M Systems – simulator architecture and practical examples

Petr Sosík¹, Vladimír Smolka¹, Jan Drastik¹, Jaroslav Bradík¹, Max Garzon²

¹ INSTITUTE OF COMPUTER SCIENCE, FACULTY OF PHILOSOPHY AND SCIENCE, SILESIAN UNIVERSITY IN OPAVA, CZECH REP.

² THE UNIVERSITY OF MEMPHIS, TENNESSEE, USA
Motivation

• Create a computational model with focus on basic morphogenetic phenomena such as:
  • Growth
  • Homeostasis
  • Self-reproduction
  • Self-healing

• Simulate morphogenesis from scratch
  • Not to use atomic assembly units (cells)
  • Start from 1D/2D/3D primitives
  • Use self-assembly feature to create 3D cell-like forms
M systems – formal definition
Morphogenetic systems (M systems)

- Based on principles of common membrane computing models, especially with proteins on membranes
- Live in a 3D space (generally 3D)
- Introduce explicit geometric features and self-assembly capabilities
  - Each elementary object has a fixed shape and position in space at any point in time
- Exhibit emergent behavior from local interactions
- Informed by tile assembly models
  - Polytopes and connectors like tiles and glues
- Use 3 types of objects:
  - Floating objects
  - Tiles
  - Protons (abstraction of biological "proteins")
Basic M system objects

- Floating objects
  - Small shapeless atomic objects floating freely within the environment
  - With a nonzero volume and specific position

- Tiles
  - Have their predefined size and shape (convex bounded polytopes)
  - Can stick together along their edges or at selected points called connectors
  - Can self-assembly into interconnected structures

- Protions
  - Are placed on tiles
  - Catalyze reactions of floating objects
  - Serve as "protion channels" through \((d-1)D\) tiles

```xml
<floatingObjects>
  <floatingObject name="a">
    <shape value="sphere"/>
    <size value="0.05"/>
    <color name="Lime" alpha="255"/>
    <mobility value="15"/>
    <concentration value="0.1"/>
  </floatingObject>
</floatingObjects>

<tile name="q0">
  <polygon>...</polygon>
  <positions>...</positions>
  <connectingAngle value="2.034443935795703" unit="rad"/>
  <connectors>...</connectors>
  <surfaceGlue name="gx"/>
  <color name="DeepSkyBlue" alpha="64"/>
</tile>

<proteins>
  <protein name="p0"/>
  <protein name="p1"/>
  <protein name="p2"/>
  <protein name="p3"/>
  <protein name="p4"/>
</proteins>
```
Formal definition

- **M**orphogenetic system \( M = (F, P, T, \mu, R, \sigma) \)
  - \( F \) - the catalog of floating objects
  - \( P \) - the set of protions
  - \( T \) - is a polytopic tile system
  - \( \mu \) - maps proteins to positions on M-tiles
  - \( R \) - is a set of reaction rules
  - \( \sigma \) - maps glue pairs to a multiset of floating objects produced when the binding is established
  - concentration of each object in the environment
  - mean mobility of each floating object
  - radius of interaction of each floating object
  - \( e \) - concentration of each object in the environment
  - \( p \) - is the set of protions
  - maps proteins to positions on M-tiles
Formal definition

```xml
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="http://www.example.com"
>
  <tiling>
    <tiles>...</tiles>
    <glues>...</glues>
    <glueRelations>...</glueRelations>
    <initialObjects>...</initialObjects>
    <glueRadius value="0.1"/>
  </tiling>
  <Msystem>
    <floatingObjects>...</floatingObjects>
    <proteins>...</proteins>
    <proteinsOnTiles>...</proteinsOnTiles>
    <evoRulesWithPriority>...</evoRulesWithPriority>
    <signalObjects>...</signalObjects>
    <reactionRadius value="14"/>
  </Msystem>
</root>
```
Reaction rules

• Are used for reactions and modifications of the M system during growth

• Four types of reaction rules:
  • Metabolic rules
  • Creation rules
  • Destruction rules
  • Division rules
**Metabolic rules**

