Vivarium: an interface and engine for (multi-algorithm/multi-approach/multi-paradigm) simulations

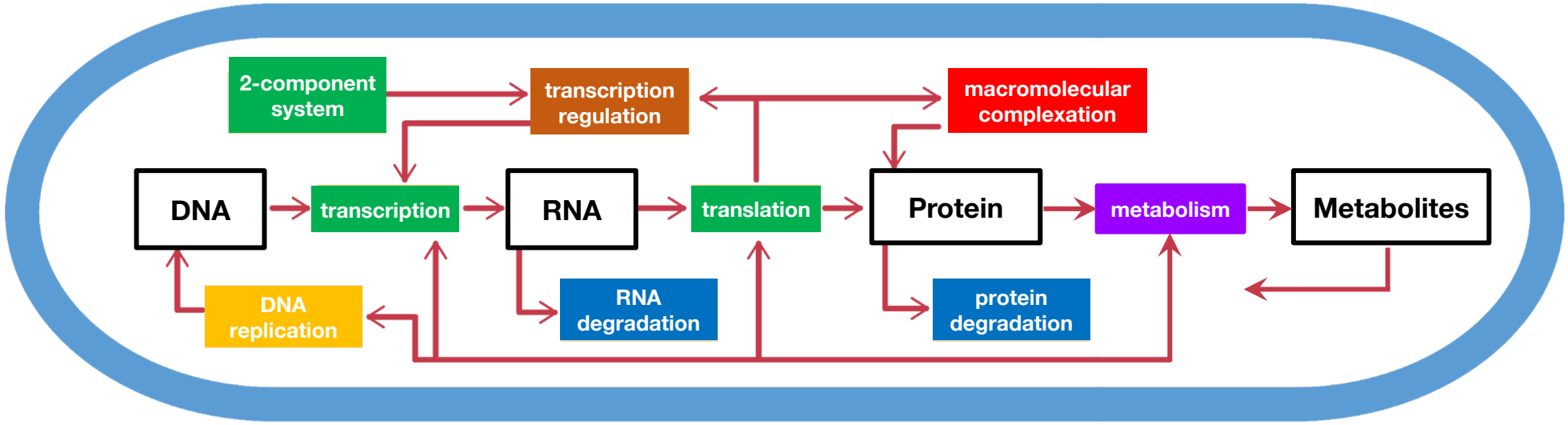
Eran Agmon, PhD

Department of Bioengineering | Stanford University

COMBINE 2021 | 10/11/2021

Background: Whole-cell modeling

the Covert Lab's *wcEcoli* model



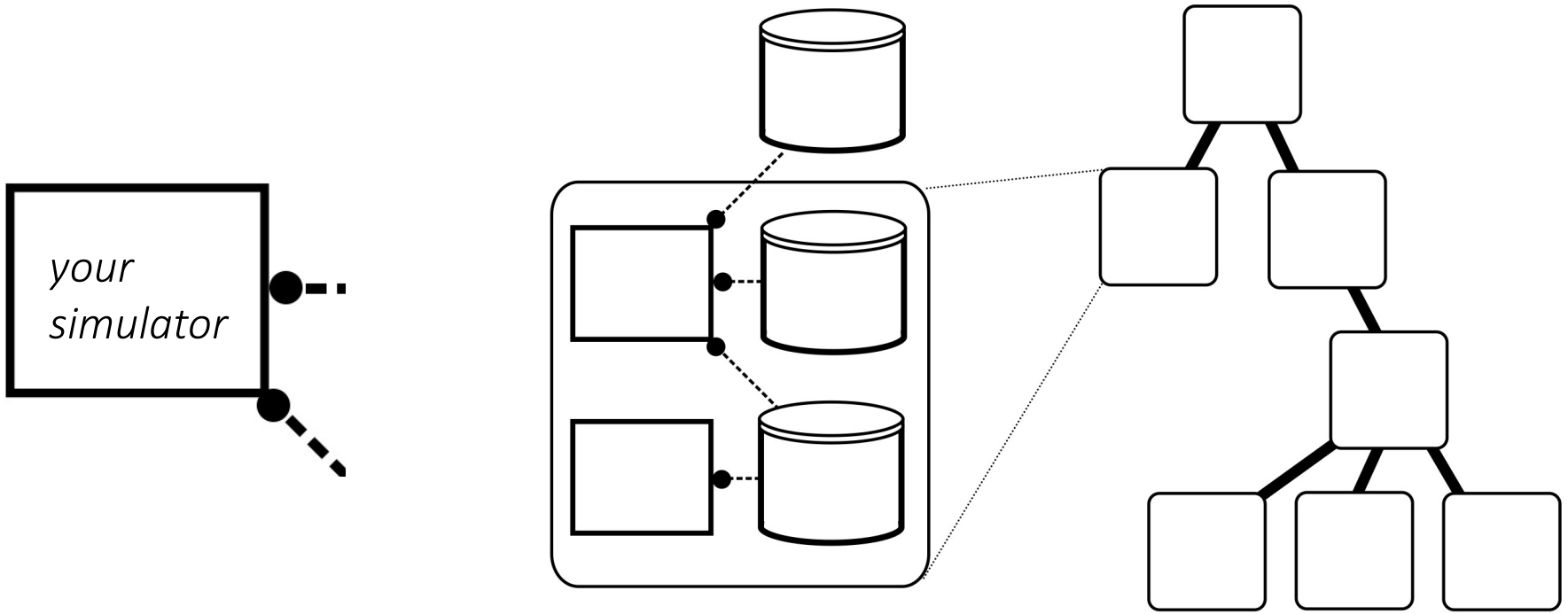
Ordinary Differential Equations | Stochastic Simulation | Poisson Process | Boolean | Agent-based model | Flux Balance Analysis

GitHub: https://github.com/CovertLab/WholeCellEcoliRelease

Macklin, D.N., et al. (2020). Simultaneous cross-evaluation of heterogeneous *E. coli* datasets via mechanistic simulation. *Science.*

Vivarium Overview

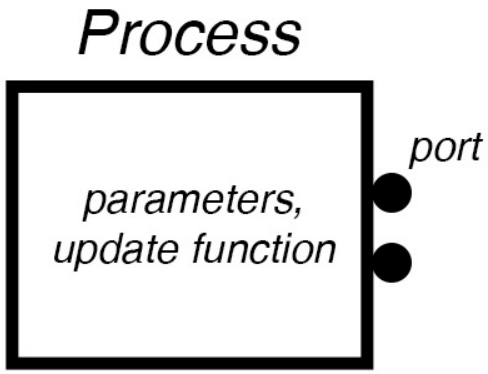
We need an "interface protocol" for connecting separate models, simulators, and data into a large, complex, and open-ended network that anyone can contribute to.

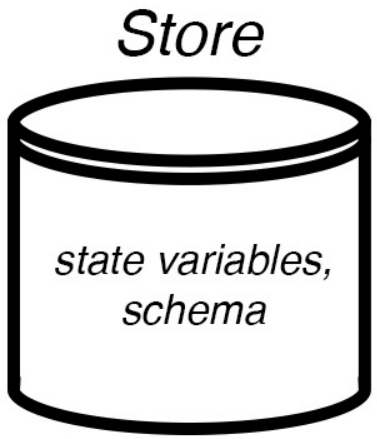


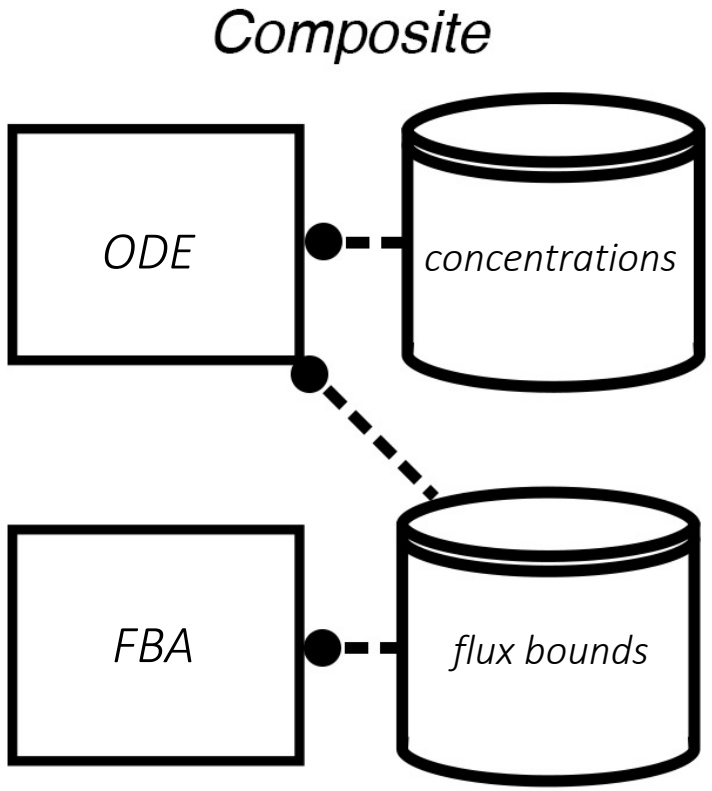
Agmon, E., et al. (2021). Vivarium: an interface and engine for integrative multi-scale modeling in computational biology. *bioRxiv.*

Basic Elements

- **Processes:** consist of parameters, ports, and an update function (i.e. the simulator).
- **Stores:** hold the state variables, map the variable names to their values, and apply the process updates.
- **Composites:** bundles of processes and stores, wired together by their ports, and run together in time.

