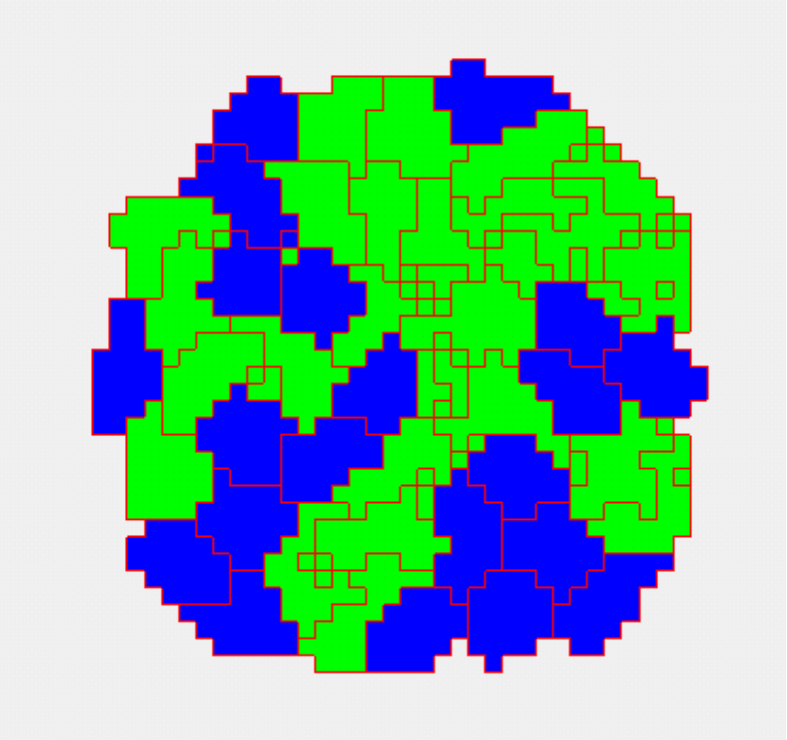
Integrating CompuCell3D and Tellurium/SBML

Julio M. Belmonte

Indiana University, Bloomington

CPM modeling Scope

- Cellular behaviors:
  - Location
  - Volume
  - Shape
  - Movement
  - Adhesion
  - Mitosis
  - Death
  - Differentiation
  - Polarization
  - Etc…

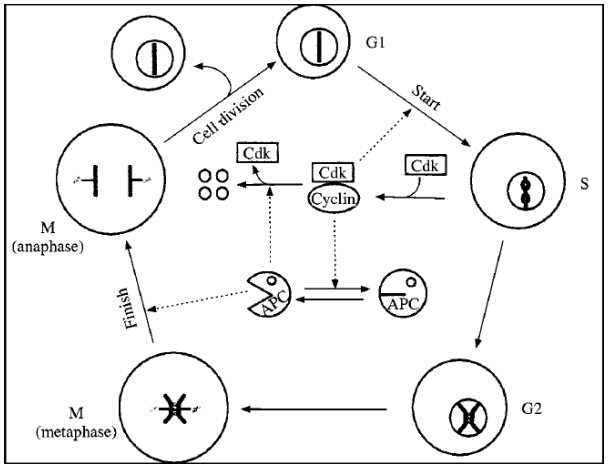


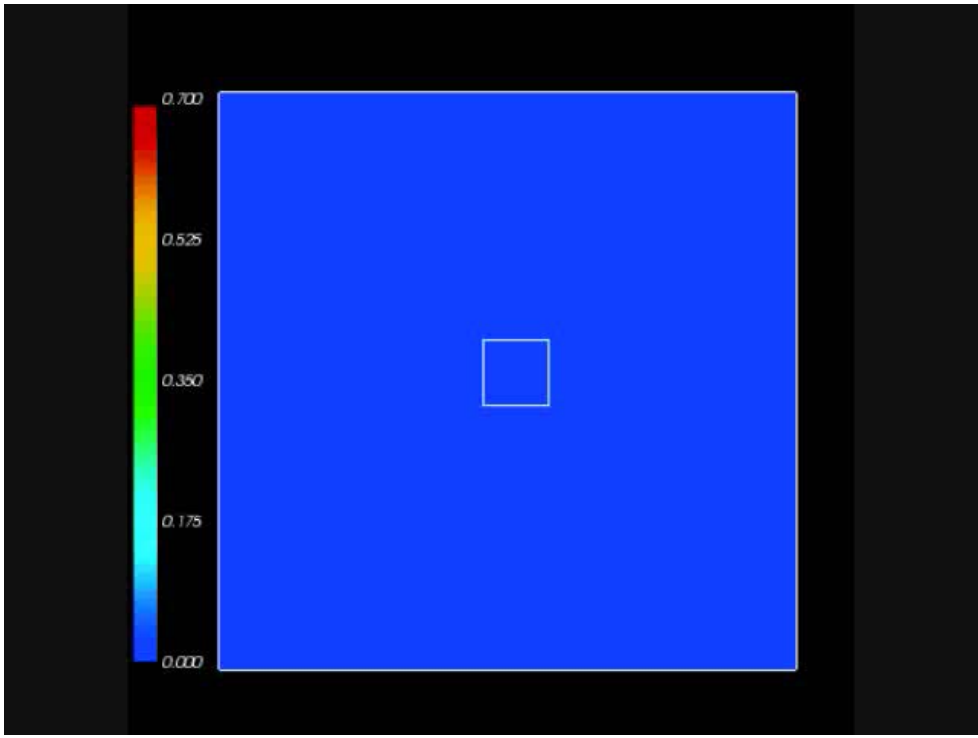
Subcellular modelling

- Biochemical Kinetics:
  - Cell-Cycle
  - Circadian rhythms
  - Cardiac rhythms
  - cAMP oscillations
  - Delta-Notch patterning
  - WNT pathway
  - FGF pathway
  - Etc…

Subcellular modelling

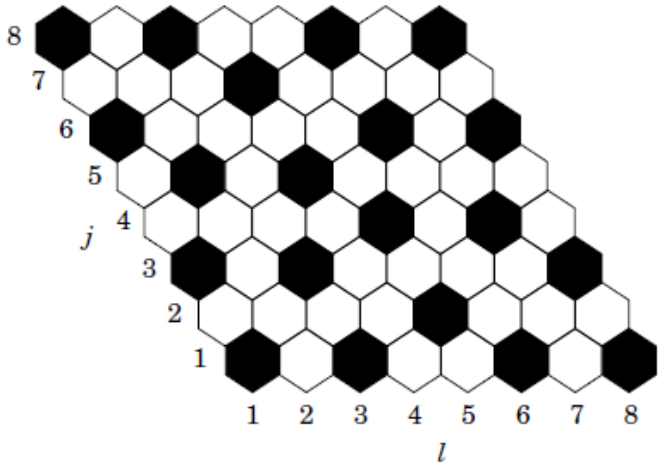
- Biochemical Kinetics:
  - Cell-Cycle
  - Circadian rhythms
  - Cardiac rhythms
  - cAMP oscillations
  - Delta-Notch patterning
  - WNT pathway
  - FGF pathway
  - Etc…

