_p],[y_q]) \equiv \rho_r([x_p]) \times P([y_q] \mid [x_q])$$

- Simplified DGG graph notation[2], where λ is a label vector:

$$G\langle\langle\lambda\rangle\rangle \longrightarrow G'\langle\langle\lambda'\rangle\rangle \quad \text{with } \rho_r \text{ or solving } \dot{x} = v$$

Stochastic Growth Rule:

Left-hand Side (LHS) $(\bigcirc_1 - \bullet_2)\,\langle\!\langle(\boldsymbol{x}_1,\boldsymbol{u}_1),(\boldsymbol{x}_2,\boldsymbol{u}_2)\rangle\!\rangle$

Right-hand Side (RHS) $\longrightarrow (\bigcirc_1 - \bigcirc_3 - \bullet_2)\,\langle\!\langle(\boldsymbol{x}_1,\boldsymbol{u}_1),(\boldsymbol{x}_2,\boldsymbol{u}_2),(\boldsymbol{x}_3,\boldsymbol{u}_3)\rangle\!\rangle$

ρ_r in factored form, where new labels are sampled in the where clause.

$$\textbf{with} \quad \hat{\rho}_{\text{grow}} \times H(\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|; L_{\text{div}})$$

$$\textbf{where} \quad \begin{cases} \boldsymbol{x}_3 = \boldsymbol{x}_2 - (\boldsymbol{x}_2 - \boldsymbol{x}_1)\gamma \\ \boldsymbol{u}_3 = \frac{\boldsymbol{x}_3 - \boldsymbol{x}_2}{\|\boldsymbol{x}_3 - \boldsymbol{x}_2\|} \end{cases}$$

1. (Yosiphon and Mjolsness, 2009); 2. (Mjolsness 2019)

Improving the Exact Algorithm

Proposal for potential points of parallelization

- Parallelizable over propensity calculations.
- Parallelizable over propensity sums.
- Parallelizable over ODE solving when appropriate scaling and resources are available.
- Reactions must be still fired in order.
- Has potential for quick testing of medium-sized systems.
- Could be used to incorporate hierarchical parallelism in the approximate algorithms.

Parallel Exact Hybrid Parametrized SSA/ODE Algorithm

factor $\rho_r([x_p],[y_q]) = \rho_r([x_p]) * P([y_q]\,|\,[x_p])$;
while $t \leq t_{max}$ **do**
  ParFor *initialize* SSA propensities as $\rho_r([x_p])$;
  ParReduce *initialize* $\rho^{(total)} := \sum_r \rho_r([x_p])$;
  initialize $\tau := 0$;
  draw effective waiting time τ_{max} from $\exp(-\tau_{max})$;
  while $\tau < \tau_{max}$ **do**
    ParFor **solve** ODE system, plus an extra ODE updating τ;
      $\frac{d\tau}{dt} = \rho^{(total)}(t)$;
  draw reaction r from distribution $\rho_r([x_p])/\rho^{(total)}$;
  draw $[y_q]$ from $P([y_q]\,|\,[x_p])$ and *execute* reaction r;