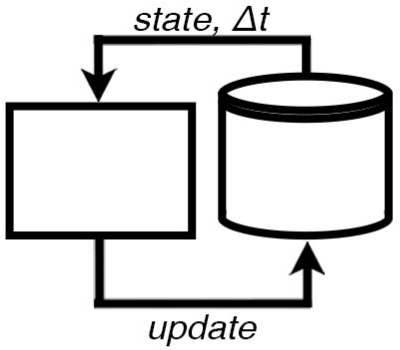




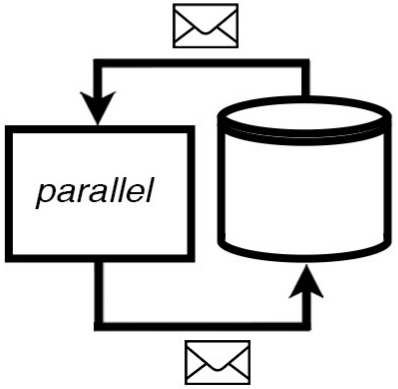


Co-Simulation Engine

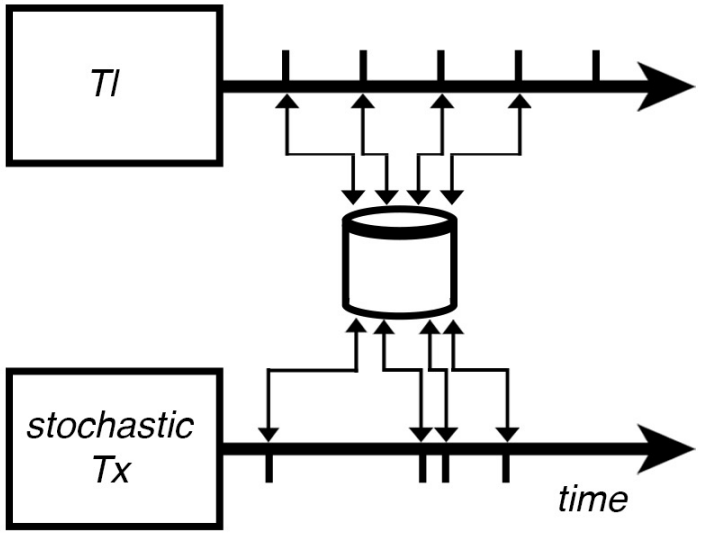
basic simulation cycle



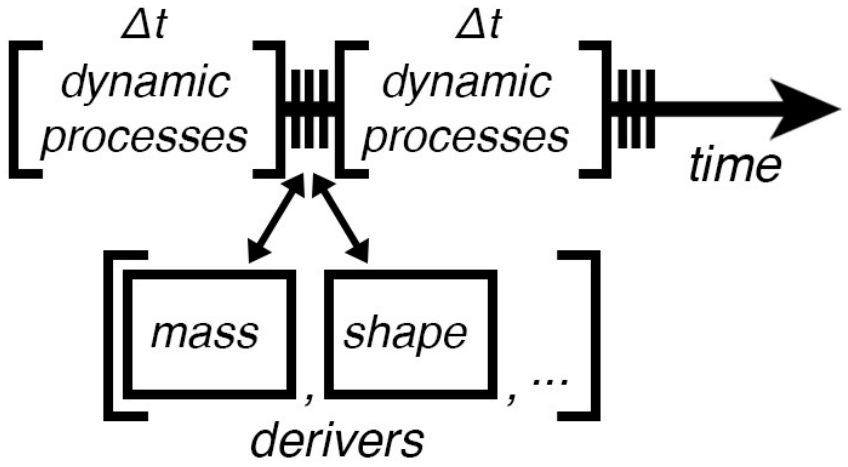
distributed processing



multi-timestepping

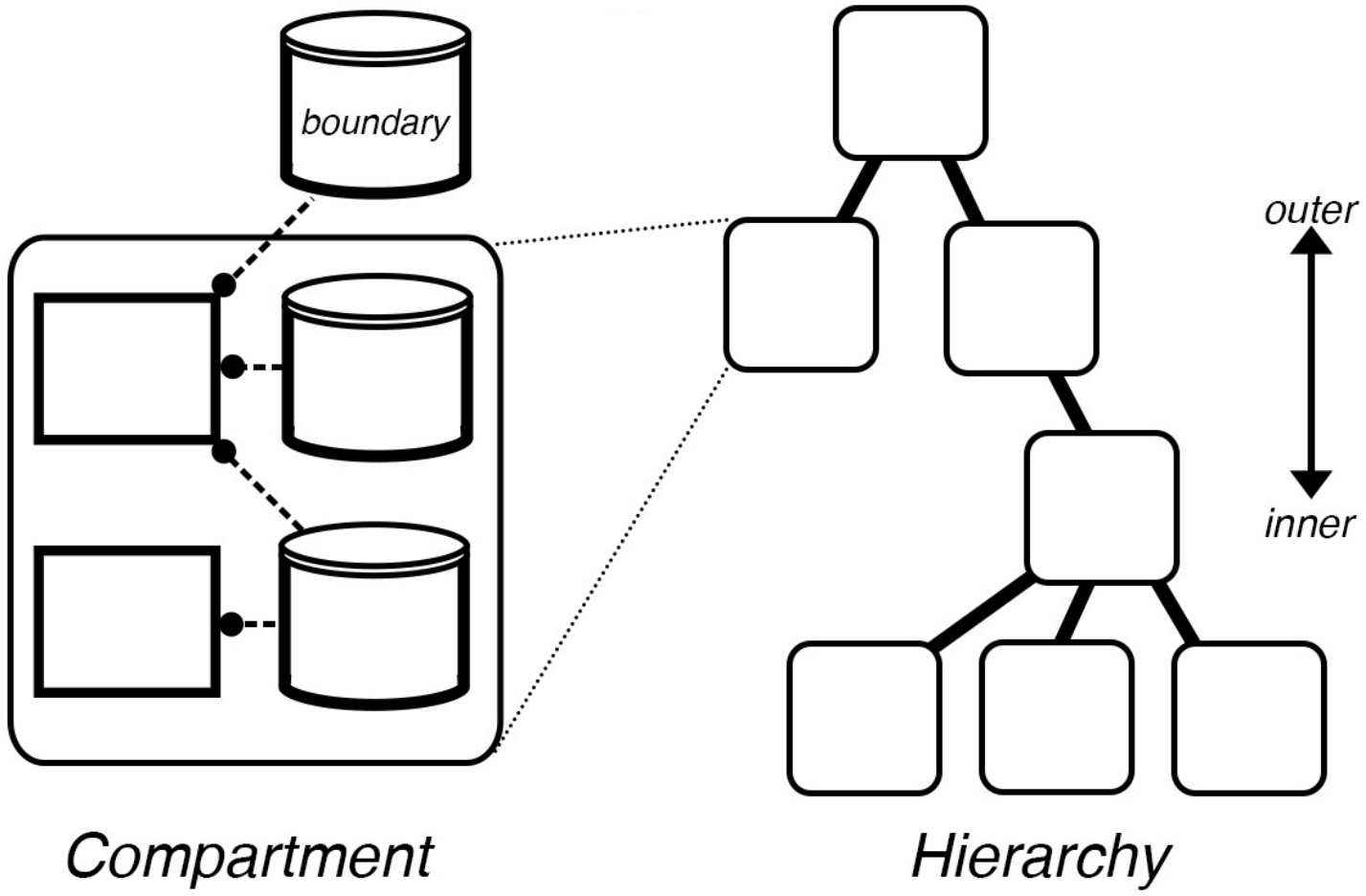


"derivers" run between
the time-dependent processes



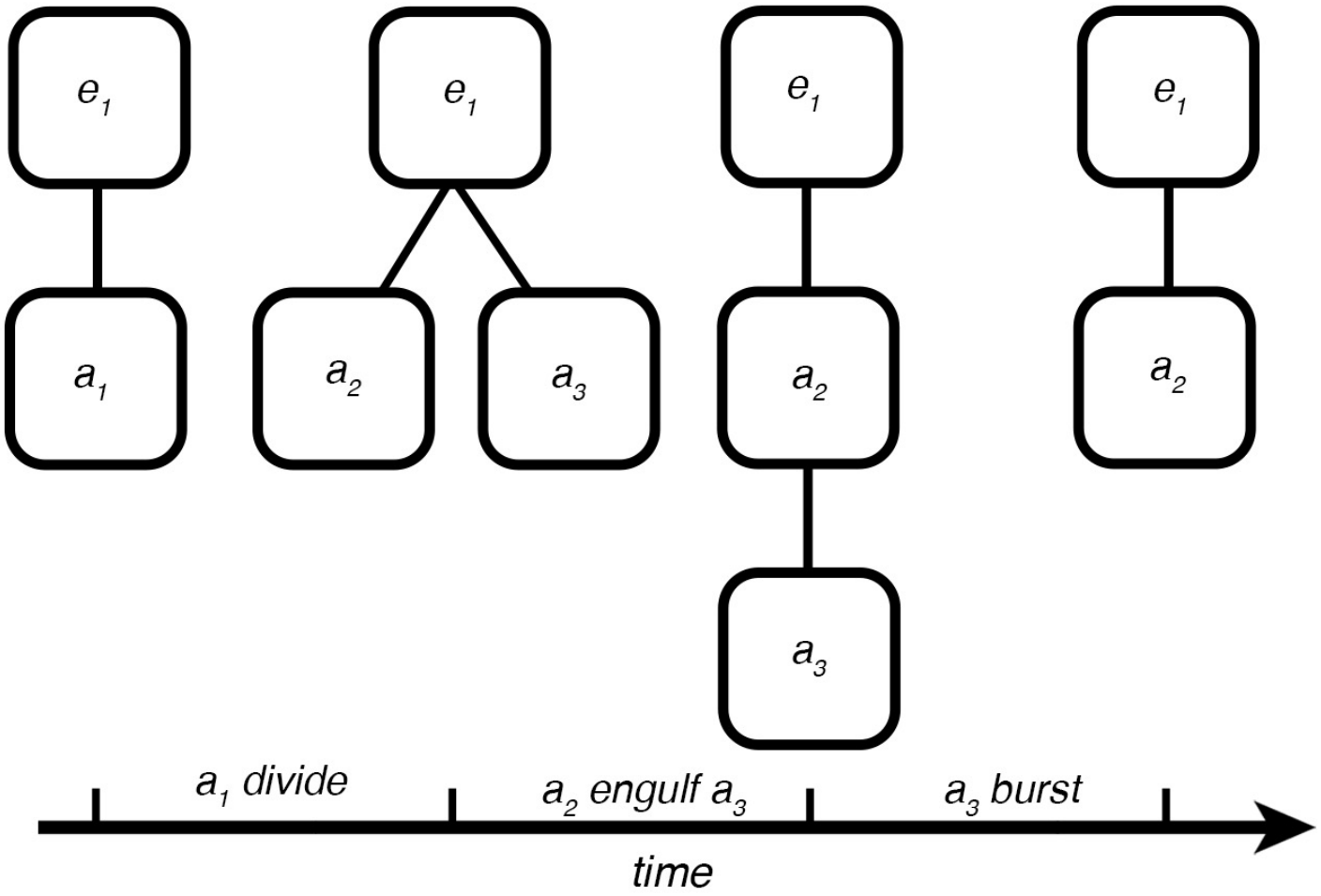
Hierarchical Embedding

