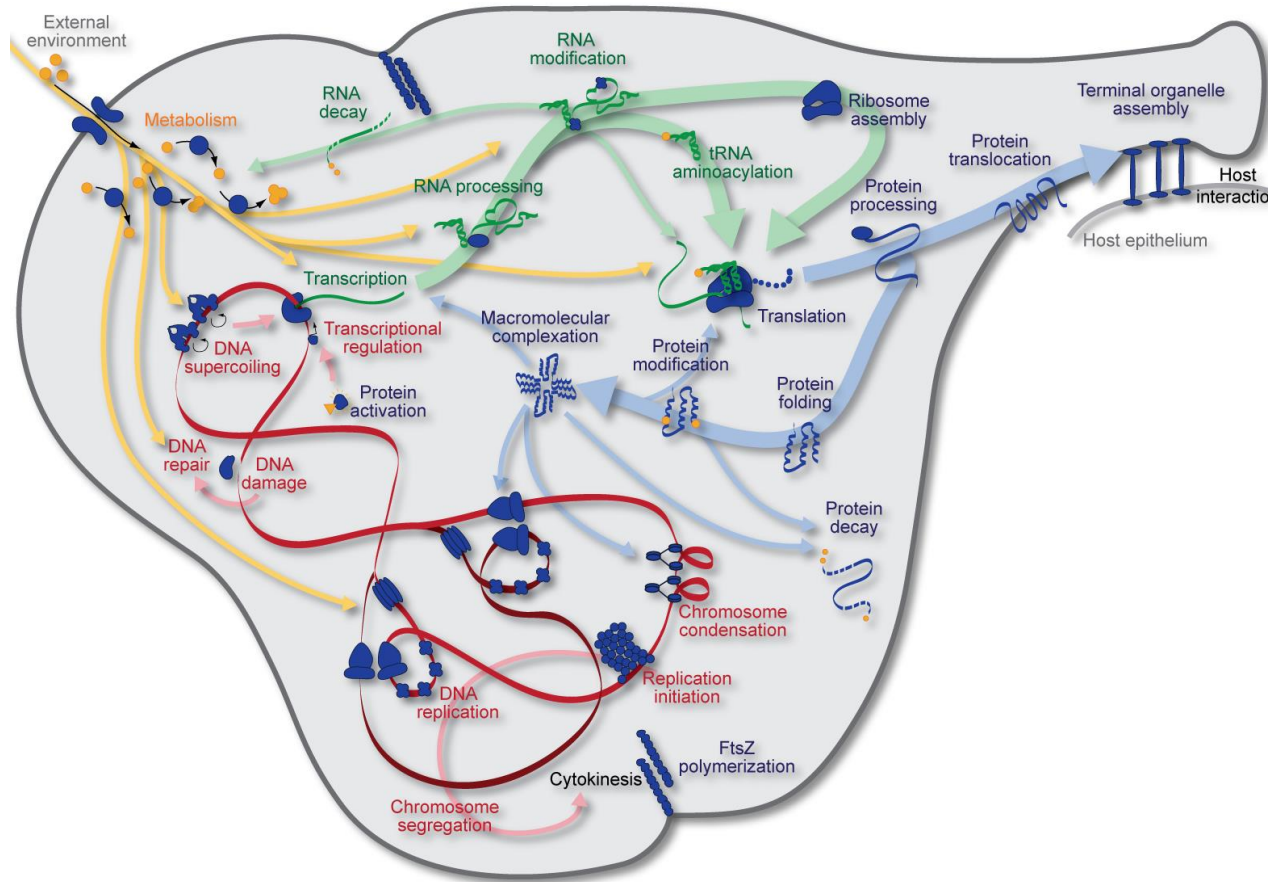


# Whole-cell modeling



# Outline

## **Introduction**

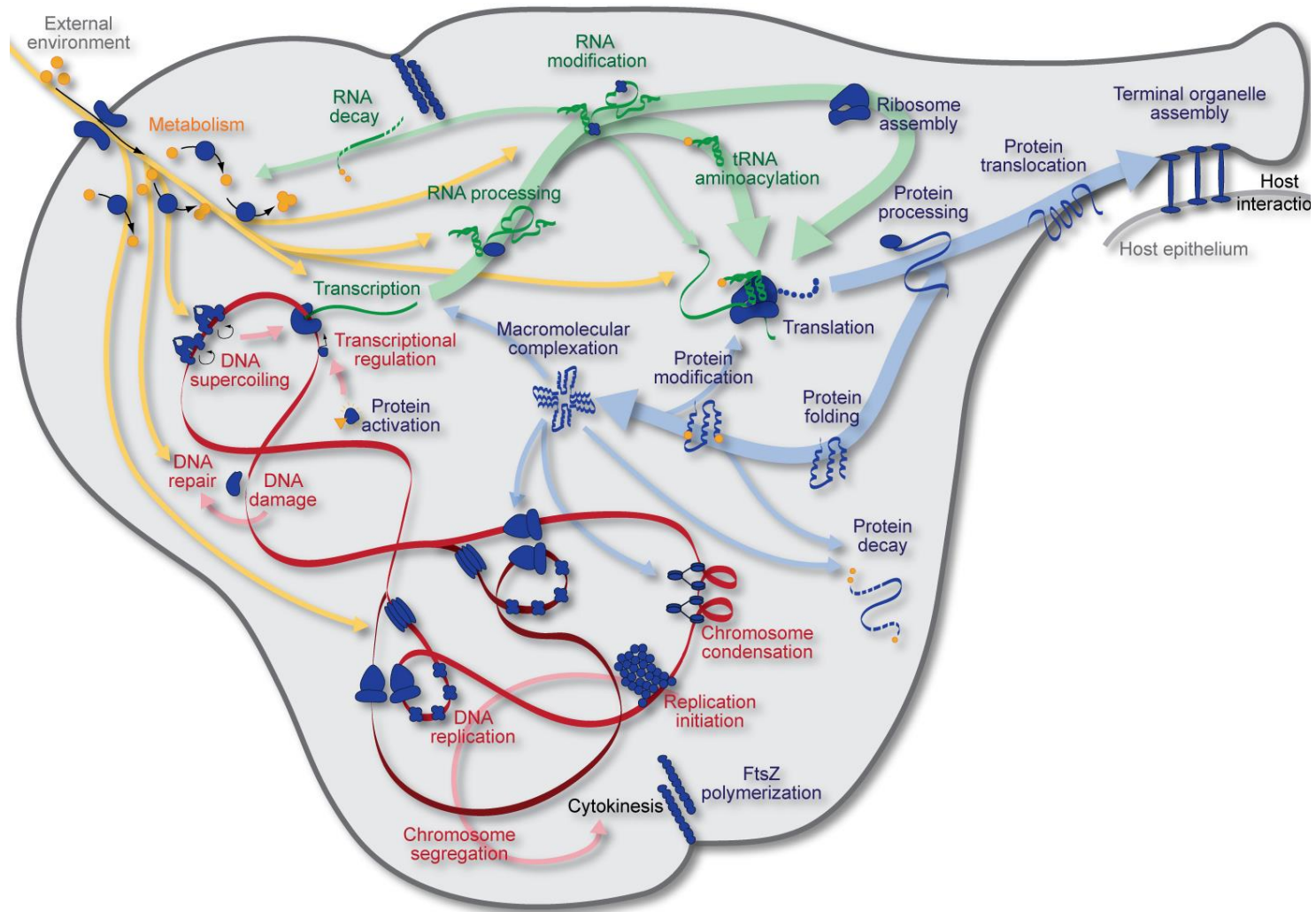
- Goals
- Approaches
- Multi-algorithm simulation
- Parameter estimation, verification, best practices

## **Exercises**

- Model building
- Parameter estimation
- Multi-algorithm simulation
- Submodel simulation
- Model annotation

