A Simulation Framework for Pervasive Service Ecosystems

Danilo Pianini
danilo.pianini@unibo.it

Alma Mater Studiorum—Università di Bologna

July 6, 2011
1. SAPERE background
   - SAPERE
   - The role of simulation in SAPERE
   - Possible simulation approaches

2. Alchemist
   - Taking the best of both approaches
   - Computational model
   - Dynamic Engine
   - Alchemist Architecture

3. Case study
   - A Crowd evacuation scenario
   - Simulation results
The SAPERE Project

SAPERE (Self-aware Pervasive Service Ecosystems) is an EU STREP project under the FP7 FET Proactive Initiative: Self-Awareness in Autonomic Systems (AWARENESS).

The objective of SAPERE is the development of a highly-innovative theoretical and practical framework for the decentralized deployment and execution of self-aware and adaptive services for future and emerging pervasive network scenarios.
Simulation of a SAPERE environment

The role of simulation

- No way to assert the emergent properties of the system
- System must be resilient to changes: it’s hard to plan test the real system in a whole lot of different situations
- It’s impossible to deploy the whole ecosystem without any warranty

First class abstractions

- High dynamic environment composed of different, mobile, communicating nodes
- Autonomous agents
- Programmability through a set of chemical-like laws, working with CTMC model
Two approaches

Classic ABM modelling
- High flexibility
- Topology as first-class abstraction
- Dynamics explicitly modelled
- No native support for CTMC model

Chemical inspired modelling
- Natively CTMC
- Monstrously fast engines for stochastic simulation
- Very limited topology: multicompartment as best
- No other things than reactions: extremely limited flexibility
Alchemist simulation approach

Base idea

- Start from the existing work with stochastic chemical systems simulation
- Extend it as far as needed to reach desired flexibility

SSA

- Continuous Time Markov Chain (CTMC)
- Stochastic and precise
- Complete solution unfeasible for non-trivial problems
- Approximated solution through series of Monte Carlo simulations
- Several efficient algorithms
Extending the concept of reaction

Chemical reaction
- A set of reactants combine themselves in a set of products
- The speed of the reactions is function of:
  - the concentration of the reactants
  - a kinetic constant

Alchemist reaction
- A set of conditions allows the execution of a set of actions
  - A condition associates to each state of the system a boolean value
  - An actions describes a modification of the environment status
- The speed of the reactions is function of the current environment status (rate equation)
Enriching the environment description

Alchemist world

- Environment which contains Nodes
- Each Node is programmed with a set of Reactions
- Nodes contain Molecules
- Each Molecule in a node is described with a Concentration
Making efficient SSA Algorithms more flexible

Existing SSA algorithms

- Several versions, but same base schema [Gillespie, 1977]:
  1. Select next reaction to execute according to the markovian rates
  2. Execute it
  3. Update the markovian rates which may have changed

- Very efficient versions exist: logarithmic [Gibson and Bruck, 2000] and constant [Slepoy et al., 2008] time scaling with reactions number

What they miss is what we added

- Reactions can be added and removed during the simulation
- Support for non-exponential time distributed events (e.g. triggers)
- Dependencies among reactions are evaluated considering their “context”, speeding up the update phase
Smart data structures ensure bleeding edge performances

Next Reaction efficient structures made dynamic

- **Dynamic Indexed Priority Queue**
  - Allow to access the next reaction to execute in \(O(1)\) time
  - Worst case update in \(\log_2(N)\) (average case a lot better)
  - Extended to ensure balancing with insertion and removal

- **Dynamic Dependency Graph**
  - Allows to smartly update only a subset of all the reaction
  - Extended with the concept of input and output context
Alchemist is fully modular

- Core Engine
  - Reaction Manager
  - Simulation Flow
  - Dependency Graph
- Reporting System
- Interactive UI
- Environment
- Language Parser
  - Environment Instantiator
- XML Bytecode
- Application-specific Alchemist Bytecode Compiler
- Environment description in application-specific language
Run away from the burning hall!

**Scenario description**

- A big hall containing a lot of people and a sensor network
- Two exit doors, reachable through a corridor
- A fire sets up in a point
- Everybody must be guided to safeness, avoiding crowded paths
Eco-laws for this scenario

\[
\begin{align*}
\langle \text{source}, T, M, A \rangle & \xrightarrow{R_{\text{init}}} \langle \text{source}, T, M, A \rangle, \langle \text{grad}, T, 0, M, A \rangle \\
\langle \text{grad}, T, V, M, A \rangle & \xrightarrow{R_{\text{s}}} \langle \text{grad}, T, V, M, A \rangle, \langle \text{grad}, T, \text{min}(V+\#D, M), M, A \rangle \\
\langle \text{grad}, T, V, M, A \rangle, \langle \text{grad}, T, W, M, A \rangle & \rightarrow \langle \text{grad}, T, \text{min}(V, W), M, A \rangle \\
\langle \text{grad}, T, V, M, A \rangle & \xrightarrow{R_{\text{ann}}(A)} \langle \text{grad}, T, V+1, M, A \rangle \\
\langle \text{info}, \text{attr}, A, TS \rangle, \langle \text{info}, \text{attr}, A2, TS+T \rangle & \rightarrow \langle \text{info}, \text{attr}, A2, TS+T \rangle \\
\langle \text{grad}, \text{exit}, E, Me, Ae \rangle, & \xrightarrow{R_{\text{att}}} \langle \text{grad}, \text{exit}, E, Me, Ae \rangle, \langle \text{grad}, \text{fire}, F, Mf, Af \rangle, \\
\langle \text{grad}, \text{fire}, F, Mf, Af \rangle, & \langle \text{info}, \text{crowd}, CR, TS \rangle, \\
\langle \text{info}, \text{crowd}, CR, TS \rangle & \xrightarrow{R_{\text{disp}}(\Delta)} \langle \text{info}, \text{escape}, \#O \rangle, \langle \text{info}, \text{attr}, A, TS \rangle, \\
\langle \text{info}, \text{escape}, L \rangle, \langle \text{info}, \text{attr}, A, TS \rangle, & \xrightarrow{R_{\text{disp}}(\Delta)} \langle \text{info}, \text{escape}, \#O \rangle, \langle \text{info}, \text{attr}, A, TS \rangle, \\
+ \langle \text{info}, \text{attr}, A+\Delta, TS2 \rangle & + \langle \text{info}, \text{attr}, A+\Delta, TS2 \rangle
\end{align*}
\]
Follow the gradient
I have some nice videos

- A SAPERE-oriented scenario: the crowd evacuation
- A biology-oriented application: simple morphogenesis

See you at the demo session

- I’ll let Alchemist run, showing two more examples
  - Falling back to classic ABM: synchronizing fireflies
  - Crowd steering in a slightly more complex scenario
Feel free to ask :)


A Simulation Framework for Pervasive Service Ecosystems

Danilo Pianini
danilo.pianini@unibo.it

Alma Mater Studiorum—Università di Bologna

July 6, 2011