- A bigraph is a graph with embeddable nodes that can be placed *within* other nodes.



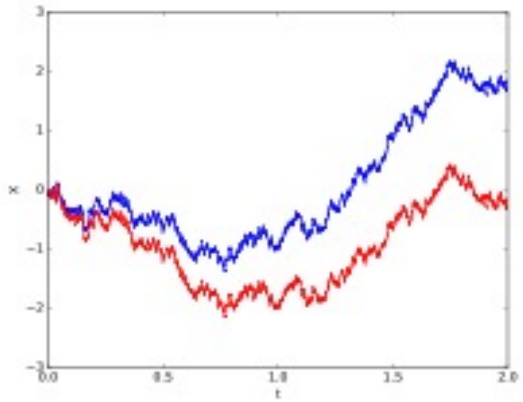
Hierarchical Updates

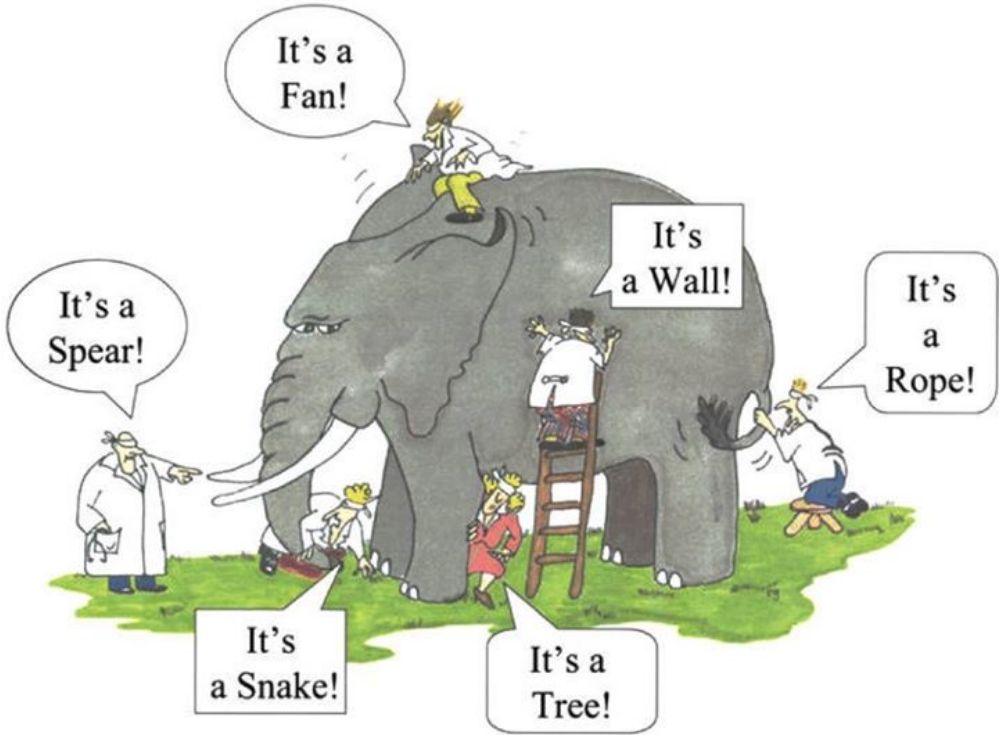
Processes, stores, and entire sub-graphs can be added/removed/moved during simulation runtime.

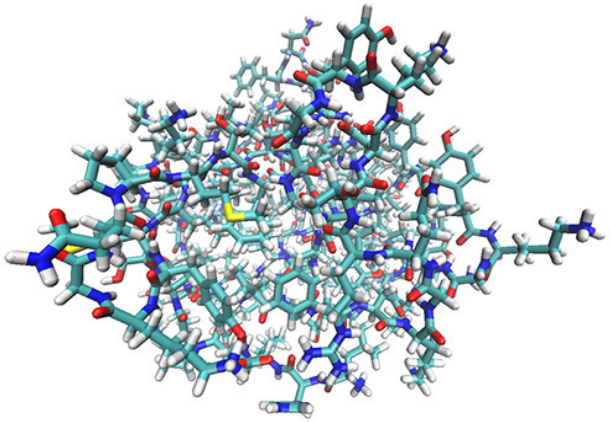


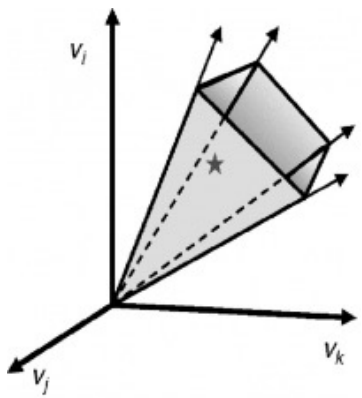
Multi-Paradigm Composites

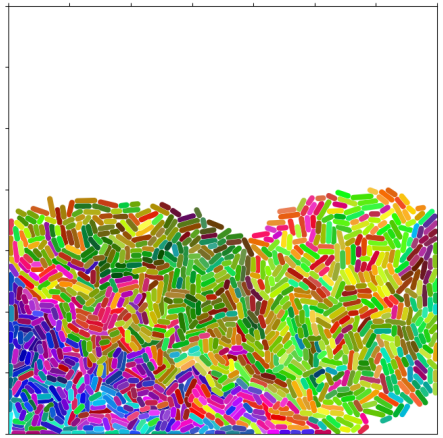
Biology is multi-scale and multi-modal. Our models must be as well.