# Introduction

# Motivation: Comprehensively understand and manipulate cells



# Goals


- Represent multiple pathways with different structures and dynamics
- Represent well- and poorly-characterized pathways
- Integrate heterogeneous data
- Train models from incomplete and noisy data
- Integrate molecular information over many orders of length and time

# Approaches

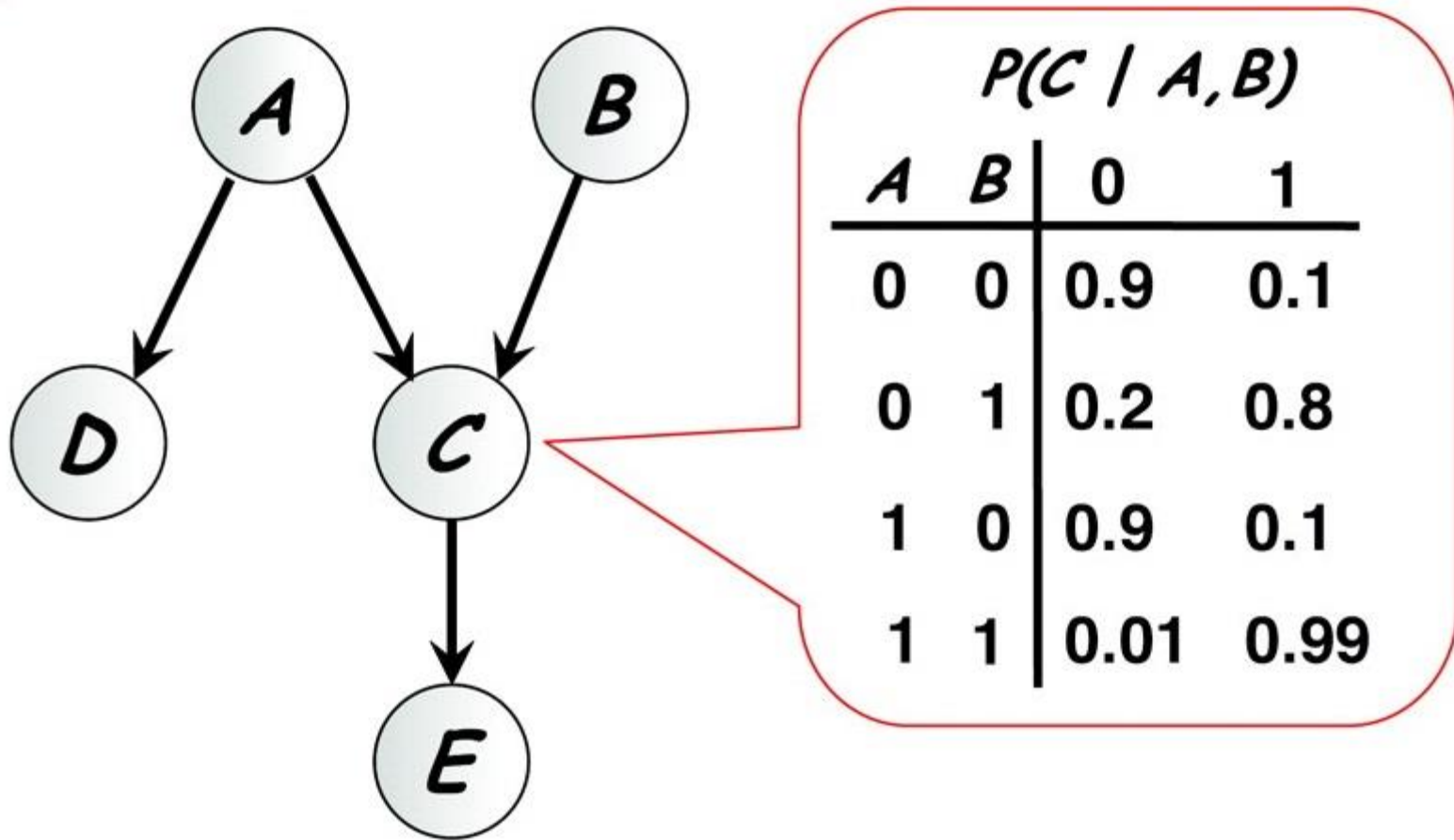
## Data-based

- Based on observed phenotypes
- Simpler mathematical formulation
- Easier to construct
- Limited ability to extrapolate beyond training data

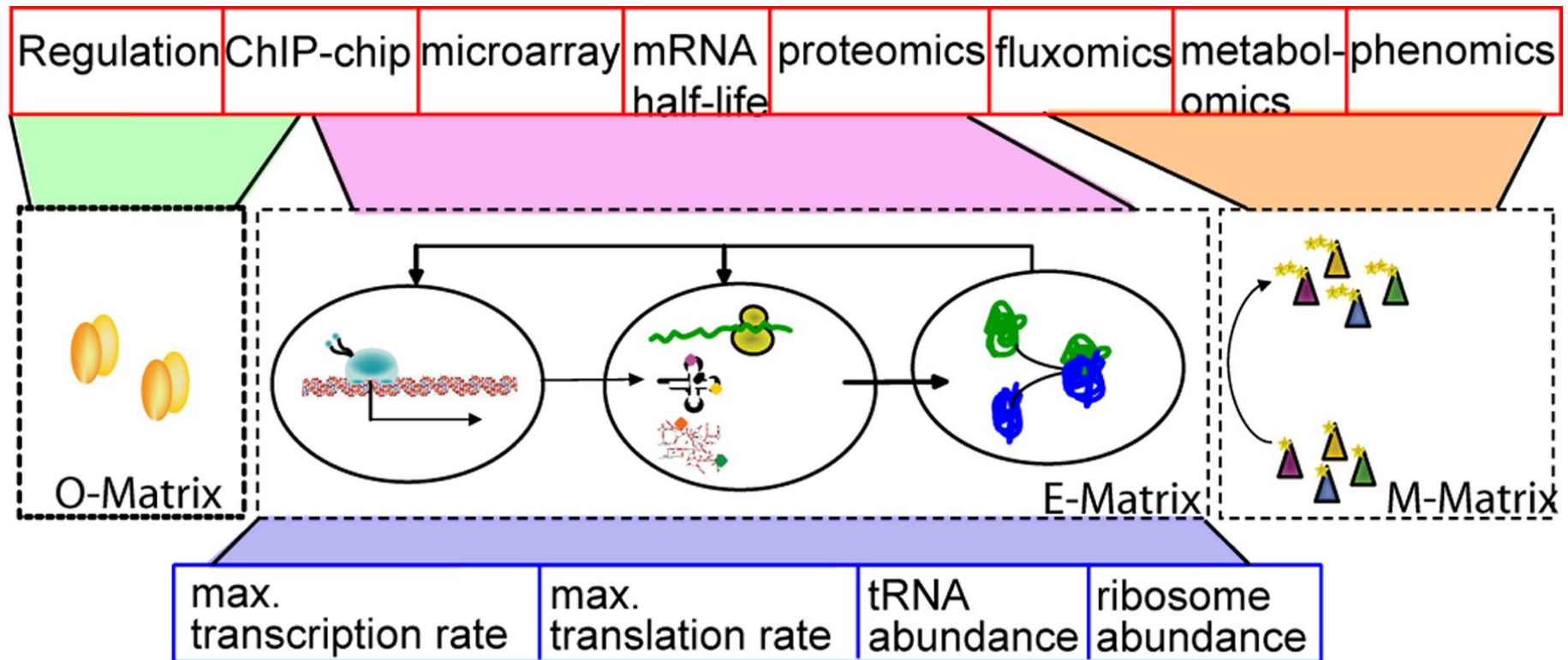
## Physics-based

- Based on known biochemistry and biophysics
- Complex, non-linear mathematics
- Time-consuming to construct
- Uses universal physical  to extrapolate beyond training data