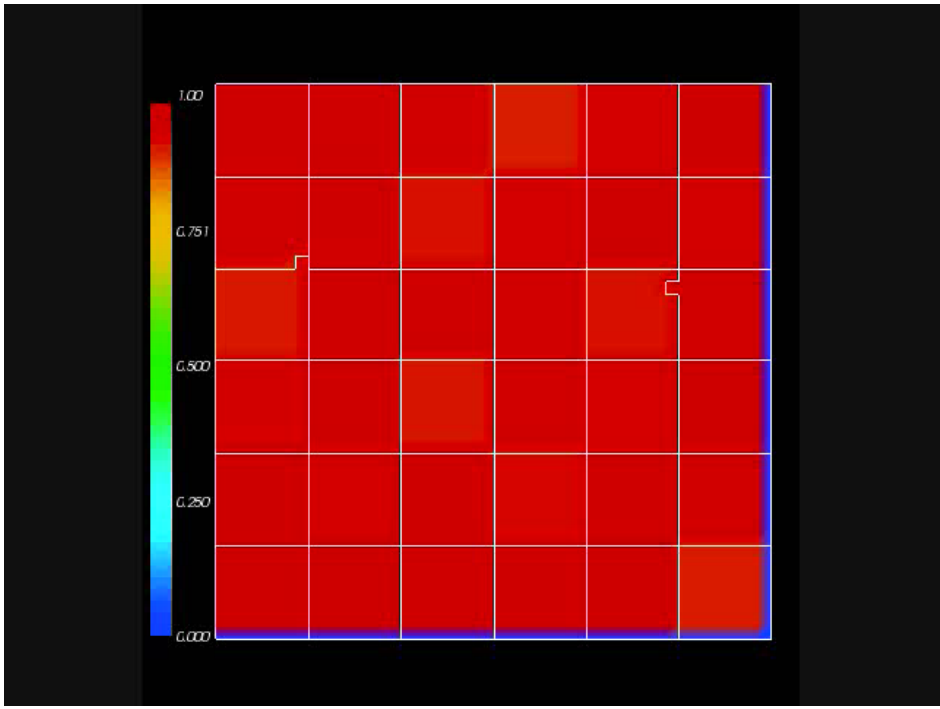




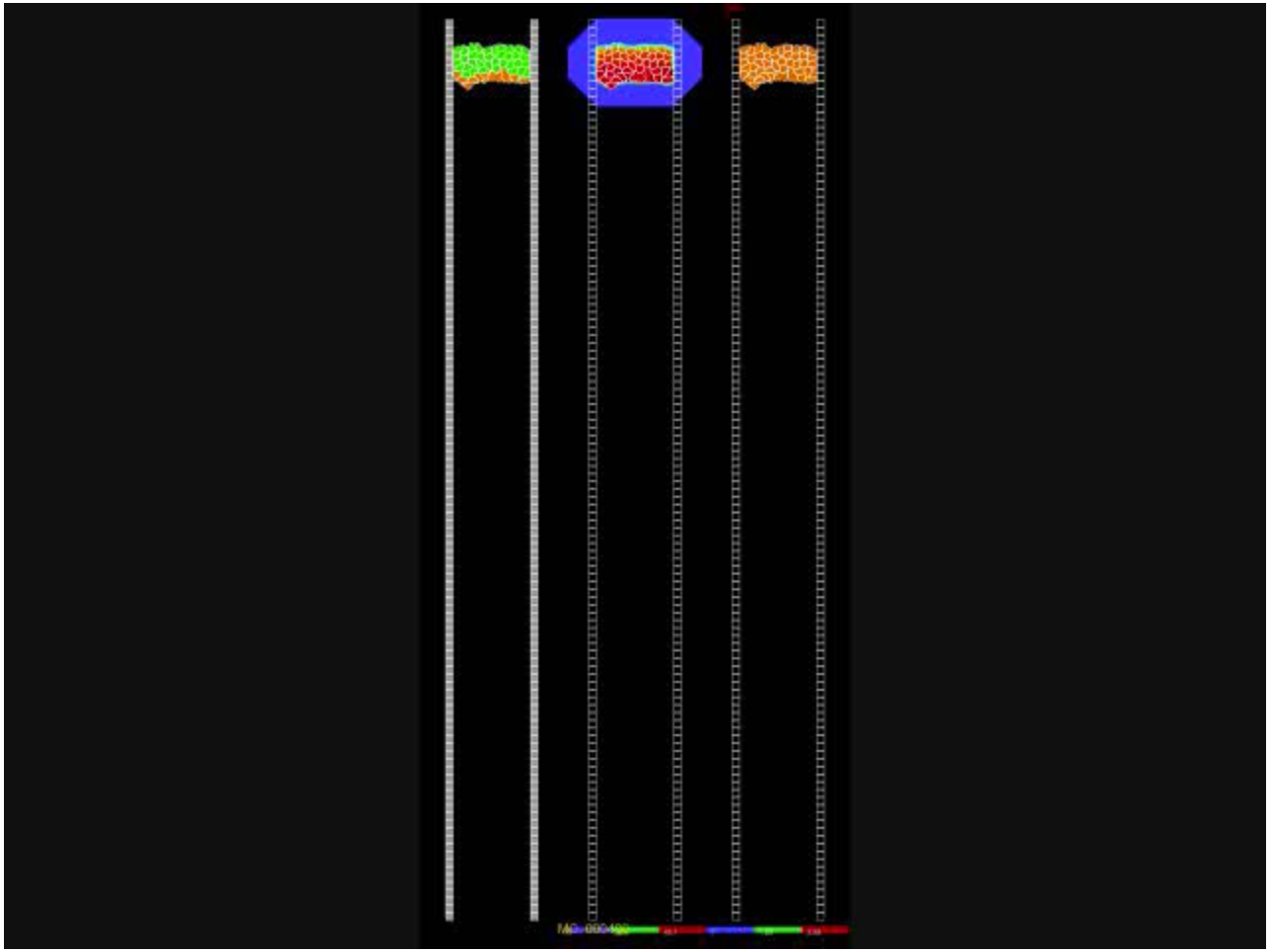
Subcellular modelling

- Biochemical Kinetics:
  - Cell-Cycle
  - Circadian rhythms
  - Cardiac rhythms
  - cAMP oscillations
  - Delta-Notch patterning
  - WNT pathway
  - FGF pathway
  - Etc…