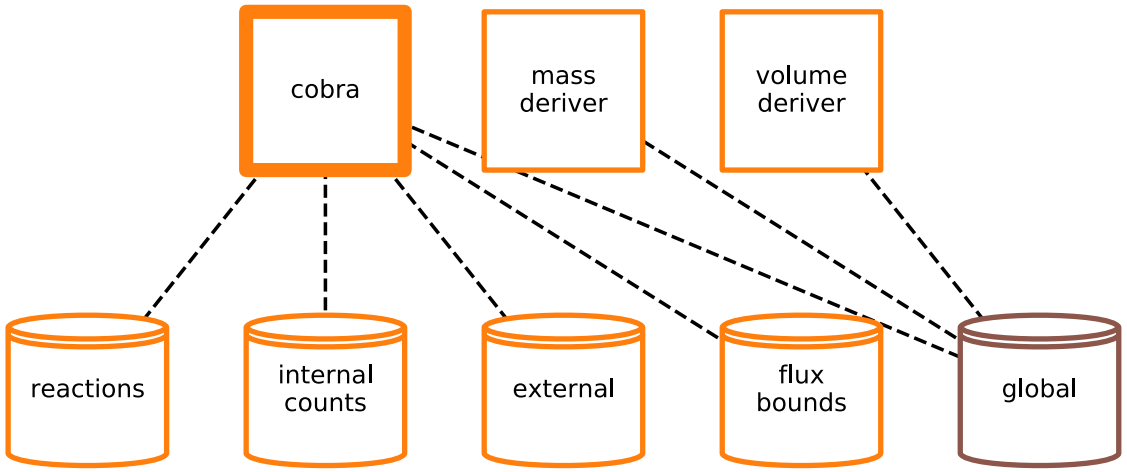


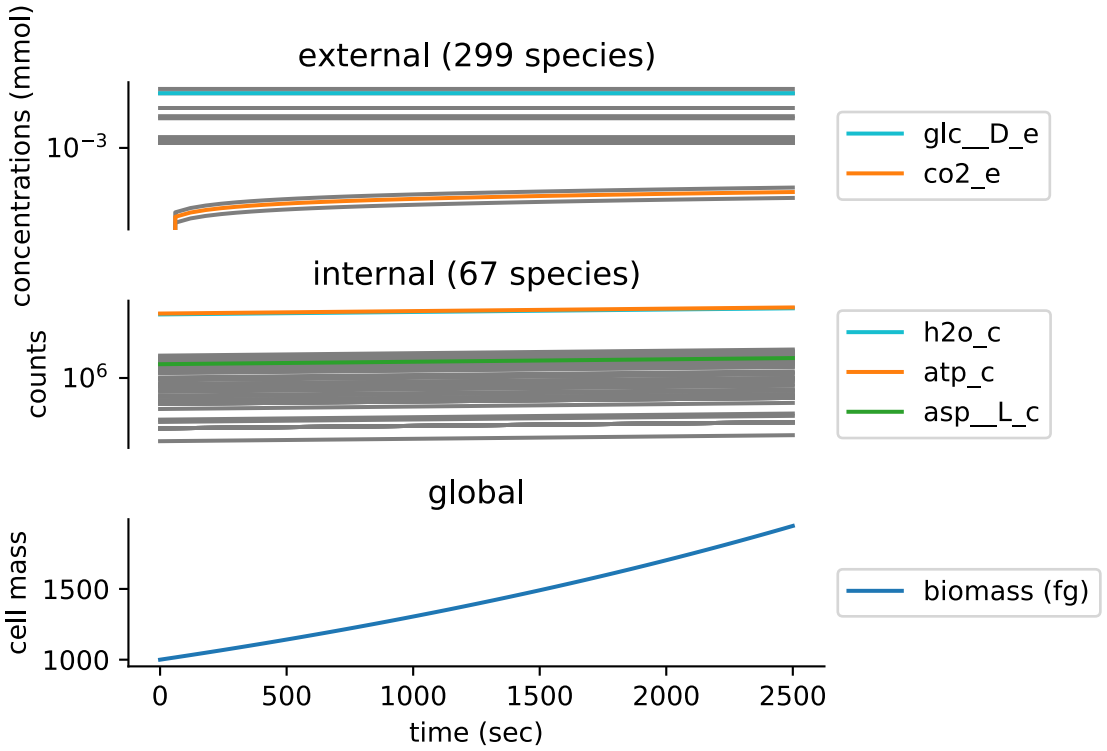




vivarium-cobra: dynamic flux-balance analysis

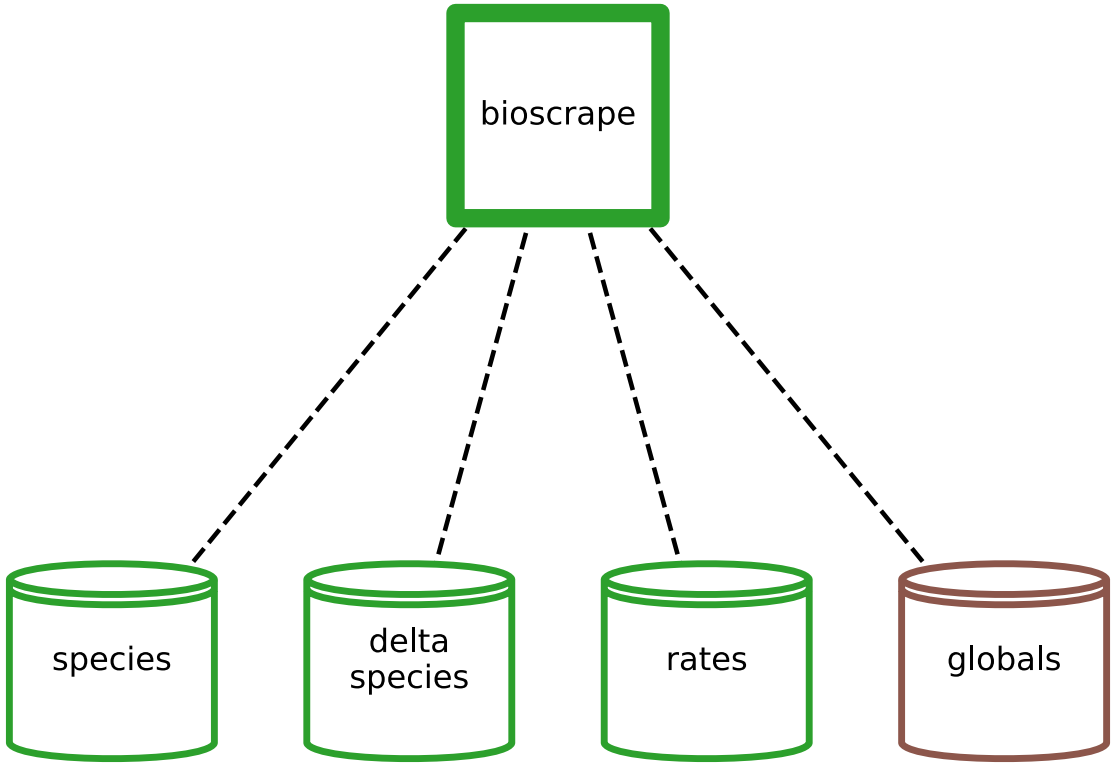
```
pip install vivarium-cobra
```

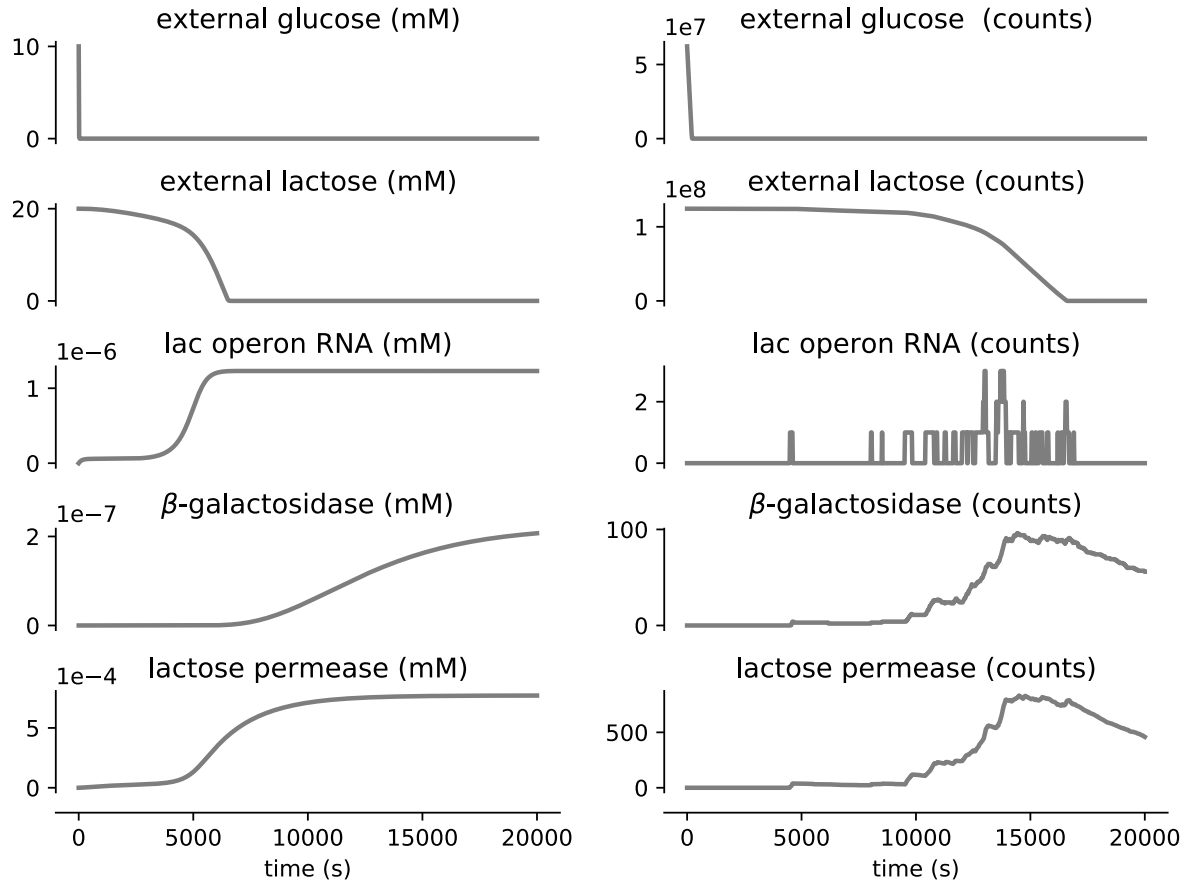




vivarium-bioscrape: chemical reaction networks

```
pip install vivarium-bioscrape
```