# Data-based approaches: Bayesian models

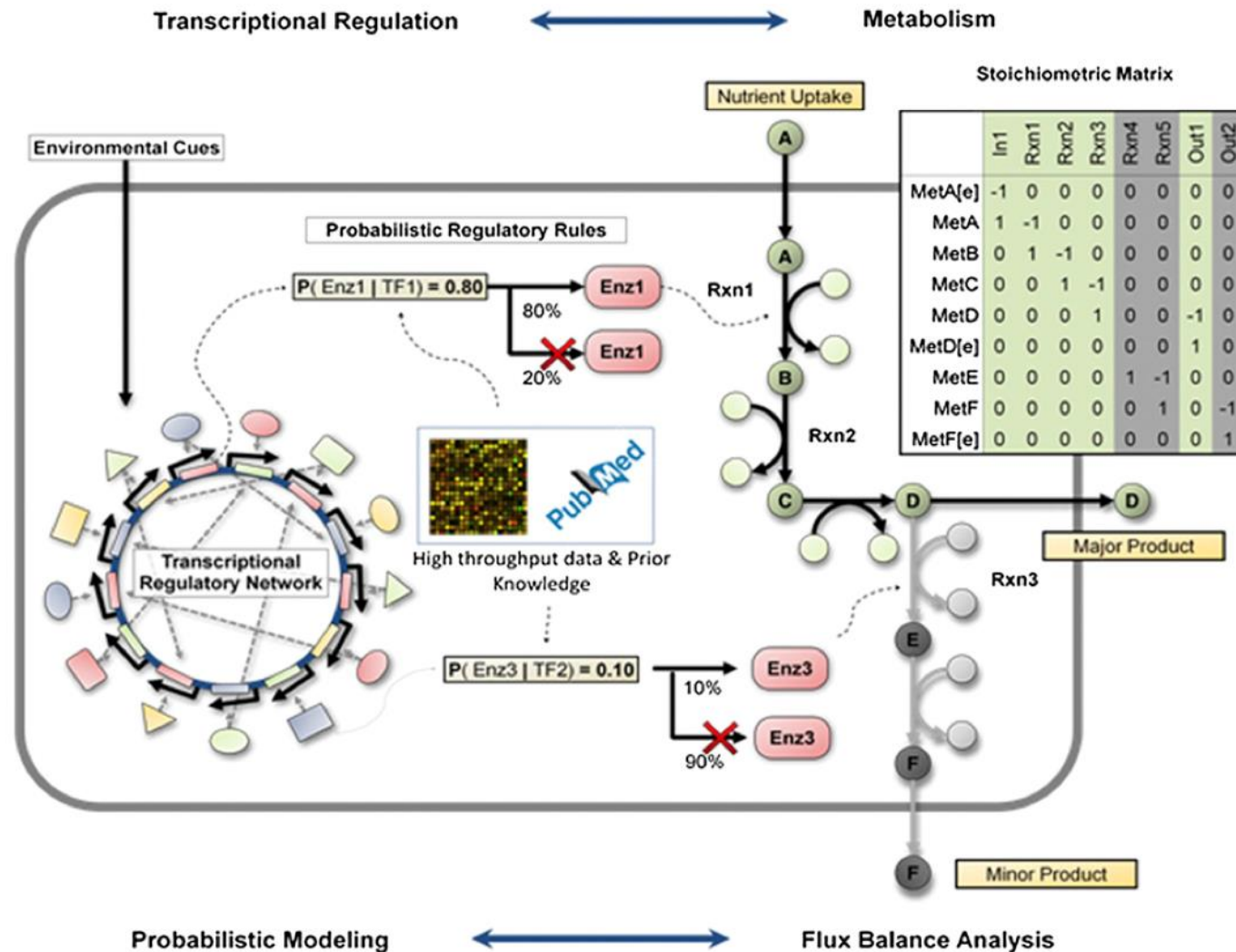


# Phenomenological approaches: FBA

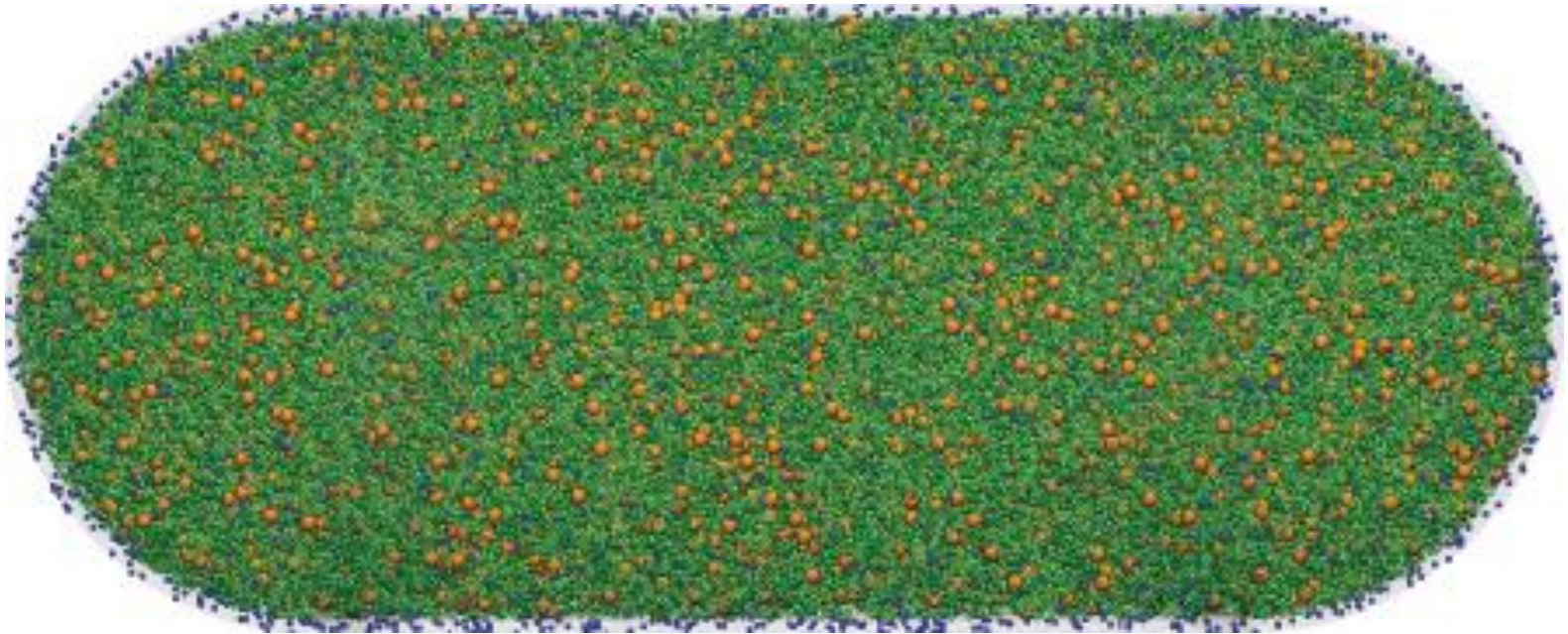




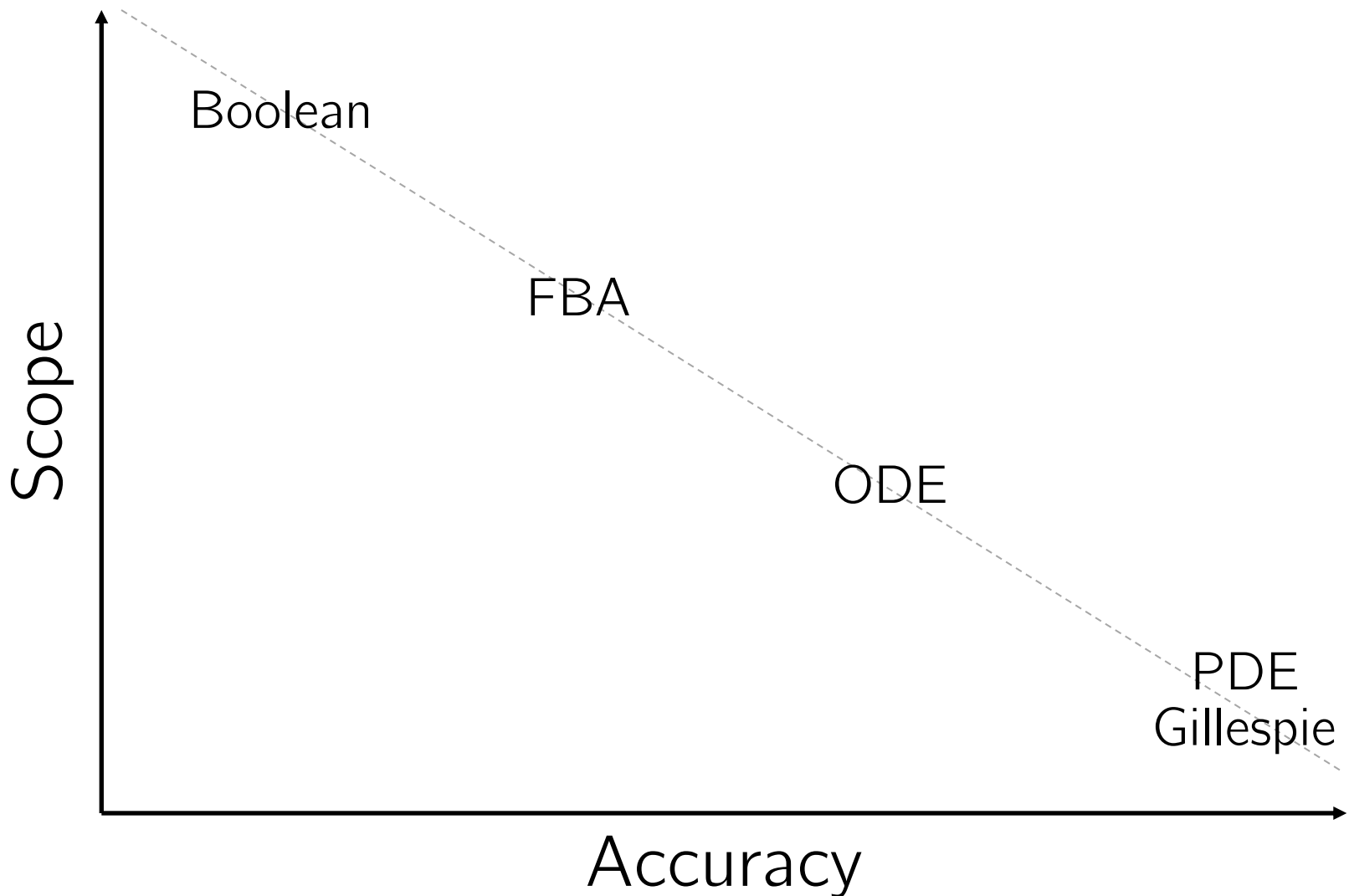
# Phenomenological approaches: PROM



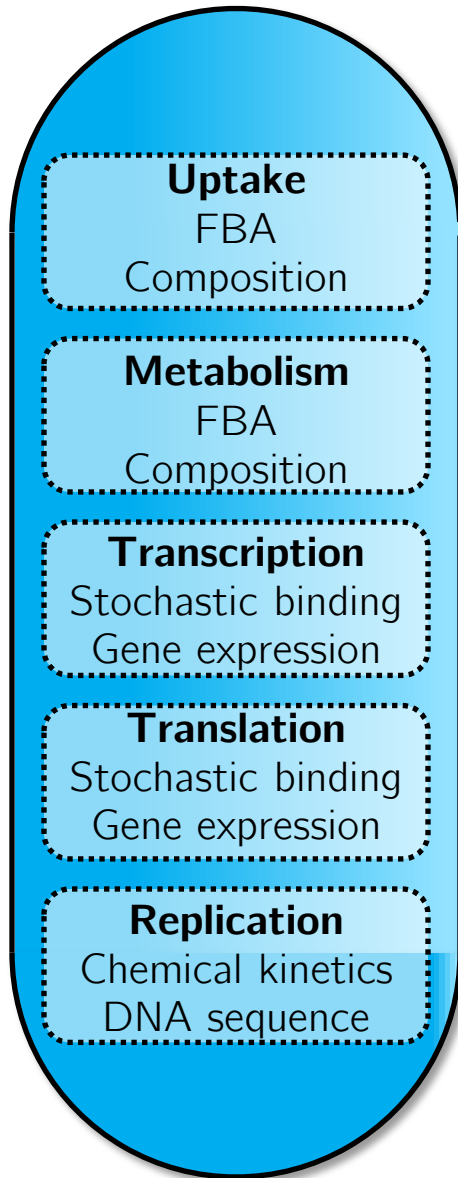
# Mechanistic approaches: Coarse-grained MD



# Mechanistic modeling formalisms



# Multi-algorithm modeling



- Models composed of submodels
- Submodels describe individual pathways
- Submodels represented using different math
- Enables representation of well- and poorly-studied pathways

# Synonyms

## **Uptake**

FBA  
Composition

## **Metabolism**

FBA  
Composition

## **Transcription**

Stochastic binding  
Gene expression

## **Translation**

Stochastic binding  
Gene expression

## **Replication**

Chemical kinetics  
DNA sequence

- Model composition
- Integrative modeling
- Hybrid modeling
- Multi-algorithm modeling
- Multi-physics modeling
- Hierarchical modeling