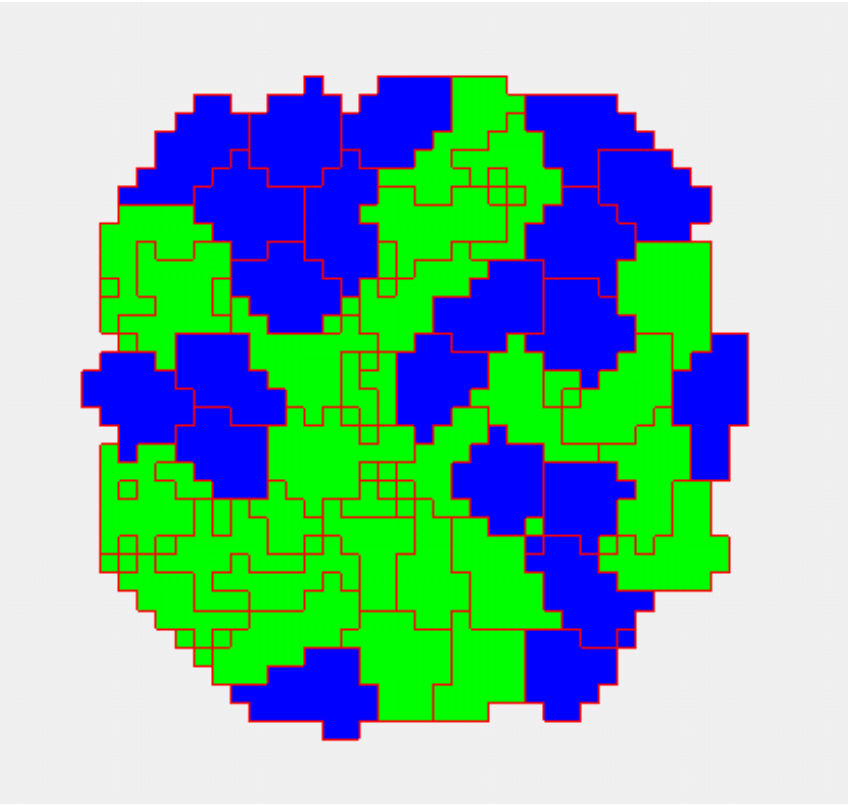


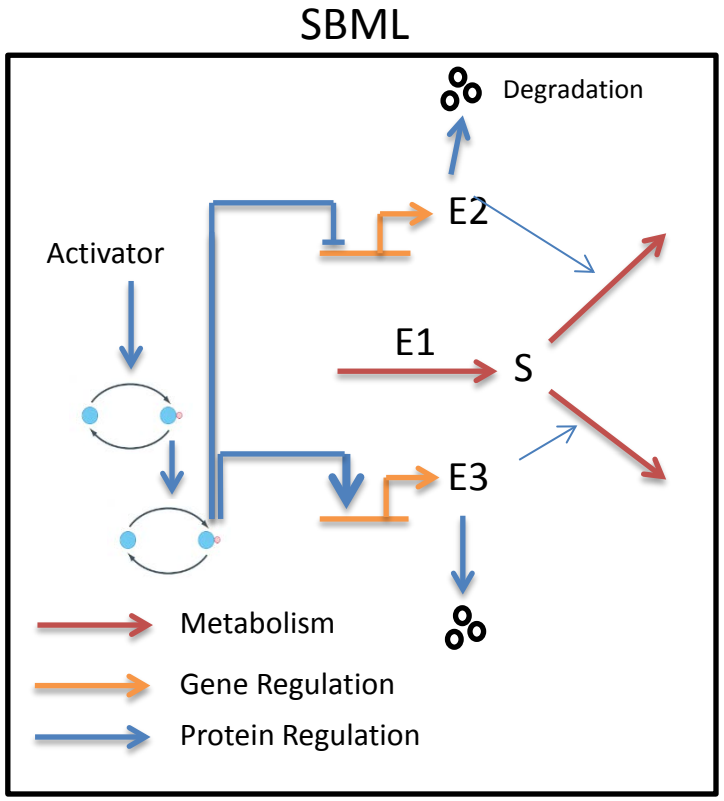
Multiscale model - Somitogenesis



CC3D+SBML

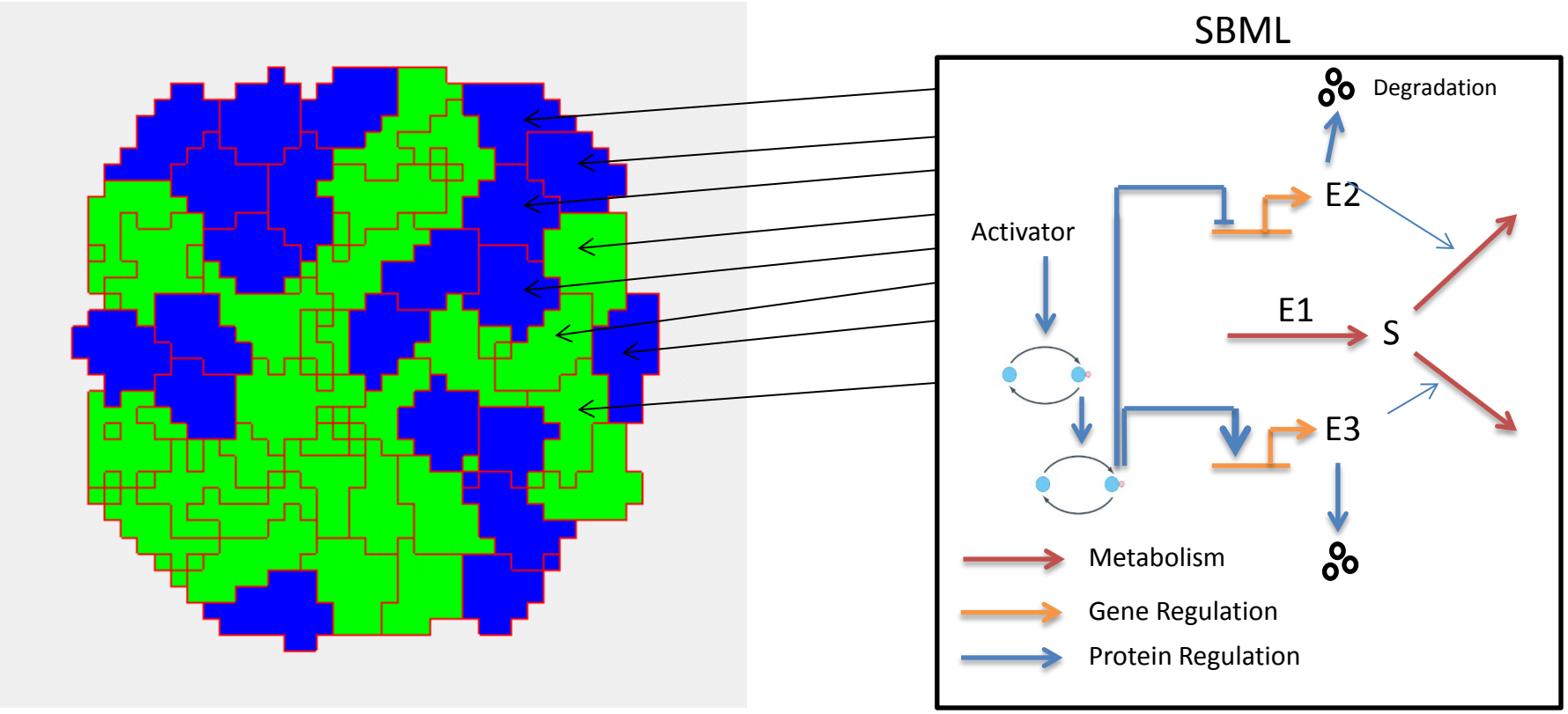
- CC3D:





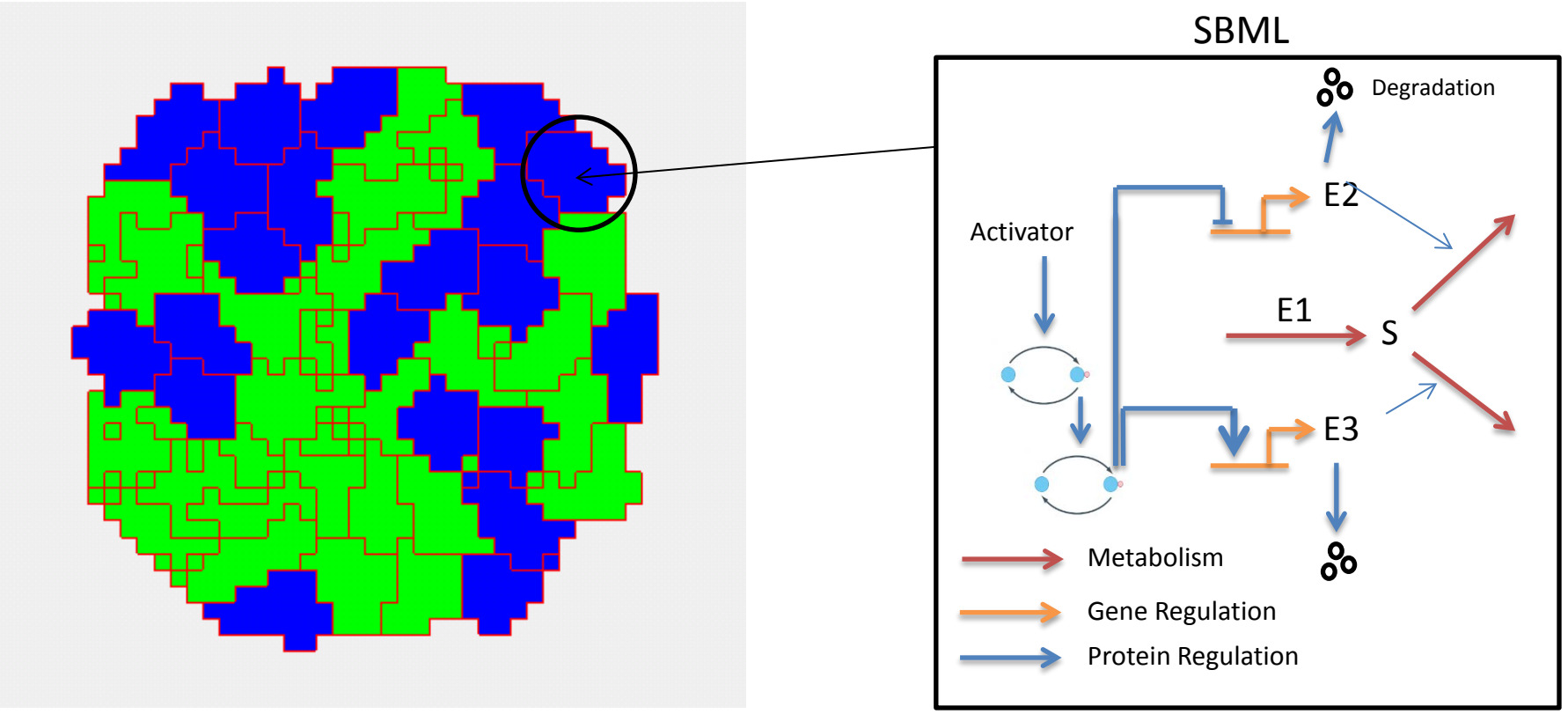
CC3D+SBML

- CC3D cells:



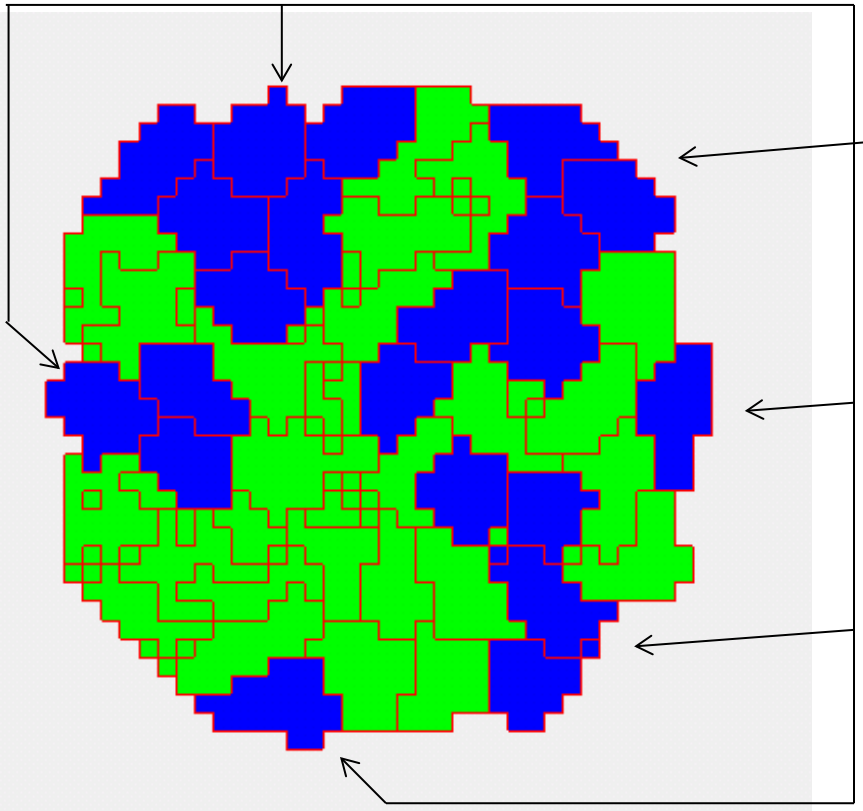
Integration with CC3D – 1 cell

- CC3D:

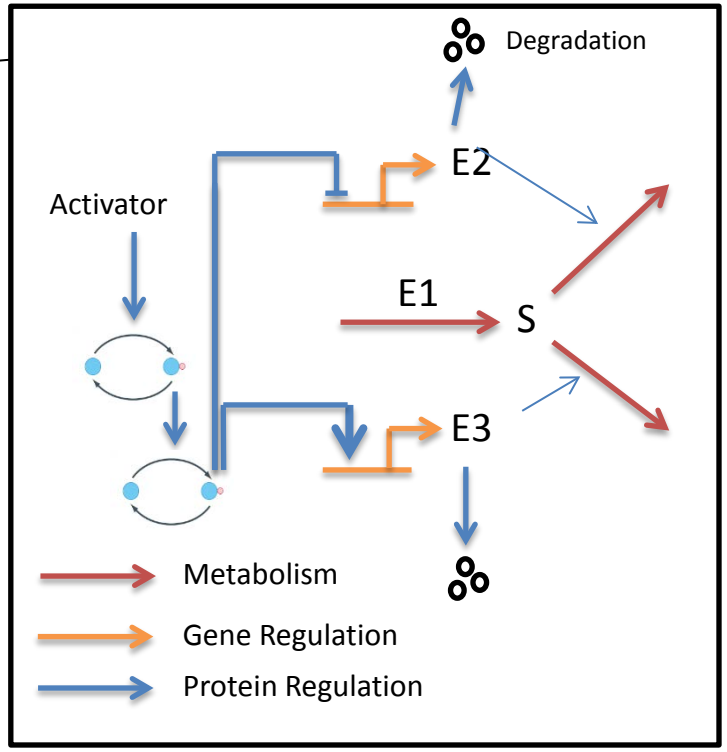


Integration with CC3D – cell type

- CC3D:

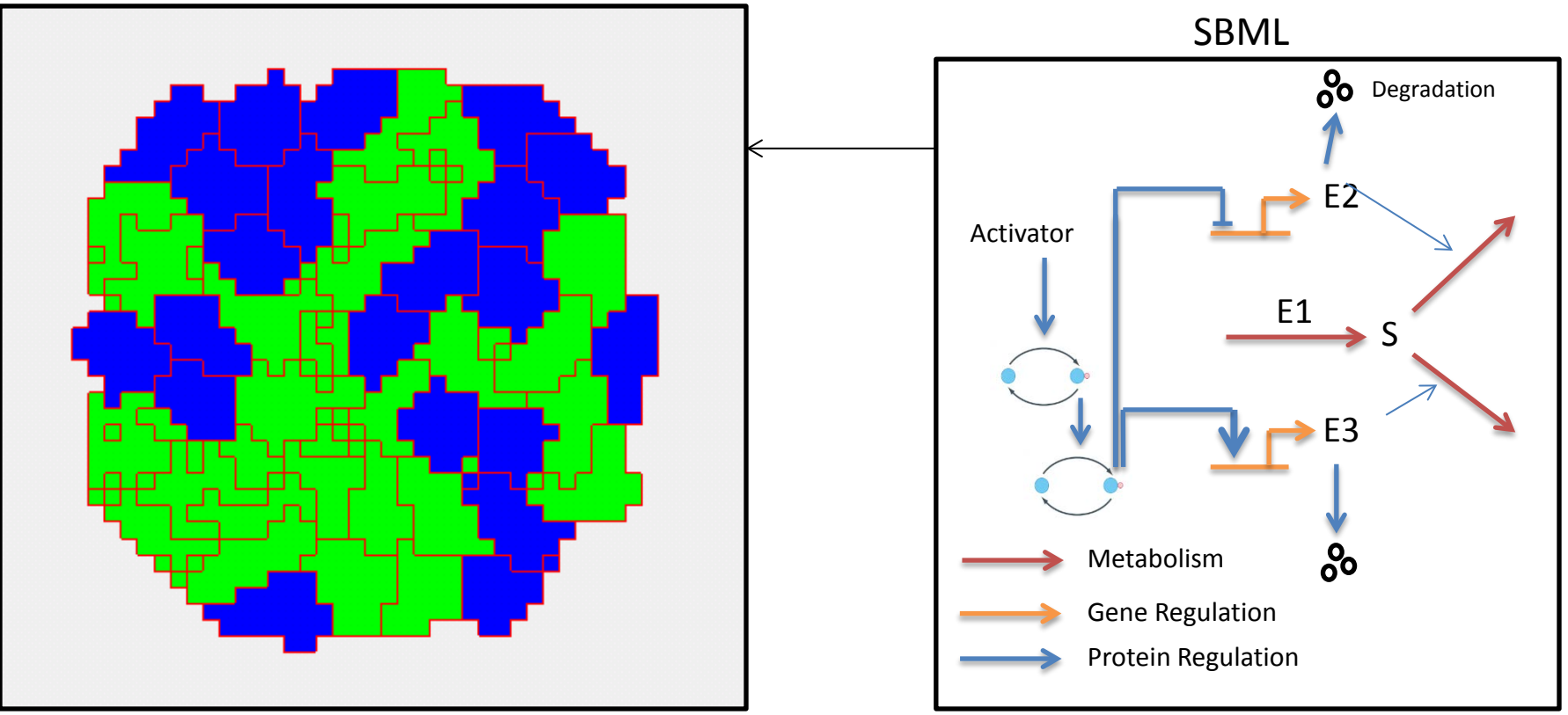


SBML



Integration with CC3D – Simulation

- CC3D:



Integration with CC3D – cell

```
class <someClass>(SteppableBasePy):
  def __init__(self,_simulator,_frequency):
    SteppableBasePy.__init__(self,_simulator,_frequency)

  def start(self):
    # Define a SBML model
    Path = <ModelPath>   # Path where the model is stored
    Name = <ModelName>   # Name of the model
    Step = <timeStep>    # Time step of integration

    # Add SBML model to a cell
    for cell in self.cellList:
       self.addSBMLToCell(_modelFile=Path, _modelName=Name, _cell=cell, _stepSize=Step)

  def step(self,mcs):
    # Iterate the model (run it for the time step specified on the load command)
    self.timeStepSBML()

    # Get the parameter value or molecular concentration from a cell
    for cell in self.cellList:
       <var> = self.getSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _cell=cell)

       # Set a new parameter value or molecular concentration value for a cell
       self.setSBMLValue(_modelName=<Name>,_valueName=<param/molecule>,_value=<val>,_cell=cell)

       # Change integration step of a cell
       self.setStepSizeForCell(_modelName=<Name>, _cell=cell, _stepSize=<newStep>)

    # Copy a SBML model from one cell to another
    self.copySBMLs(_fromCell=cell1, _toCell=cell2, _sbmlNames=[<ModelName1>,<ModelName2>,…])

    # Delete a SBML model from a cell
    self.deleteSBMLFromCell(_modelName=<Name>, _cell=cell)
```