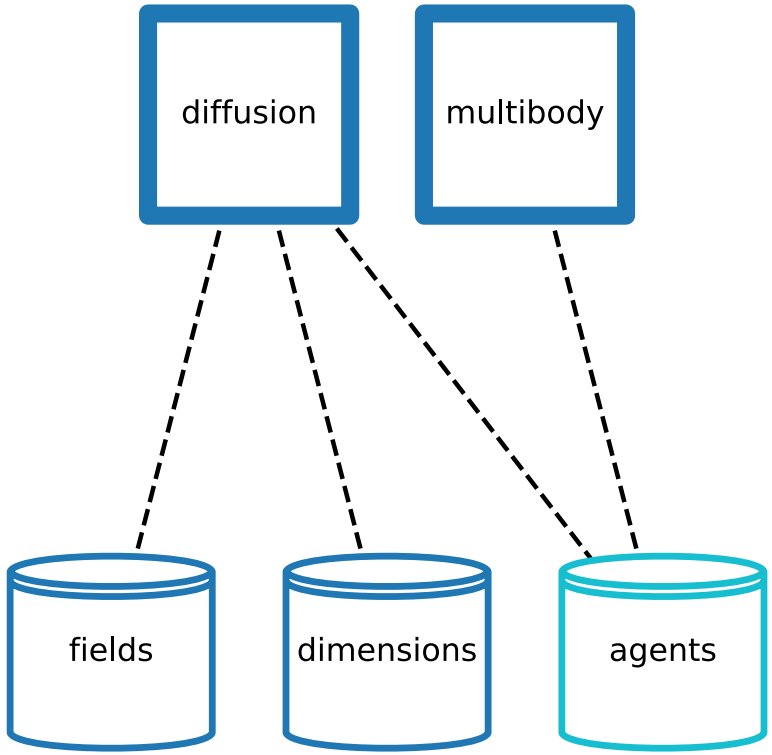


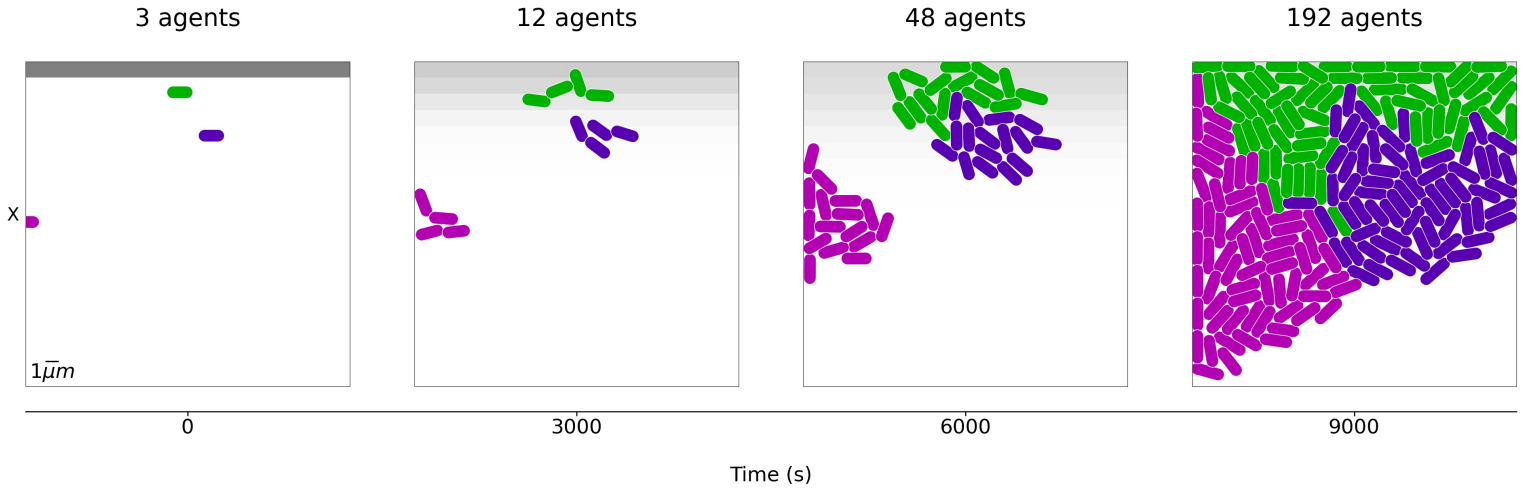
deterministic

stochastic

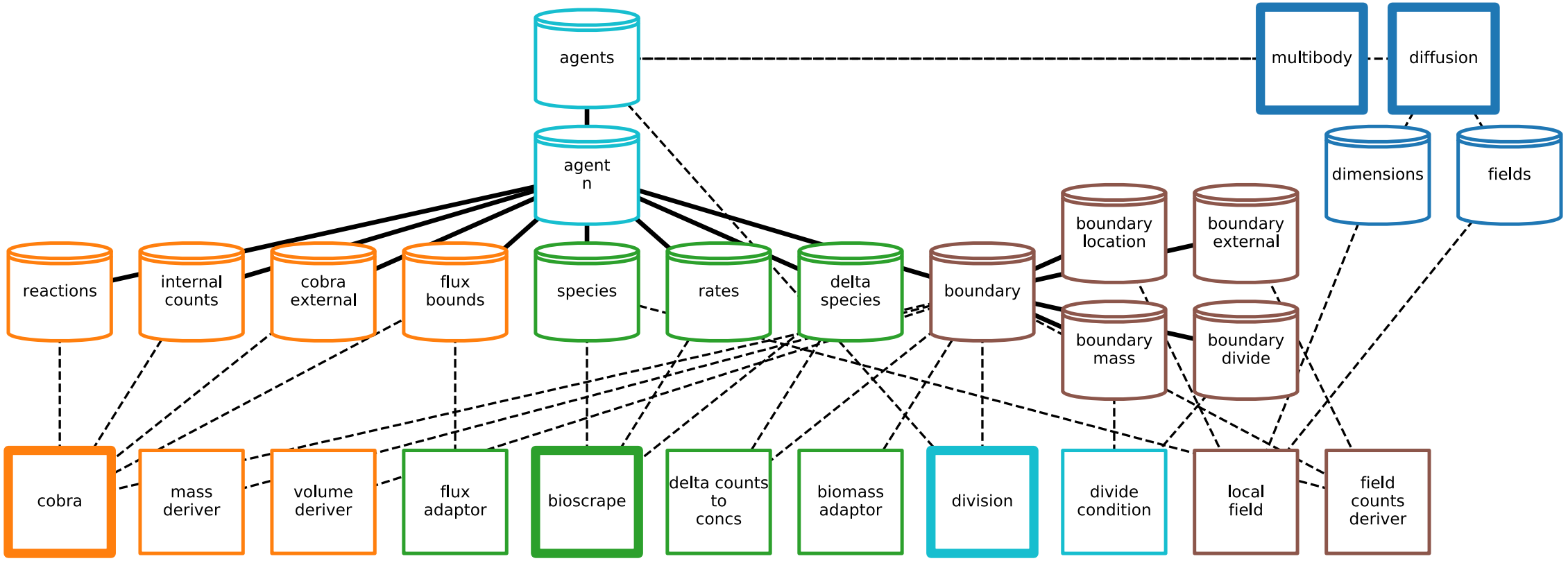
Vivarium-multibody: solid body physics + diffusion

```
pip install vivarium-multibody
```