# Advantages of multi-algorithm models

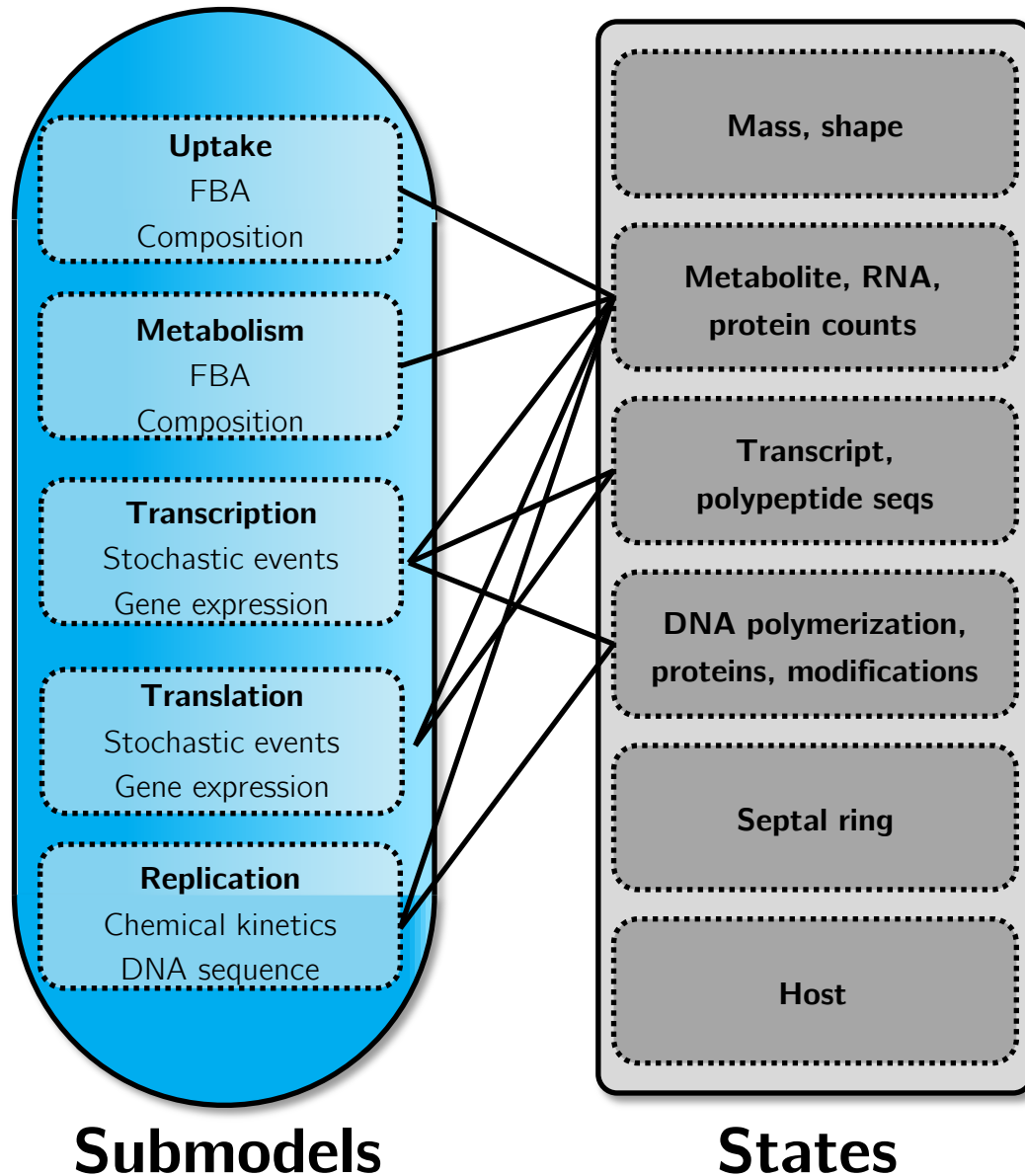
## Advantages

- Model composition
  - Encapsulates pathways
  - Enables collaborative model design, estimation, and testing
- Combines coarse- and fine-grained submodels
- Enables thorough representation of knowledge and data
- Avoids unknown parameters
- Simplifies parameter estimation
- Enables more comprehensive and more accurate models

## Disadvantages

- Few established methods and software
- Challenging to build, simulate, identify and verify
- Require expert knowledge and intense effort

# WC modeling building



1. Curate data
2. Construct submodels
3. Define global state
4. Combine submodels
5. Simulate

# Model consistency

- Utilize same species, reaction names
- Resolve conflicting species/reaction representations across submodels
- Calculate RNA and protein sequences from gene sequences
- Calculate species molecular weights from structures
- Calculate transcription, translation, RNA degradation reactions from sequences
- Calculate cell mass, volume from species counts and molecular weights



# Reproducibility

**Every model element should be defined without references to external databases**

- Metabolites: InCHI, SMILES
- RNA, protein: sequences
- Reactions: stoichiometry

**Cross references should be provided where possible**

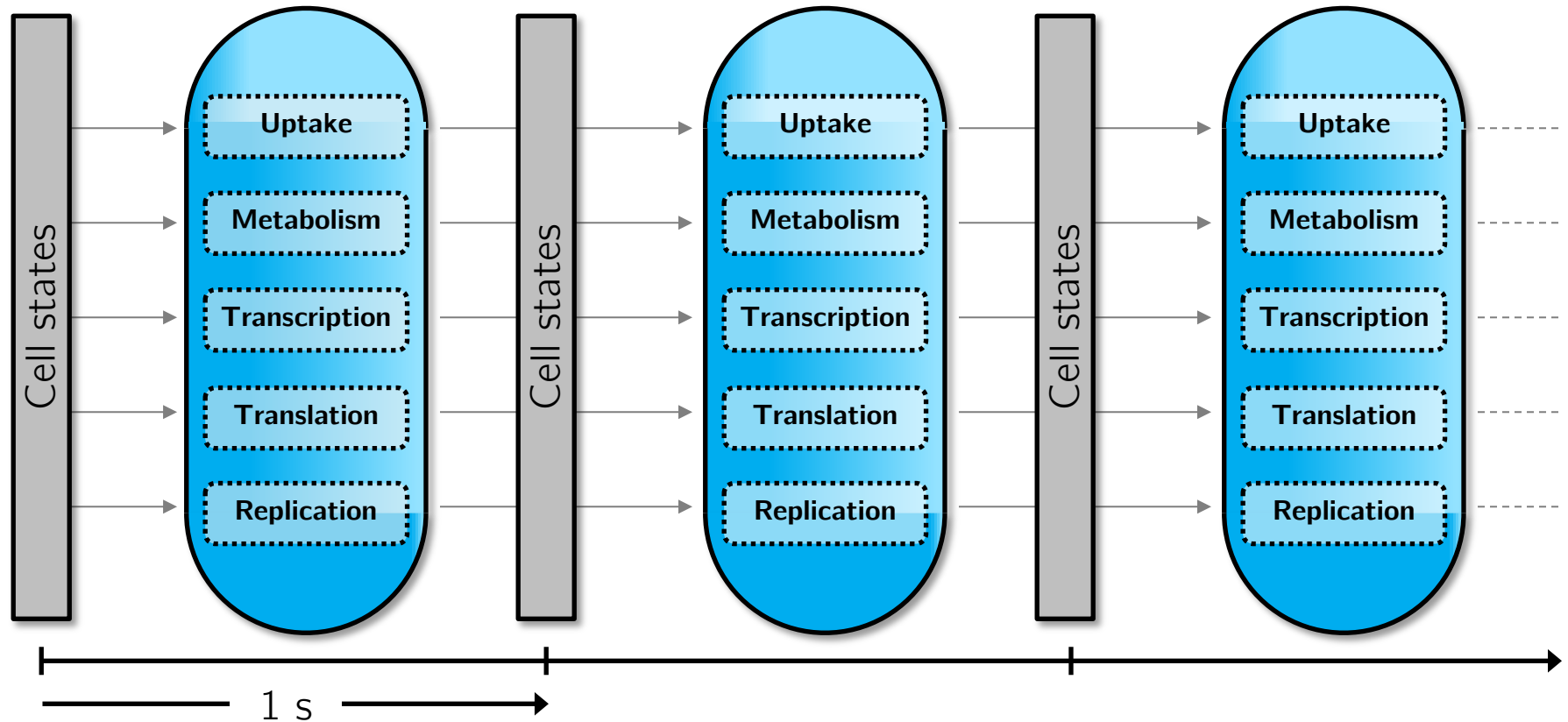
- Metabolites: ChEBI
- RNA: NCBI
- Proteins: UniProt
- Reactions: EC numbers

**Every data source and model assumption should be recorded**

# Multi-algorithm simulation

- Area of active research
- A few algorithms have been developed
- All have limitations
- Tutorial: foundational concepts

# Multi-algorithm simulation



**Assumption: pathways are independent over short time periods**

# Concurrent submodel integration

## **Approximate/continuous/timestep**

- Simulate models over short timesteps
- Synchronize models between timesteps
- Low computational cost
- Low numerical accuracy