Integration with CC3D – cell id

```
class <someClass>(SteppableBasePy):
  def __init__(self,_simulator,_frequency):
    SteppableBasePy.__init__(self,_simulator,_frequency)

  def start(self):
    # Define a SBML model
    Path = <ModelPath>   # Path where the model is stored
    Name = <ModelName>   # Name of the model
    Step = <timeStep>    # Time step of integration

    # Add SBML model to a cell
    for cell in self.cellList:
       self.addSBMLToCellIds(_modelFile=Path, _modelName=Name, _ids=[cell.id], _stepSize=Step)

  def step(self,mcs):
    # Iterate the model (run it for the time step specified on the load command)
    self.timeStepSBML()

    # Get the parameter value or molecular concentration from a cell
    for cell in self.cellList:
       <var> = self.getSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _cell=cell)

       # Set a new parameter value or molecular concentration value for a cell
       self.setSBMLValue(_modelName=<Name>,_valueName=<param/molecule>,_value=<val>,_cell=cell)

       # Change integration step of a cell (overwrites the initial step value)
       self. setStepSizeForCellIds(_modelName=<Name>, _ids=[cell.id], _stepSize=<newStep>)

    # Copy a SBML model from one cell to another
    self.copySBMLs(_fromCell=cell1, _toCell=cell2, _sbmlNames=[<ModelName1>,<ModelName2>,…])

    # Delete a SBML model from a cell
    self.deleteSBMLFromCellIds(_modelName=<Name>, _ids=[id1,id2,…])
```

Integration with CC3D – cell types

```
class <someClass>(SteppableBasePy):
  def __init__(self,_simulator,_frequency):
    SteppableBasePy.__init__(self,_simulator,_frequency)

  def start(self):
    # Define a SBML model
    Path = <ModelPath>    # Path where the model is stored
    Name = <ModelName>    # Name of the model
    Step = <timeStep>     # Time step of integration

    # Add SBML model to a cell type (all cells of that cell type will have it)
    self.addSBMLToCellTypes(_modelFile=Path,_modelName=Name,_types=[self.TYPE1…],_stepSize=Step)

  def step(self,mcs):
    # Iterate the model (run it for the time step specified on the load command)
    self.timeStepSBML()

    # Get the parameter value or molecular concentration from a cell
    for cell in self.cellList:
       <var> = self.getSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _cell=cell)

       # Set a new parameter value or molecular concentration value for a cell
       self.setSBMLValue(_modelName=<Name>,_valueName=<param/molecule>,_value=<val>,_cell=cell)

    # Change integration step of a cell type (overwrites the initial step value)
    self.setStepSizeForCellTypes(_modelName=<Name>, _types=[self.TYPE1,…], _stepSize=<newStep>)

    # Copy a SBML model from one cell to another
    self.copySBMLs(_fromCell=cell1, _toCell=cell2, _sbmlNames=[<ModelName1>,<ModelName2>,…])

    # Delete a SBML model from a cell type (all cells of that type will loose the model)
    self.deleteSBMLFromCellTypes(_modelName=<ModelName>, _types=[self.TYPE1,…])
```

Integration with CC3D – outside cells

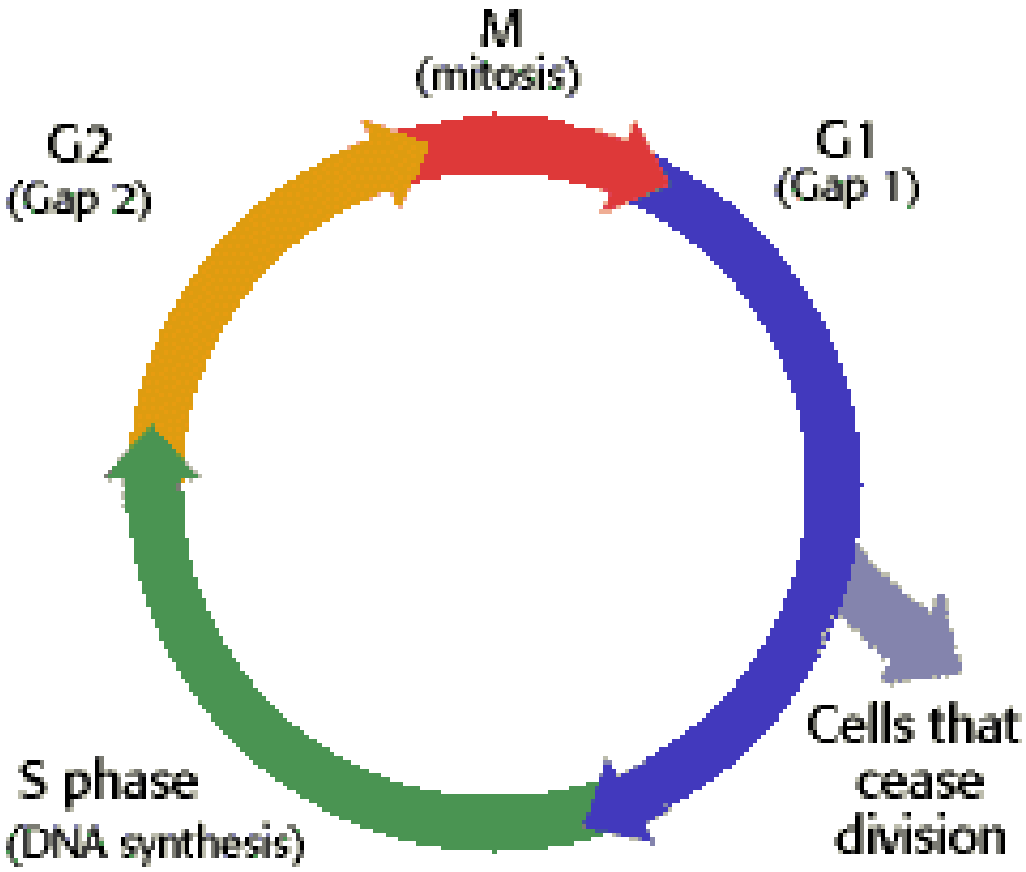
```
class <someClass>(SteppableBasePy):
  def __init__(self,_simulator,_frequency):
    SteppableBasePy.__init__(self,_simulator,_frequency)

  def start(self):
    # Define a SBML model
    Path = <ModelPath>   # Path where the model is stored
    Name = <ModelName>   # Name of the model
    Step = <timeStep>    # Time step of integration

    # Add a free floating SBML
    self.addFreeFloatingSBML(_modelFile=Path, _modelName=Name, _stepSize=Step)

  def step(self,mcs):
    # Iterate the model (run it for the time step specified on the load command)
    self.timeStepSBML()

    # Get the parameter value or molecular concentration from a free floating SBML
    <var> = self.getSBMLValue(_modelName=<Name>, _valueName=<param/molecule>)

    # Set a new parameter value or molecular concentration value for a free floating SBML
    self.setSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _value=<val>)

    # Change integration step of a free floating SBML (overwrites the initial step value)
    self.setStepSizeForFreeFloatingSBML(_modelName=<Name>, _stepSize=<newStep>)

    # Delete a SBML model from a free floating SBML
    self.deleteFreeFloatingSBML(_modelName=<ModelName>)
```