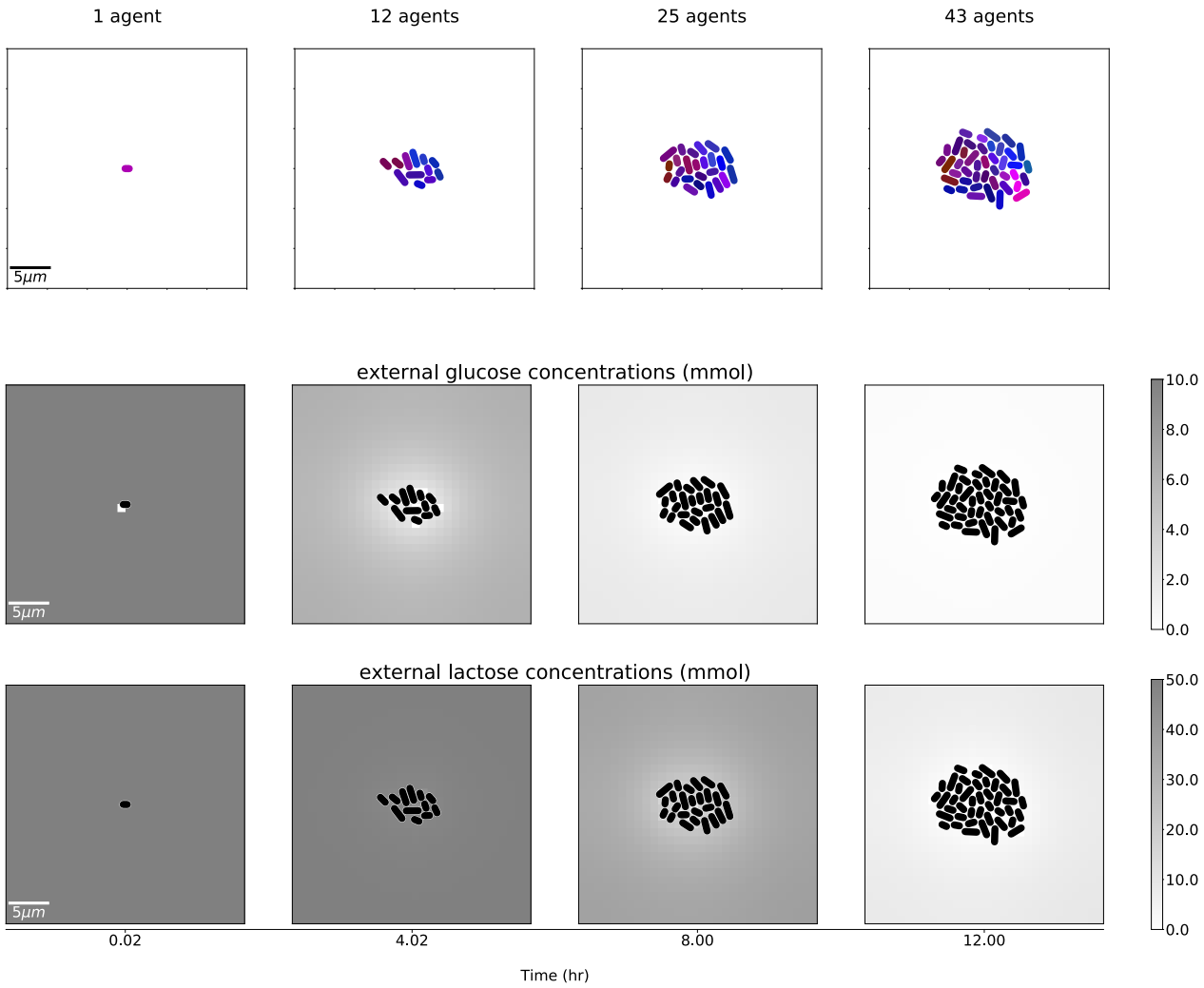


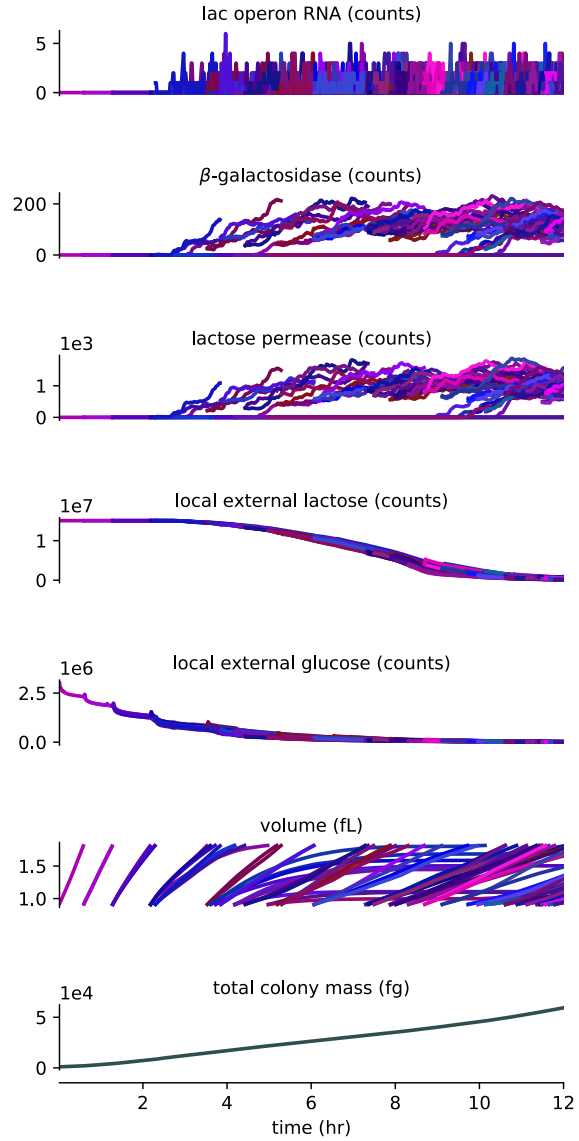


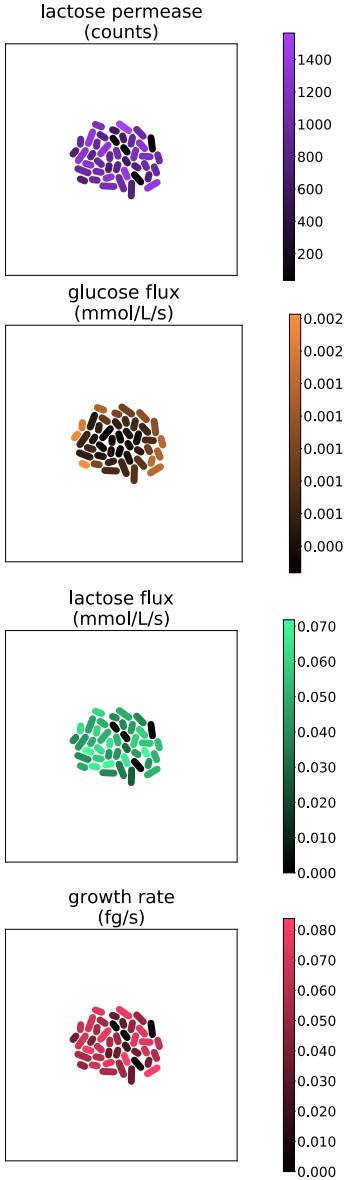
Results: putting it all together



Results: simulation output







Look at some code

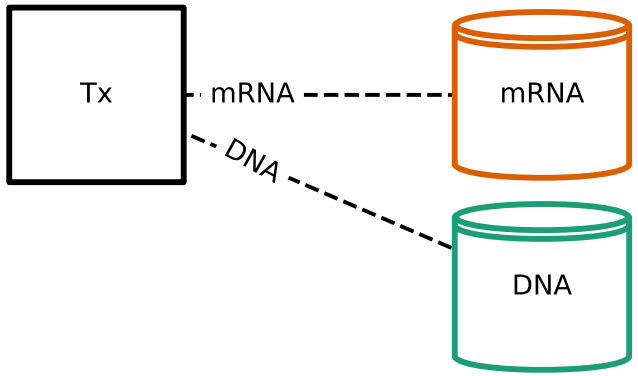
- Demonstrate the Vivarium interface by working through a minimal example system of unregulated gene expression.
- This demo is up on the Vivarium docs: https://vivarium-core.readthedocs.io/en/latest/notebooks/Vivarium_interface_basics.html

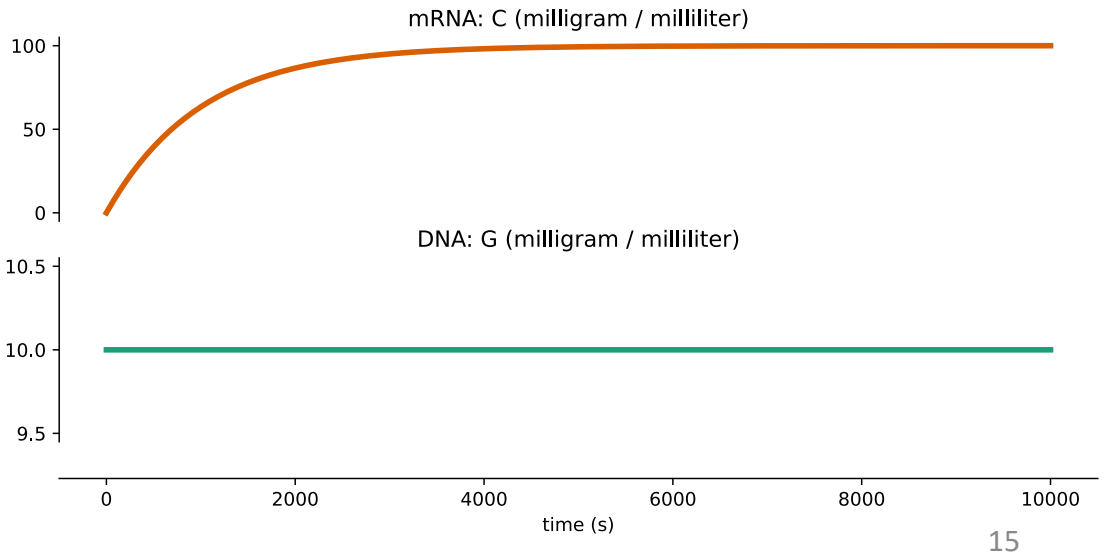
Minimal Example: Transcription ODE

Process Interface Protocol:

1. **constructor***:* accepts parameters and configures the model.
2. **ports_schema***:* declares the ports and their schema.
3. **next_update***:* runs the model and returns an update.

```python
class Tx(Process):

    defaults = {
        'ktsc': 1e-2,
        'kdeg': 1e-3}

    def __init__(self, parameters=None):
        super().__init__(parameters)

    def ports_schema(self):
        return {
            'DNA': {
                'G': {
                    '_default': 10 * units.mg / units.mL,
                    '_updater': 'accumulate',
                    '_emit': True}},
            'mRNA': {
                'C': {
                    '_default': 100 * units.mg / units.mL,
                    '_updater': 'accumulate',
                    '_emit': True}}}

    def next_update(self, timestep, states):
        G = states['DNA']['G']
        C = states['mRNA']['C']
        dC = (self.parameters['ktsc'] * G - self.parameters['kdeg'] * C) * timestep
        return {
            'mRNA': {
                'C': dC}}
```