## **Exact/discrete**

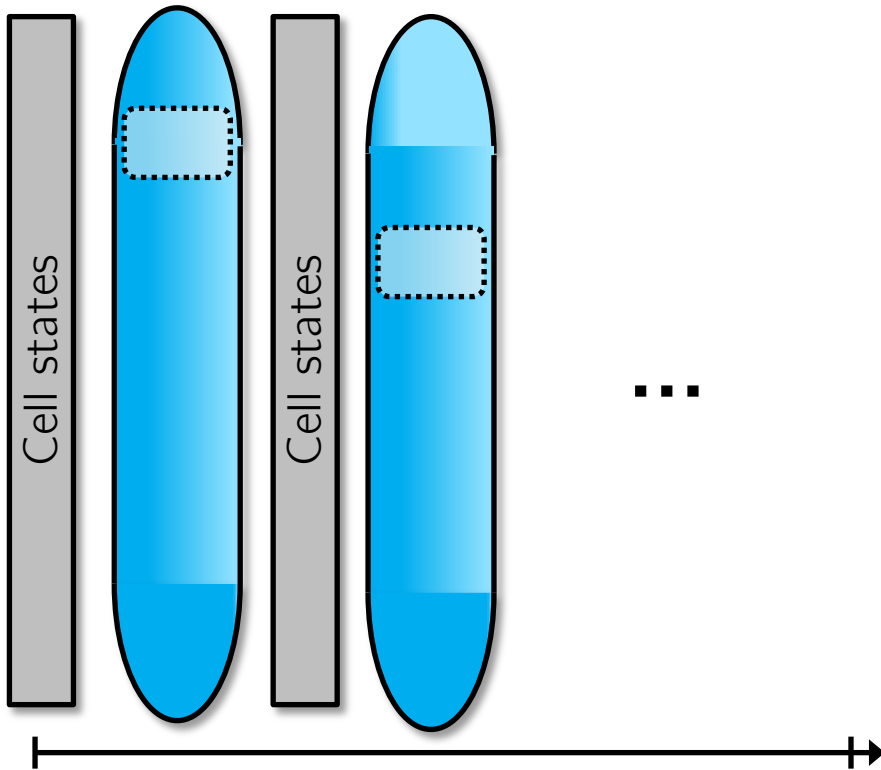
- Resolve order of every individual reaction
- Synchronize submodels after every reaction
- High computational cost
- High numerical accuracy

# Approximate simulation

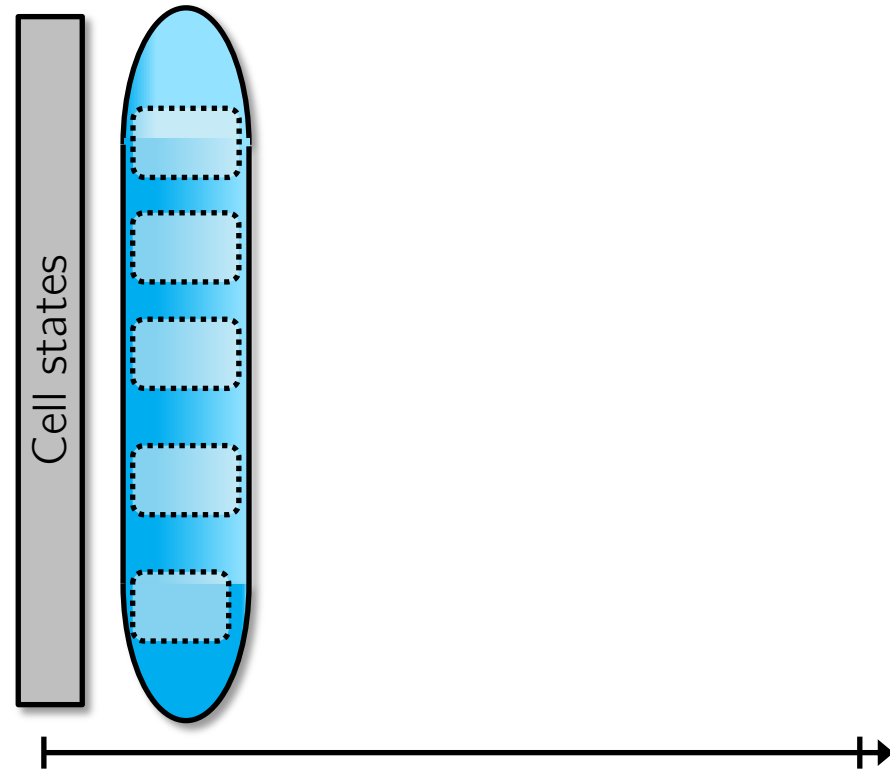
- Divide time into small steps
- Integrate submodels separately
- Update global state
- Reduce timestep until results converge

# Submodel execution order

## Asynchronous integration



## Synchronous integration



# Exact simulation

## Based on SSA

**Where possible, convert submodels to discrete submodels and simulate using SSA**

- Simulate ODE with SSA
- Add explicit time to Boolean models
- Merge SSA submodels

**Discretely schedule continuous submodel updates**

- Treat continuous submodels as a single discrete pseudoreaction whose stoichiometry is time-variant

# Exact simulation

- Requires resolution of many events
- Computationally expensive
- Implement using parallel discrete event simulation