- A multiset of floating objects reacts and changes, or it is transported through a \((d-1)\)D tile.
- \((d-1)\)-dimensional tiles have their sides marked “in” and “out”, by convention.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>RULE</th>
<th>EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMPLE</td>
<td>(u \rightarrow v)</td>
<td>objects in multiset (u) react to produce (v)</td>
</tr>
<tr>
<td>CATALYTIC</td>
<td>(pu \rightarrow pv) (u[p \rightarrow v[p) ([pu \rightarrow [pv)</td>
<td>objects in (u) react in presence of (p) to produce (v); this variant requires both (u, v) at the side “out” of the tile; this variant requires both (u, v) at the side “in” of the tile;</td>
</tr>
<tr>
<td>SYMPORT</td>
<td>(u[p \rightarrow [pu) ([pu \rightarrow u[p)</td>
<td>passing objects in (u) through protonation channel (p) to the other side of the tile</td>
</tr>
<tr>
<td>ANTIPORT</td>
<td>(u[pv \rightarrow v[pu)</td>
<td>interchange of (u) and (v) through protonation channel (p)</td>
</tr>
</tbody>
</table>
Creation rules

- Creates tile \( t \) while consuming the floating object in \( u \)
- Rule format: \( u \rightarrow v \)

```xml
<evoRule type="Create">
  <leftside value="a,a,a,a,a,a,a,a"/>
  <rightside value="q2"/>
</evoRule>
```
Destruction rules

• Tile $t$ is destroyed in the presence of multiset of floating objects $u$ which is consumed
• All connections from $t$ to other tiles are released
• Rule format: $ut \rightarrow v$

```xml
<evoRule type="Destroy">
  <leftside value="a,q1"/>
  <rightside value="b"/>
</evoRule>
```
Division rules

• Two connectors with glues \( g, h \) get disconnected and the multiset \( x \) of floating objects is consumed

• Rule format: \( g \xrightarrow{x} h \rightarrow g, h \)

\[
\begin{align*}
\text{<evoRule type="Divide">} \\
\text{<leftside value="gdd, x, gdd"/>} \\
\text{<rightside value="gdd, gdd"/>} \\
\text{</evoRule>}
\end{align*}
\]
M system computation

- Initial configuration contains only seed tiles in S and random distribution of floating objects with concentration $\varepsilon$

- Computation takes place in discrete steps

- During each step, rules from R are applied in maximally parallel manner
  - Applicable rules are chosen randomly until no further rule is applicable

- Rules are applied in parallel to the actual configuration

- Each floating object changes its position randomly within its mobility perimeter

- A sequence of transitions between configurations is called a computation (nondeterministic)

- A computation ends when there is no longer any applicable rule
M systems simulator
- architecture
M systems simulator - architecture

- Modular architecture with strong OOP approach
- Separated simulation engine (standalone DLL) with user friendly API
- M system is defined as XML file (available validation against our XSD)
  - XML is also possible to create using our M System creator tool
- Using Unity Game Engine for visualization of simulated data
M systems simulator – dependency tree

- SharedComponents
- MSysytemCreator
- MSysytemSimulationEngine
- Cytos_v2
- Numerics
- Spatial
M systems simulator – modules

- **Cytos_v2**
  - Main application for running simulator
  - Contains only UI all computation is done by MSystemSimulationEngine

- **MSystemSimulationEngine**
  - Main ENGINE of whole project
  - Contains deserialization tool (usage is optional)
  - Simulation methods for simulation run
  - Serialize output data into XML structure called snapshot file
    - Contains objects (created/destroyed/moved) with positions and color informations
M systems simulator – modules

• MSystemCreator
  • Easy to use XML creator and editor
  • Creates basic XML structures
  • Contains validation features and help

• SharedComponents
  • Basic library which contains useful and shareable tools across solution

• Spatial and Numerics
  • Open source mathematical libraries (user for eg. object shifts)

• MSystemVisualization
  • Unity game engine used for visualization
M systems simulator – input/output files

• Input
  • Created M system is saved to M system description XML
    • Structure is defined in XSD
  • Deserialized description constructs all expected simulation objects
  • These objects are used for simulation

• Output
  • So-called Snapshot file
    • Also in XML structure
  • Contains visualization objects with its shape, in specified position and with defined color
  • Can be used as input file for MSystemVisualization (Unity engine)
M systems simulator – technical details

• Written in .NET 4.5.2
• Using TFS as version control
• Strong OOP approach
• Development is under SCRUM principles
• Using unit and integration testing across whole solution
M systems simulator - examples
2D square tiling

• Basic square tile (d)
  • 4 sides
  • 4 connectors
    • c1: v1 – v2, glue g1
    • c2: v2 – v3, glue g2
    • c3: v3 – v4, glue g3
    • c4: v4 – v1, glue g4
  • Angle 180°
  • Glue relations: g3 – g1, g4 – g2
  • Contains high concentration of floating objects „a“
  • Initial object (tile): d
  • Only one rule: Create (a->d)
Boxy hallows

• Harry Potter and the Deathly Hallows part 2 - inside Belatrix's vault
  • Duplication spell