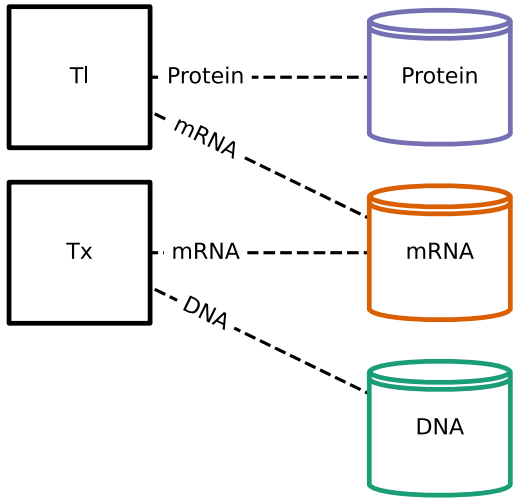


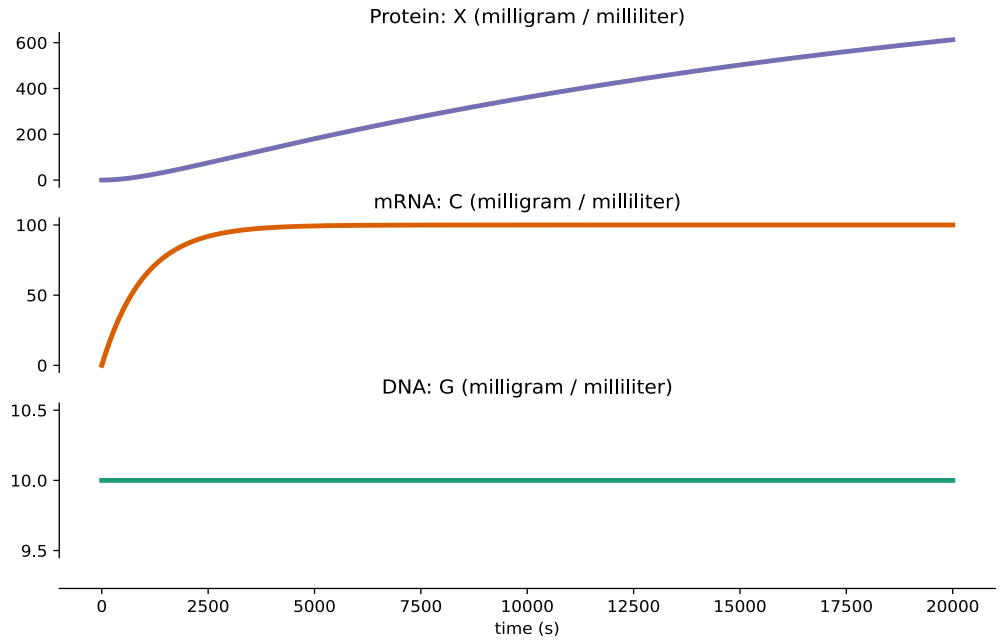
Minimal Composite: Transcription + Translation

Composition Protocol:

1. **generate_processes***:* initialize processes in a dictionary.
2. **generate_topology***:* declare how process ports are wired together.

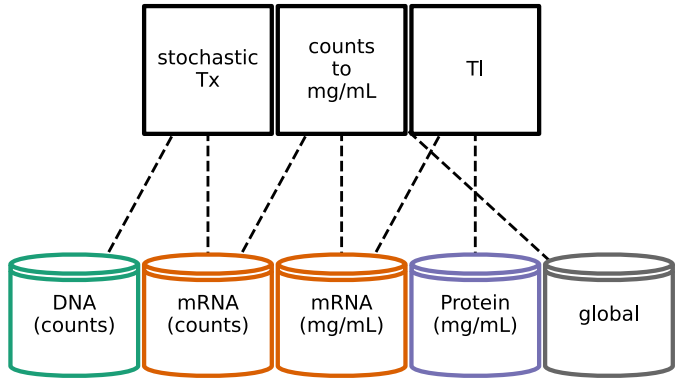
```python
class TxTl(Composer):

    defaults = {
        'Tx': {'time_step': 10},
        'Tl': {'time_step': 10}}

    def generate_processes(self, config):
        return {
            'Tx': Tx(config['Tx']),
            'Tl': Tl(config['Tl'])}

    def generate_topology(self, config):
        return {
            'Tx': {
                'DNA': ('DNA',),
                'mRNA': ('mRNA',)},
            'Tl': {
                'mRNA': ('mRNA',),
                'Protein': ('Protein',)}}
```

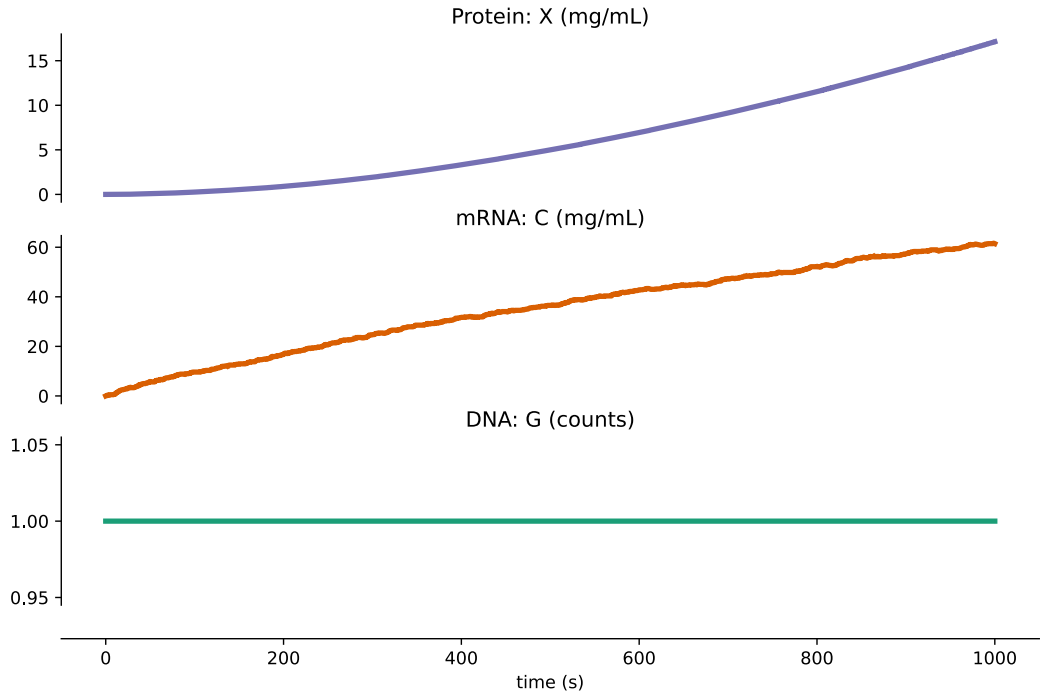




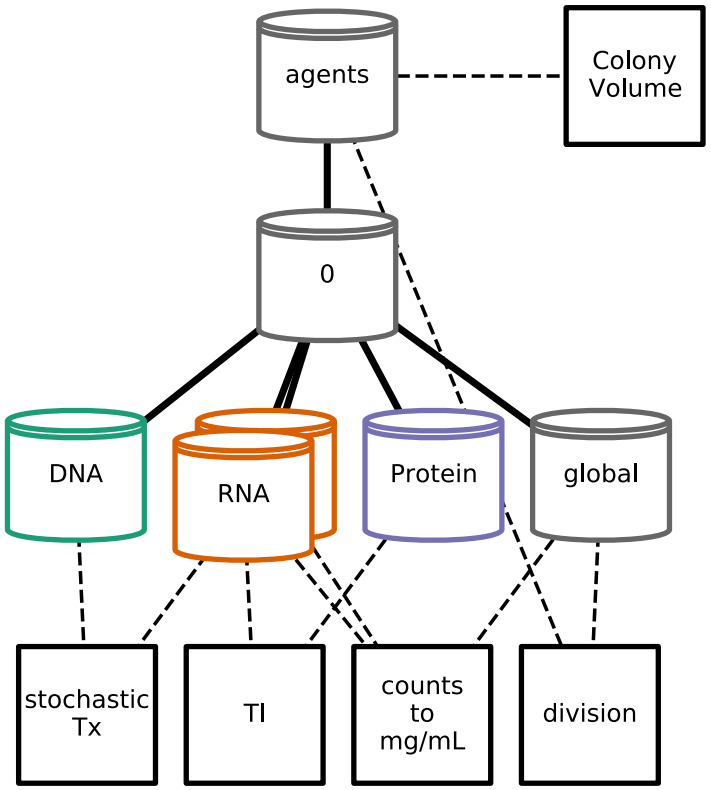
Swap out processes: Stochastic Transcription

```
class StochasticTxTl(Composer):
    defaults = {
        'stochastic_Tx': {},
        'Tl': {'time_step': 1},
        'concs': {
            'molecular_weights': mw_config}}

    def generate_processes(self, config):
        counts_to_concentration = process_registry.access('counts_to_concentration')
        return {
            'stochastic\nTx': StochasticTx(config['stochastic_Tx']),
            'Tl': Tl(config['Tl']),
            'counts\nto\nmg/mL': counts_to_concentration(config['concs'])}

    def generate_topology(self, config):
        return {
            'stochastic\nTx': {
                'DNA': ('DNA\n(counts)',),
                'mRNA': ('mRNA\n(counts)',)
            },
            'Tl': {
                'mRNA': ('mRNA\n(mg/mL)',),
                'Protein': ('Protein\n(mg/mL)',)
            },
            'counts\nto\nmg/mL': {
                'global': ('global',),
                'input': ('mRNA\n(counts)',),
                'output': ('mRNA\n(mg/mL)',)
            }}
```