Integration with CC3D

- Other commands:

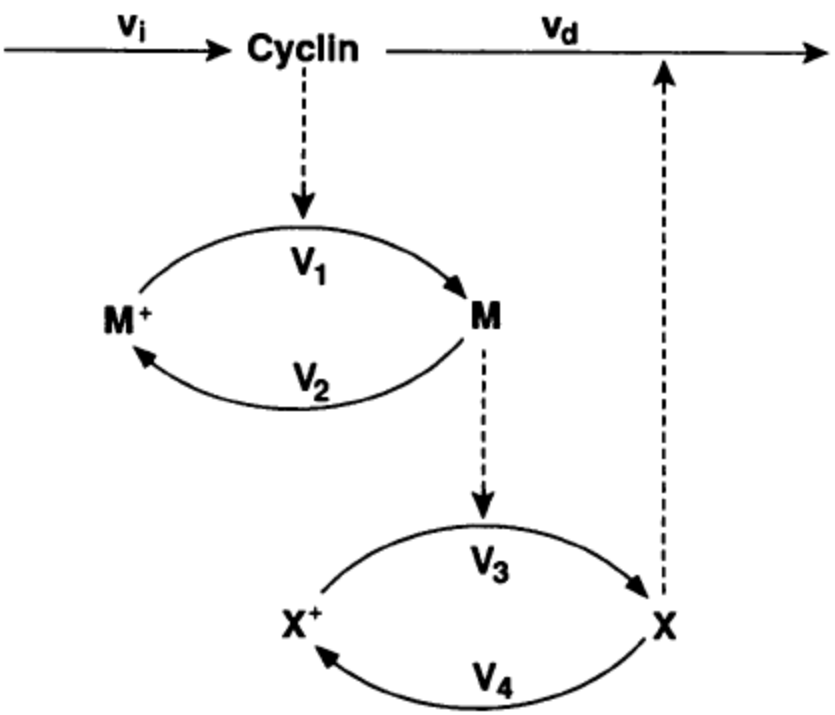
```
self.timestepSBML() # time step all models

self.timestepCellSBML() # time step all models associated with cells

self.timestepFreeFloatingSBML() # time step all free floating models

# check if a model is loaded into a cell
self.getSBMLSimulator(_modelName='',_cell=)

# check if a free floating model is defined
self.getSBMLSimulator(_modelName='')

# returns dictionary with all parameters and concentration values
state = self.getSBMLState(_modelName='',_cell=)

     state = self.getSBMLState(_modelName='myModel',_cell=cell)
     state >> {'S1':1.0, 'S2':0.0, 'k':0.1}

# sets new values for all parameters and concentrations defined in the dictionary
self.setSBMLState(_modelName='',_cell=,_state={})

     newState = {'S1':2.0, 'S2':0.5}
     self.setSBMLState(_modelName='myModel',_cell=cell,_state=newState)
```

Cell Cycle


M
(mitosis)
G1
(Gap 1)
Cells that
cease
division
S phase
(DNA synthesis)
G2
(Gap 2)

Cell Cycle

- Cyclin is produced at constant rate
- Cyclin activates cdc2 (M^+ -> M)
- Cdc2 activates protease X
- Protease degrades Cyclin



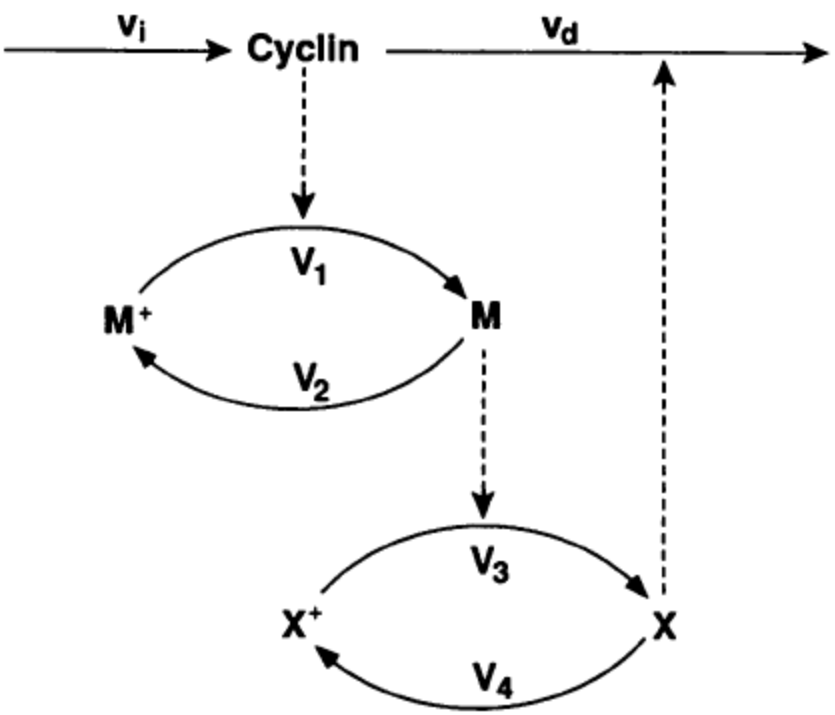
Cell Cycle

$$\frac{dC}{dt} = v_i - v_d X \frac{C}{K_d + C} - k_d C,$$

$$\frac{dM}{dt} = V_1 \frac{(1 - M)}{K_1 + (1 - M)} - V_2 \frac{M}{K_2 + M},$$

$$\frac{dX}{dt} = V_3 \frac{(1 - X)}{K_3 + (1 - X)} - V_4 \frac{X}{K_4 + X}$$

$$V_1 = \frac{C}{K_c + C} V_{M1},\ V_3 = M V_{M3}.$$



- C : cyclin concentration
- M : fraction of active cdc2 kinase
- X : fraction of active cyclin protease