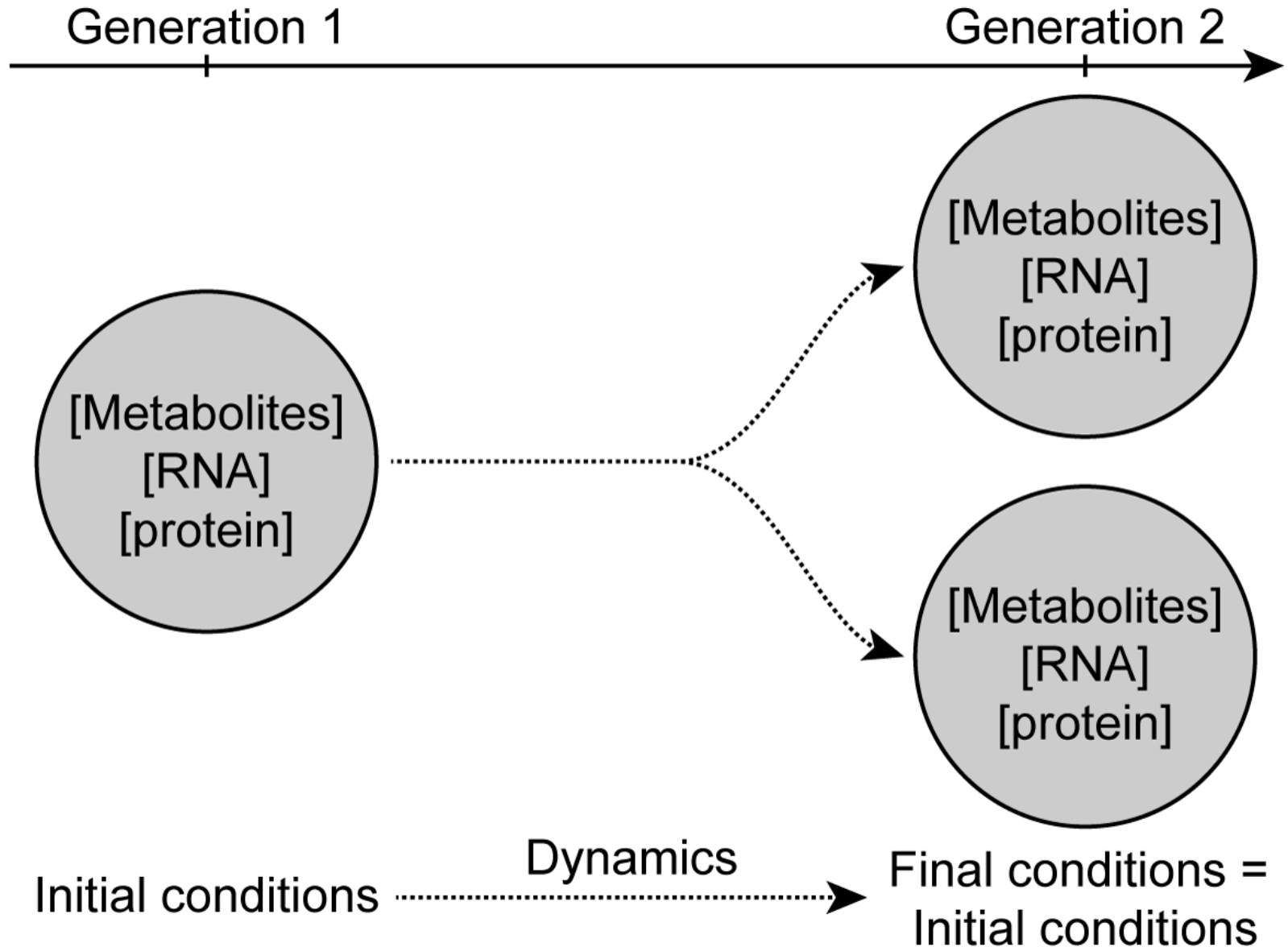
# Parameter estimation

Compare model to experimental data

Numerically minimize prediction error

Theory provides additional constraints

# Cell theory provides periodic boundary constraint



# Cell theory provides periodic boundary constraint

- Phenotype distribution constant over generations
- $\langle \text{initial conditions} + \int \text{dynamics } dt \rangle = \langle \text{initial conditions} \rangle$
- Example:
  - $\langle [\text{RNA}]_0 + \int (\text{RNA production} - \text{RNA decay}) dt \rangle = \langle [\text{RNA}]_0 \rangle$
  - $[\text{RNA}](t) = [\text{RNA}]_0 e^{\ln(2)t/\tau}$
  - $\text{RNA production} = k e^{\ln(2)t/\tau}$
  - $\text{RNA decay} = \frac{\ln(2)}{\tau_{1/2}} [\text{RNA}]$
  - $k = \left( \frac{\ln(2)}{\tau} + \frac{\ln(2)}{\tau_{1/2}} \right) [\text{RNA}]$

# Verification

## **Statically verify model**

- E.g. all reactions mass and charge balanced

## **Dynamically verify submodels**

- E.g. protein content doubles over cell cycle duration

## **Dynamically verify entire model**

- E.g. cell divides in observed doubling time

# Software engineering

## Organization

- Uses methods, objects, and modules to encapsulate data and procedures
- Repository used to track revisions

## Style

- Clearly and consistently named variables, methods, classes

## Annotation

- Author name, last updated date
- Commented

## Testing

- Code is tested formally using unit testing
- Tests evaluated at each revision using continuous integration

# Exercises

# Exercises

## **1. Model building**

Assemble a small WC model from several data points

## **2. Model alignment and parameter estimation**

Use cell theory to identify parameter values

## **3. Multi-algorithm simulation**

Implement hybrid FBA/SSA simulator

## **4. Individual submodel simulation**

Simulation single submodel

## **5. Best practices**

Annotate model

# Physiology

Model reflects typical cell biology

Motivated by *M. pneumoniae*



# Submodels

	Algorithm	Reactions	Enzymes
Metabolism	FBA	Several	Several
Transcription	SSA	1 per RNA	RNA polymerase
Translation	SSA	1 per protein	Ribosome
RNA degradation	SSA	1 per RNA	Rnase

# Metabolism submodel

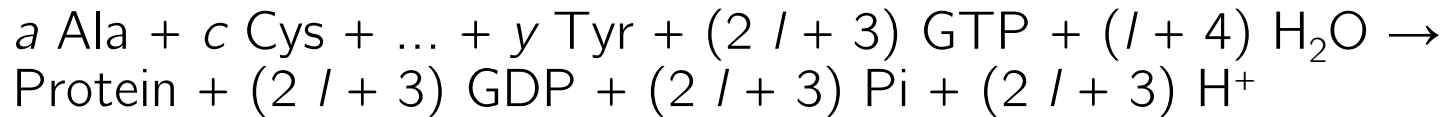
Metabolite	Biomass	Produce for other pathways	Recycle from other pathways	Import from media
Glucose				Y
Nucleobases				Y
NMPs	Y		Y	
GDP	Y		Y	
NTPs	Y	Y		
Amino acids	Y	Y		Y
NAD				Y
PPi	Y		Y	
Pi	Y		Y	Y
H <sub>2</sub> O	Y	Y	Y	Y
H <sup>+</sup>	Y		Y	Y
O <sub>2</sub>				Y
CO <sub>2</sub>				Y
Internal metabolites				

# Non-metabolic submodels

## Transcription



## Translation



## RNA degradation



# Species

## **Enzymes needed for submodels**

- RNA
- Protein

## **Metabolic reactants and byproducts of all submodels**

- Amino acids
- NTPs
- NDPs
- NMPs
- PPi
- Pi
- H<sub>2</sub>O
- H<sup>+</sup>

# Rate laws

## Metabolism

- $v = v_{\max}[\text{Enzyme}]$

## Transcription

- $v = v_{\max} \min_N \left( \frac{[NTP]}{K_M + [NTP]} \right) [\text{RnaPol}]$
- $K_M = [NTP]$

## Translation

- $v = v_{\max} \min_{AA} \left( \frac{[AA]}{K_M + [AA]} \right) [\text{RNA}][\text{Ribosome}]$
- $K_M = [AA]$

## RNA degradation

- $v = v_{\max} \frac{[\text{RNA}]}{K_M + [\text{RNA}]} [\text{RNase}]$
- $K_M = [\text{RNA}]$

# Files

File	Description
Model.xlsx	Template model description and data to build model
exercice*.py	Template code for exercises
model.py	Reads model from Excel into Python object
analysis.py	Plots simulation results
util.py	Utility methods

# Implementation: Classes

## Model represented by Model object

- Submodels
  - Species/compartments
  - Reactions
- Compartments
- Species
- Reactions
  - Participants
    - Species
    - Compartment

# Implementation: Methods

- `model.getModelFromExcel(<fileName:string>)`  
Reads model from Excel
- `model.Model.calcInitialConditions()`  
Calculates initial conditions
- `model.Submodel.updateLocalState(<model:Model>),`  
`model.Submodel.updateGlobalState(<model:Model>)`  
Updates submodel state from model, updates global state from submodel
- `model.Submodel.calcReactionRates(<reactions:list>,`  
`<speciesConcentrations:dict>)`  
Returns array with rates of every reaction in a submodel
- `model.FbaSubmodel.calcBounds(<timeStep:int>)`  
Returns array with upper and lower bounds for reactions
- `model.Submodel.executeReaction(<speciesCounts:dict>,`  
`<reaction:Reaction>)`  
Updates species counts with stoichiometry of the reaction
- `model.Model.calcMass(), model.Model.calcVolume()`  
Updates cell mass and volume



# Implementation: Cell state

- `model.Model.speciesCounts`  
Represents species copy numbers as numpy array
  - Rows represent species
  - Columns represent compartments
- `model.Submodel.speciesCounts`  
Represents species copy numbers as dict
- `model.Model.getSpeciesCountsDict()`,  
`model.Model.setSpeciesCountsDict(<counts:dict>)`  
Gets, sets dict of species counts
- `model.Model.getSpeciesConcentrations()`,  
`model.Submodel.getSpeciesConcentrations()`  
Gets species concentrations
- `model.Model.mass`, `model.Model.volume`,  
`model.Model.extracellularvolume`, `model.Submodel.volume`,  
`model.Submodel.extracellularVolume`  
Cell mass, cell volume, extracellular volume
- `model.FbaSubmodel.growth`,  
`model.FbaSubmodel.reactionFluxes`  
Growth rate and reaction fluxes

# Implementation: Methods

- `model.Model.getComponentById(<id:string>)`  
Returns model component with id
- `model.Reaction.getStoichiometryString()`  
Returns string representation of reaction
- `analysis.plot(  
    <model:Model>,  
    <time:numpy.ndarray>,  
    <volume:numpy.ndarray>,  
    <speciesCounts:numpy.ndarray>,  
    <selectedSpeciesCompartments: list of ids e.g. "ATP[c]">,  
    <units:str e.g. "mM">,  
    <fileName:str>)`  
Plots simulation results
- `numpy.random.seed(<seed:int>)`  
Seeds PRNG