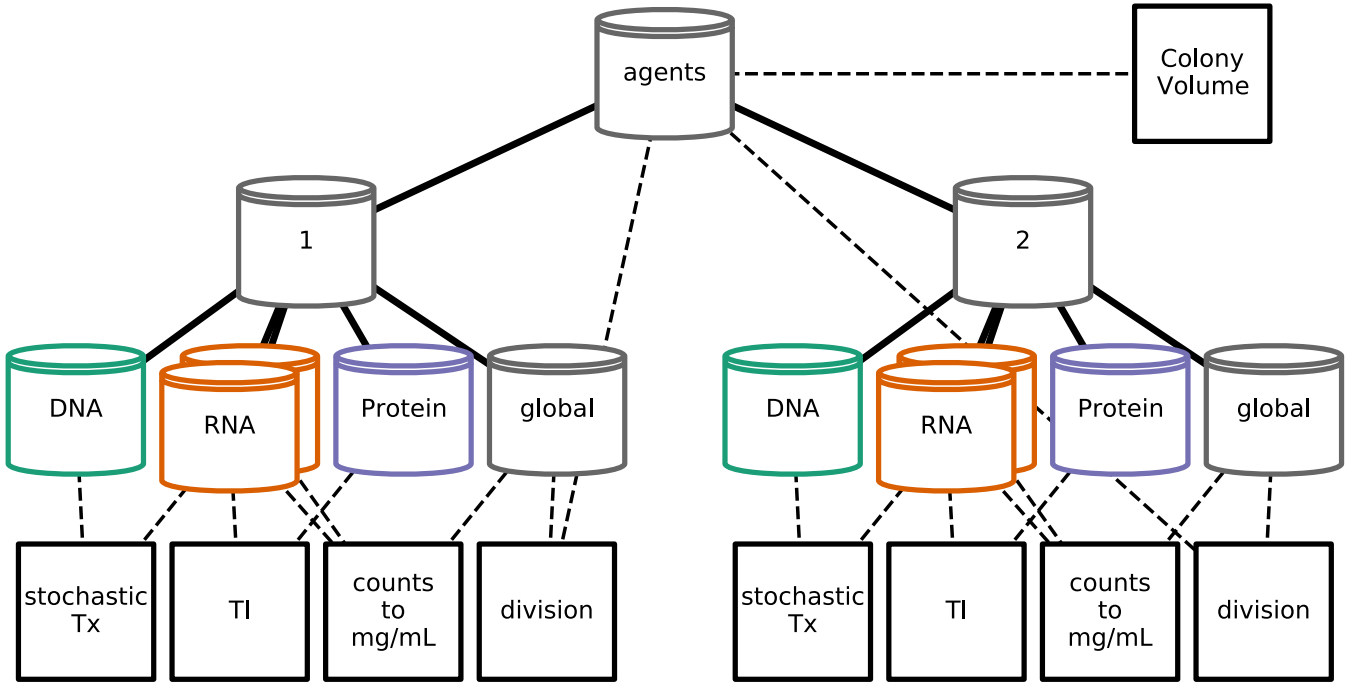




Hierarchical Updates

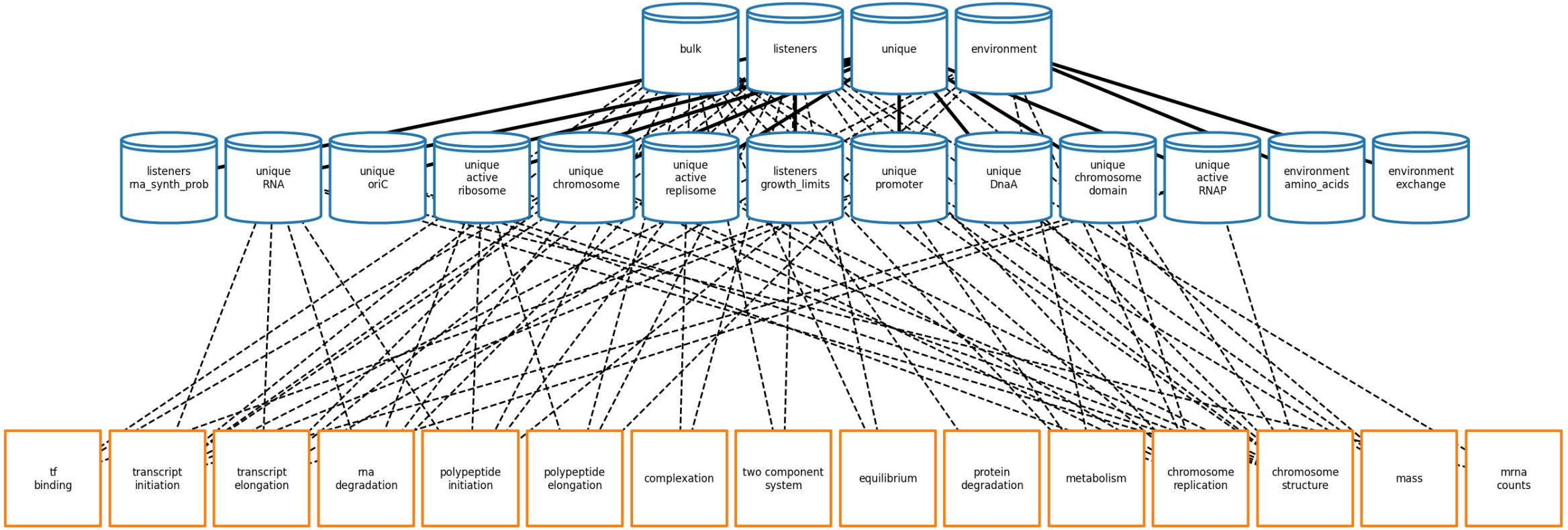


before division

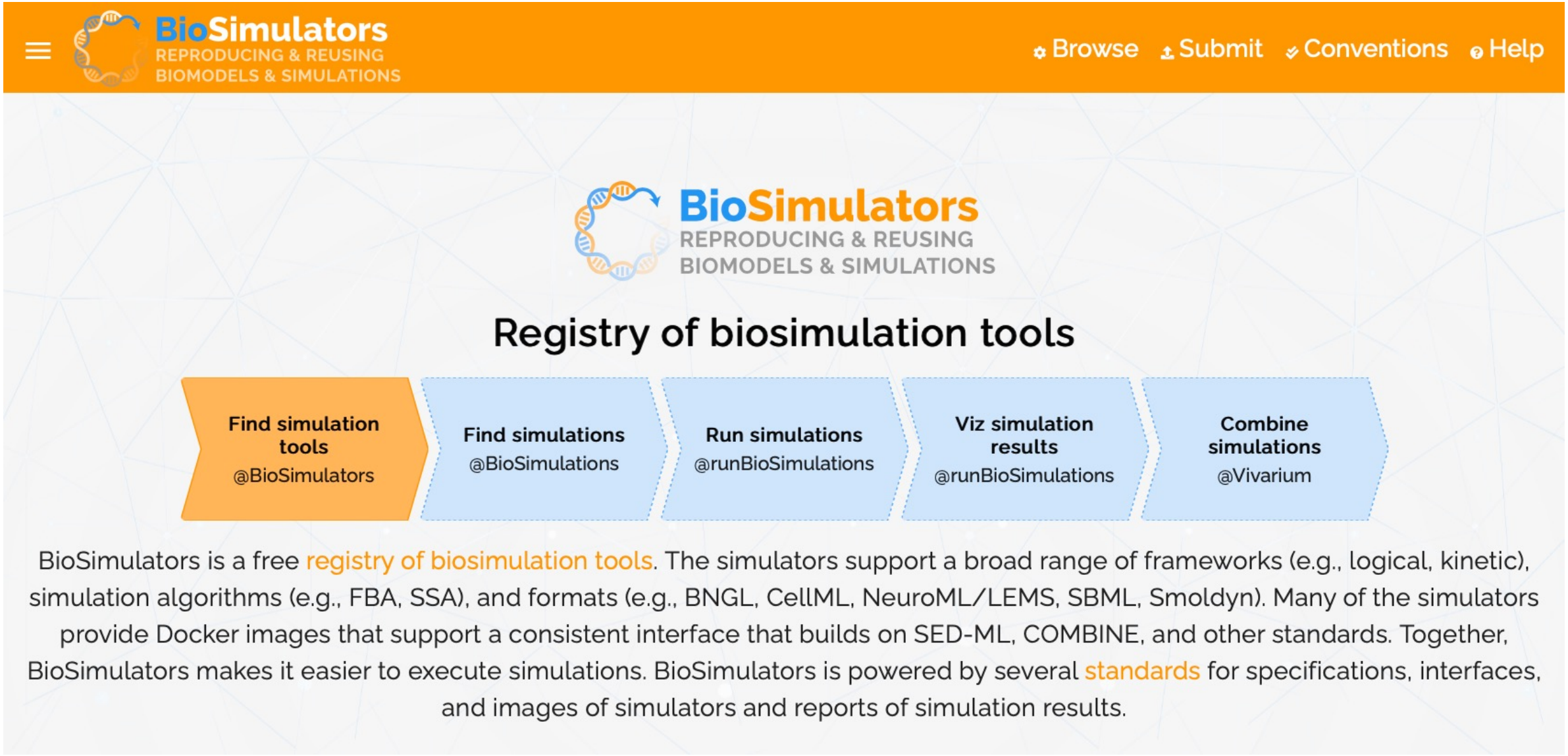


after division

Vivarium-Ecoli



BioSimulators/Vivarium Workshop



BioSimulators/Vivarium Workshop

“Standardized simulation execution with Biosimulators and co-simulation with Vivarium”

- **Wednesday**, October 13 • 14:00 - 16:00 • Overview and demos
- **Thursday**, October 14 • 14:00 - 17:00 • Hack with developers
- **Friday**, October 15 • 16:00 - 17:00 • Community input and feedback

Thank you!

Vivarium-core:

- Ryan Spangler (Allen Institute for Cell Science)
- Chris Skalnik (Stanford)
- William Poole (Caltech)
- Jerry Morrison (Stanford)
- Shayn Peirce-Cottler (U of Virginia)
- Markus Covert (Stanford)

References:

- **Vivarium-Collective:** https://vivarium-collective.github.io
- **Vivarium Documentation:** https://vivarium-core.readthedocs.io
- **Demo:** https://vivarium-core.readthedocs.io/en/latest/notebooks/Vivarium_interface_basics.html
- **bioRxiv:** Agmon, E., Spangler, R. K., Skalnik, C. J., Poole, W., Peirce, S. M., Morrison, J. H., & Covert, M. W. (2021). Vivarium: an interface and engine for integrative multiscale modeling in computational biology.

email: eagmon@stanford.edu