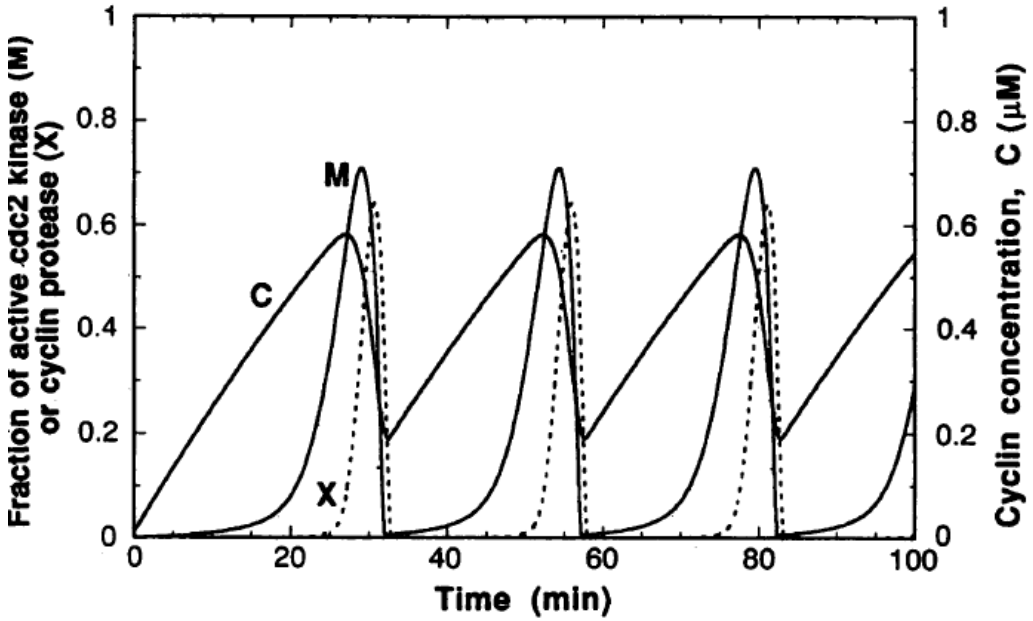
Cell Cycle

$$\frac{dC}{dt} = v_i - v_d X \frac{C}{K_d + C} - k_d C,$$

$$\frac{dM}{dt} = V_1 \frac{(1-M)}{K_1 + (1-M)} - V_2 \frac{M}{K_2 + M},$$

$$\frac{dX}{dt} = V_3 \frac{(1-X)}{K_3 + (1-X)} - V_4 \frac{X}{K_4 + X}$$

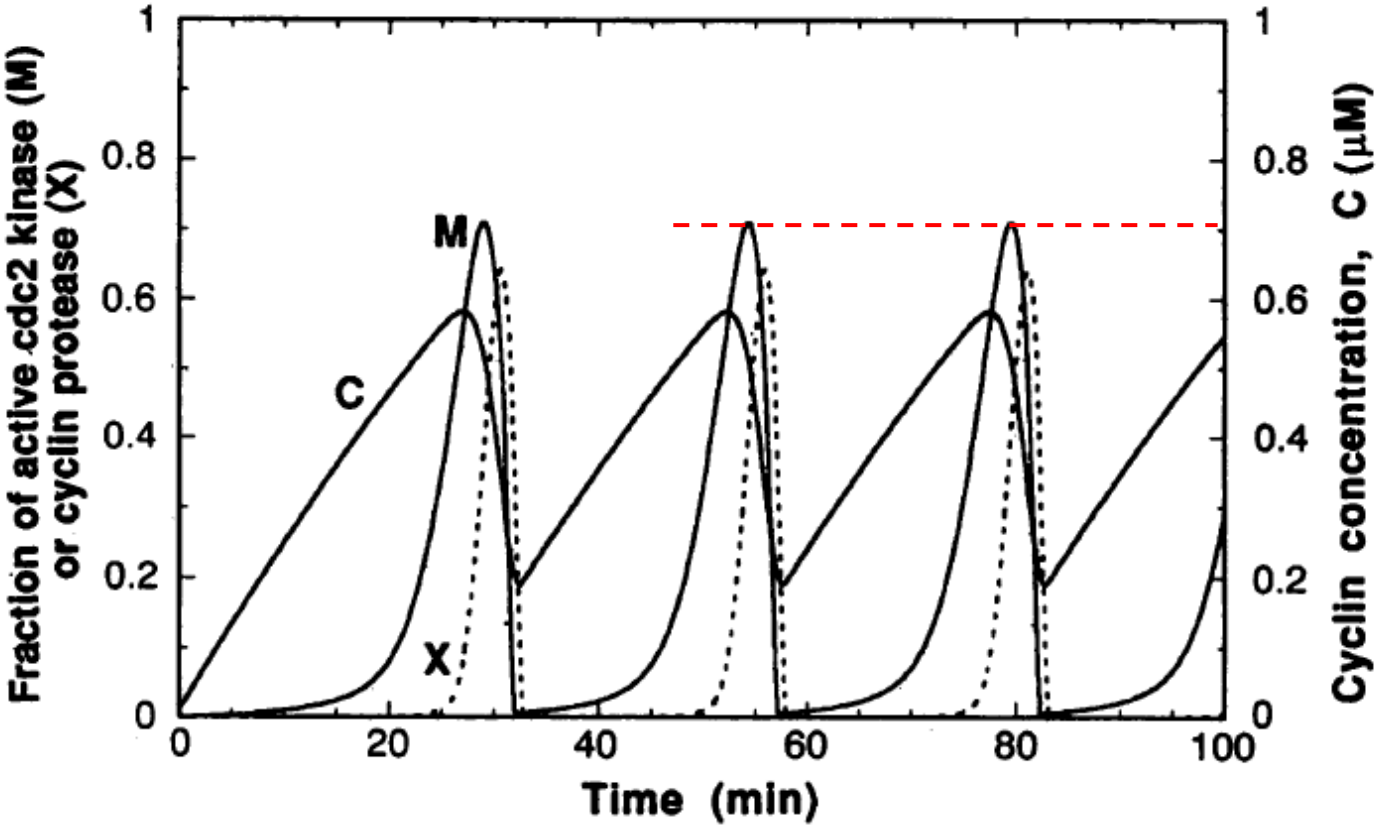
$$V_1 = \frac{C}{K_c + C} V_{M1},\ V_3 = M V_{M3}.$$



- C : cyclin concentration
- M : fraction of active cdc2 kinase
- X : fraction of active cyclin protease

Cell Cycle

- Mitosis occur when fraction of active Cdc2 kinase (M) reaches 0.7.



Cell Cycle