# Exercise 1: Building models from data

- Learn how to build WC models from data
- Build list of species, reactions from metabolic reconstruction
- Use transcription, translation, RNA degradation templates to create individual reactions
- Enumerate initial conditions from RNA and metabolite copy numbers/concentrations

# Exercise 2: Aligning submodels

- Learn how to build internally consistent models by
  - Calculating transcription, translation, RNA degradation rate parameters
  - Calculating metabolism output pseudoreaction (FBA objective)
- Use cell theory to calculate rate constants
- Sum net effects of non-metabolic submodels and cell composition to calculate metabolism output (FBA objective)
  - $\text{Production}_i = [\text{Metabolite}]_i + \sum_j S_{ij} \int_0^\tau v_j dt$

# Rate parameters

## Metabolism

- Curated from literature
- $v = v_{\max}[\text{Enzyme}]$

## Transcription

- Calculated from RNA copy number, half-life, and cell cycle length
- $v = \text{degradation} + \text{dilution} = \frac{\ln(2)}{\tau_{\text{RNA}}} [\text{RNA}] + \frac{\ln(2)}{\tau_{\text{cell}}} [\text{RNA}] = v_{\max} \min\left(\frac{[\text{NTP}]}{K_M + [\text{NTP}]}\right) [\text{RnaPol}]$
- $K_M = [\text{NTP}]$

## Translation

- Calculated from amino acids, RNA, and ribosome concentrations
- $v = \text{dilution} = \frac{\ln(2)}{\tau_{\text{cell}}} [\text{Protein}] = v_{\max} \min\left(\frac{[\text{AA}]}{K_M + [\text{AA}]}\right) [\text{RNA}][\text{Ribosome}]$
- $K_M = [\text{AA}]$

## RNA degradation

- Characterized by typical RNA half-life
- $v = v_{\max} \frac{[\text{RNA}]}{K_M + [\text{RNA}]} [\text{RNase}] = \frac{\ln(2)}{\tau_{\text{RNA}}} [\text{RNA}]$
- $K_M = [\text{RNA}]$

# Rate parameters

Submodel	Vmax	Km (mM)
Metabolism	Given	N/A
Transcription	$2.33 \times 10^{-4} \text{ s}^{-1}$	1.00
Translation	$2.66 \times 10^2 \text{ M}^{-1} \text{ s}^{-1}$	5.00
RNA degradation	$2.31 \times 10^{-4} \text{ s}^{-1}$	$1.81 \times 10^{-4}$

# Metabolism production

Metabolite	Molecules/cell	Metabolite	Molecules/cell
ALA	$3.42 \times 10^4$	PHE	$2.39 \times 10^4$
ARG	$4.09 \times 10^4$	PRO	$3.44 \times 10^4$
ASN	$2.32 \times 10^4$	SER	$4.45 \times 10^4$
ASP	$2.43 \times 10^4$	THR	$3.62 \times 10^4$
ATP	$1.10 \times 10^6$	TRP	$2.22 \times 10^4$
CTP	$1.16 \times 10^6$	TYR	$2.20 \times 10^4$
CYS	$2.23 \times 10^4$	UTP	$1.13 \times 10^6$
GLN	$2.47 \times 10^4$	VAL	$3.38 \times 10^4$
GLU	$2.27 \times 10^4$	AMP	$-1.04 \times 10^6$
GLY	$3.12 \times 10^4$	CMP	$-1.10 \times 10^6$
GTP	$1.73 \times 10^6$	GDP	$-5.81 \times 10^5$
H2O	$1.52 \times 10^9$	GMP	$-1.05 \times 10^6$
HIS	$2.16 \times 10^4$	H	$-4.97 \times 10^6$
ILE	$2.93 \times 10^4$	PI	$-4.71 \times 10^5$
LEU	$4.12 \times 10^4$	PPI	$-4.38 \times 10^6$
LYS	$2.20 \times 10^4$	UMP	$-1.07 \times 10^6$
MET	$2.14 \times 10^4$		

# Exercise 3: Multi-algorithm simulation

## Goal

- Learn how to simulate multi-algorithm models by
  - Implementing a simulator for FBA and SSA

## Approach

- Combine non-metabolic submodels into a single submodel
- Synchronize state between FBA and SSA models
- Use SSA and FBA and simulate the combined model



# Pseudocode

Initialize state

for  $t = 0$ ;  $t < t_{\text{Max}}$ ;  $t += dt$

    Simulate SSA submodels for  $dt$

$t_2 = t$

        Calculate SSA reaction rates

        Calculate time to next SSA reaction

$dt_2 = \text{exponential}(\text{sum}(\text{rates}))$

        Calculate next SSA reaction:

$\text{multinomial}(\text{rates})$

        Update time:  $t_2 = t_2 + dt_2$

        Update state

    Simulate FBA submodel

        Calculate growth rate

        Update state

# Exercise 4: Testing submodels

## Goal

- Learn how to test individual submodels so that submodels can be developed separately by
  - Testing metabolism submodel in a meaningful way without simulating the other submodels

## Approach

- Simulate metabolism submodel coupled with other reduced or “mocked” versions of the other submodels
- The other submodels can be mocked by replacing them with a single pseudoreaction with represents their net effect

# Pseudocode

Calculate net transcription, translation, RNA degradation reactions

Initialize state

$t = 0$

while  $t < t_{\text{Max}}$

    Calculate growth and fluxes

    Update state

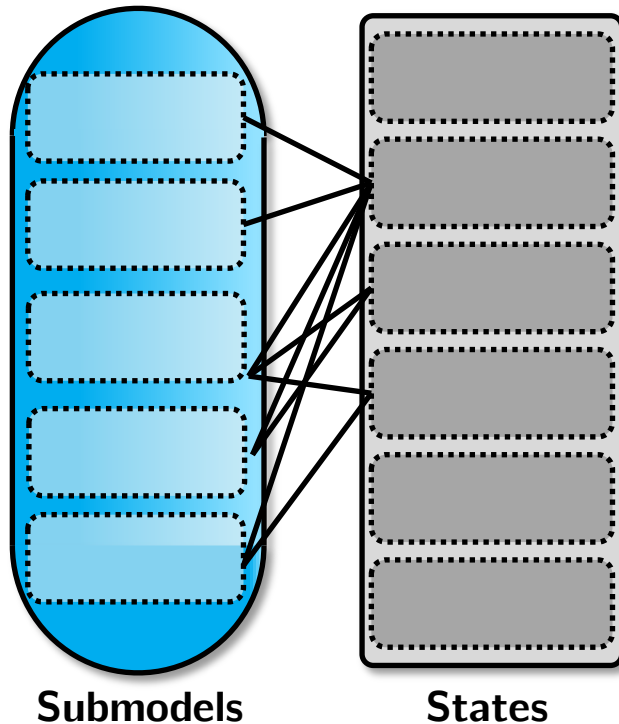
    Update state by growth \* net transcription, translation, RNA degradation reactions

    Update time:  $t = t + dt$

# Exercise 5: Best practices

- Learn and reinforce best WC modeling practices by
  - Annotating model with cross references
  - Calculating chemical formulae, molecular weights from structures
  - Testing model
- Use ChEBI, Enzyme, NCBI, and UniProt to annotate species and reactions
- Use ChEBI to annotate metabolite structures
- Use mcule to calculate chemical formulae, molecular weights, charges

# Summary



**Definition** Represent well- and poorly-characterized pathways

**Motivation** Represent well- and poorly-characterized pathways

**Advantages** Enables comprehensive models

**Disadvantages** Complex, few methods and tools

**Construction** Map submodels onto common state

**Simulation** Concurrently integrating submodels

**Outlook** Research needed to develop simulation algorithms

# Acknowledgements

Yin Hoon Chew, Mount Sinai

Javier Carrera, Stanford



# Feedback

We're very interested in improving the course

**Please complete tutorial survey**