Universität Rostock Traditio et Innovatio

One Model – Two Approaches

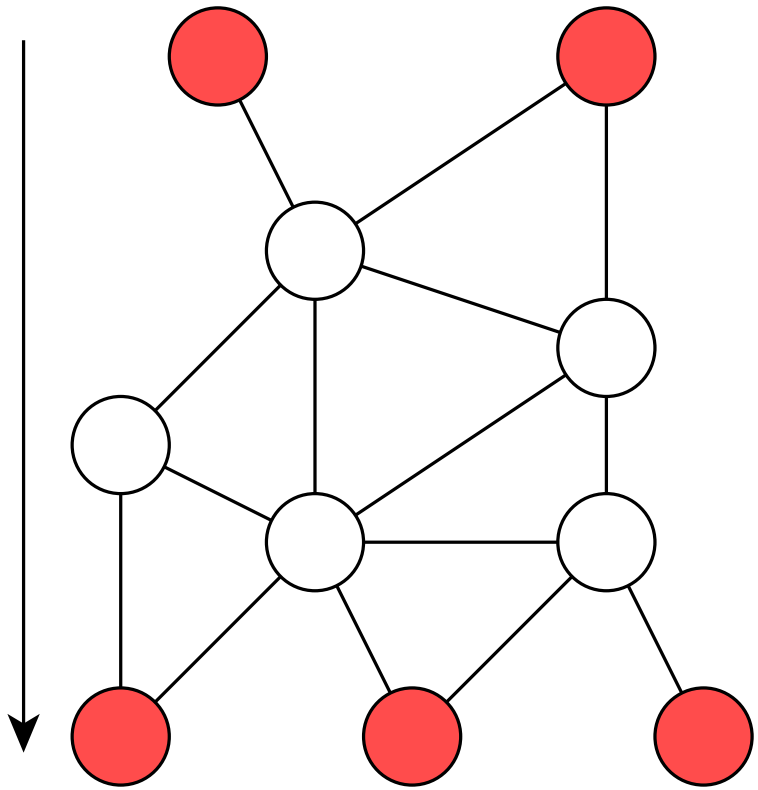
Lessons Learned from Implementing a Model Twice

Oliver Reinhardt [1] **Martin Hinsch** [2] **Jakub Bijak** [2]
Adelinde M. Uhrmacher [1]

[1] University of Rostock
Institute of Computer Science

[2] University of Southampton
Social Statistics and Demography

A Model of Routes and Rumors (Model World)



Entries

Cities

resources $\in [0, 1]$

control $\in [0, 1]$

Transport Links

friction $\in [0, 1]$

distance $\in [0, 1]$

Exits

A Model of Routes and Rumors (Migrant Behavior)

Agents start at an **entry**, and try to find the way to any **exit**, acting on **limited information**.

Information: (subject, value, trust)

Behavior options:

- Explore the current cities
- Make contacts
- Exchange information with contacts
- Move to a neighboring cities

Two Approaches

Approach 1: Julia

- Modern (somewhat) general purpose language
- Designed for scientific computing

Approach 2: ML3

- Domain-specific modeling language for agent-based demographic models
- Agents interacting in dynamic networks
- Stochastic rules in continuous time

Implementation 1: Julia

Discrete time steps.

At each time steps migrants might ...

- Stay to explore, make contacts
- Move to a neighboring city

... in addition to always exchanging information

Implementation 2: ML3

Competing risk in continuous time.

All behavior options are “processes” that happen in parallel. Decision between them stochastically via the rate.

```
Migrant
| ego.capital >= 0 && !ego.in_transit
@ ego.move_rate()
-> ego.in_transit := true
   ego.destination := ego.decide_destination()
   ego.capital -= ego.move_cost(ego.destination)
```

A Comparison

Separation of Model and Simulation Logic

Why: Clearer model logic and reusable simulation logic.

Julia: no separation

```
function step_agent!(agent, model, par)
  if decide_stay(agent, par) # model logic
    step_agent_stay!(agent, model.world, par)
  else
    step_agent_move!(agent, model.world, par)
  end
    step_agent_info!(agent, model, par)
  ...
end
```

ML3: complete separation

A Comparison

Handling Knowledge

Why: It is difficult.

Very similar in both implementations.

```
Information(
  quality : real,
  resources : real,
  quality_trust : real,
  resources_trust : real
)

subject:Location[1]<->[n]Information:information
knowledge:Information[n]<->[1]Migrant:owner

Migrant.knowledge_about(?loc : Location) : Information :=
  ego.knowledge.filter(alter.subject = ?loc).only()
```

A Comparison

Simulation Infrastructure

Why: We need to experiment with the model.

Julia:

- Ad-hoc solutions for observation, parameterization, parallelization of runs, storing results, …

ML3:

- SESSL binding (a DSL for experiment specification)

A Comparison

Simulation Infrastructure

SESSL:

```
new Experiment with Observation with ParallelExecution with CSVOutput {
model = "routes_rumors.ml3"
simulator = NextReactionMethod()
parallelThreads = -1
stopTime = 100

initializeWith(new RoutesRumorsInit())

scan("speed_expl_stay" <~ range(0,0.1,1), "p_info_contacts" <~ range(0,0.1,1))

observeAt(stopTime)
observe("exiting" ~ expressionDistribution(agentType = "Location", filter =
      "ego.type = 'exit'", expression = "ego.migrants.size()"))

withExperimentResult(writeCSV)
}
```

A Comparison

Language Infrastructure

Why: It makes life easier.

Julia:

- IDE
- Ample Documentation
- Community of Users
- Debugger
- REPL

ML3:

- rudimentary Editor
- Some papers with models
- If there are problems, ask me

Lessons Learned

- Using a DSML is worth it, if it suits the needs of the model.
- The rule-based approach works well for external events and continuous behavior, less so active decision making.
- While ML3 is already very expressive (compared to DSMLs for other domains), it is not expressive enough (data structures, attributed links).

→ An (internal) DSML for modeling decision making with limited information?