• Basic square tile (d)
  • 4 sides
  • 4 connectors
    • c1: v1 - v2, glue g1
    • c2: v2 - v3, glue g1
    • c3: v3 - v4, glue g1
    • c4: v4 - v1, glue g1
  • Angle 90°
  • Glue relations: g1 – g1
  • Contains high concentration of floating objects „a“

• Initial object (tile): d
• Rules: Create (a->d), Divide (g1 -> g1, g1)
Cell growth

- Large system of interconnected tiles
- Contains large pentagonal tiles (cytoskeleton), small pentagonal tiles (core) and interconnecting rods
- Cell growth behaviour is simulated by Metabolic/Division/Creation rules
- Initial object is one large pentagonal tile
Circuit Tile Assembly Models (cTAM)
Circuit Theory Background

- Voltage is the electric potential energy between two nodes in a circuit.
- The ground node in the circuit has a potential of zero.
- Voltages relative to ground are associated with each node in the circuit.
- It is analogous to pressure in a water pipe:
  - Pressure causes water to flow.
  - Voltage causes current to flow.
- Voltage is necessary to move a positive charge against an electric field.
- Voltage gives rise to current in a circuit.
- Electrical resistance is analogous to the resistance produced by different diameter pipes in a water system.
- Resistors are units of dielectric material that impede current flow.
Circuit Tile Assembly Models (cTAM)

- new self-assembly analog model
- motivated by Tile Assembly Models (aTAM)
- DC resistive circuits self-assemble under voltage control
- This model exhibits both self-assembly and self-control
- attachments are controlled by local voltage differences as the assembly grows
- resource can be thought of as a source of energy for growth reaction

cTAM – formal definition

A circuit tile assembly system is a tuple $C = (\Gamma, S, \tau, v, \zeta)$

- $\Gamma$ – finite set of circuit tile types
- $S = S \subseteq \Gamma$ set of seed circuit tiles that includes a source and ground
- $\tau = \tau \in \mathbb{R}$ is the threshold voltage for attachment
- $v = N \rightarrow \mathbb{R}$ is the electric potential energy at a node relative to ground
- $\zeta = N_{in} \rightarrow N_{out}$ maps input nodes to output nodes
Circuit Tile Assembly Models (cTAM)

1. Seed circuit tile for all assemblies

2. Circuit tile for the ladder circuit.
The circuit has $R$ in series with $R^eq(m)$. This series combination is then parallel with $R$ resulting in $R^eq(m) = \frac{R(R + R^eq(m-1))}{2R + R^eq(m-1)}$. Solving the quadratic equation and knowing that resistance cannot be negative.

Recurrent relationshit calculate voltage at the cTAM connecting new tiles as long as the output voltage exceeds the threshold limit.
3. Ladder circuit assembly with $V_0 = 50$ and $\tau = 1$. The maximum size is $n = 5$
cTAM and M system

- M system simulator is extended with electric cTAM mode
- cTAM can be seen as a special case of M system with dynamical glues
- Now 1D simulate only. 2D simulate in the Spring 2019
cTAM computation

- During each step we take random tile
- Calculate threshold, for connection of a new tile
- A computation ends when output energy of all end tiles is lower than the threshold

Example
- Two different types of tiles (A, B)
- Voltage: 100
- Alpha ratio: A – 1, B - 2
- Threshold: 1
cTAM example
cTAM example
cTAM example
cTAM example (Vo=100, thP = 0.1)
cTAM example ($Vo=100$, $thP = 0.1$)
Unity Game Engine

- Developer is the Unity Technologies
- is a multi-platform game engine (Windows, Mac, ...)
- unity supports two and three-dimensional environment
- supports natively C# and UnityScript
- supports building to 27 different platforms
Unity Game Engine
Unity Game Engine

Monument Valley 2
Visualization

- Input data, we get from our simulator (Snapshot file – XML),
- Input file deserialize to new steps
- For tile, we use method Mesh drawing
- In visualization, we do not show floating objects
- For easy control in environment, user can use spectator mode, forward and backwards stepping
- Output data – complete visualization (Example $M_0$)
Example: Tile drawing

<tile name="s" objectID="2340" type="tile" state="Create">
  <vertices>
    <vertex>
      <posX value="0" />
      <posY value="10" />
      <posZ value="0" />
    </vertex>
    ....
    </vertices>
    <color name="40ff1493" />
  </tile>
SnapshotStep total: 42
Actual step: 2
Thank you, any questions?

For more information and free download of the M system simulator and the visualization engine please consult Morphogenetic systems download page:

http://sosik.zam.slu.cz/msystem/

or use QR code