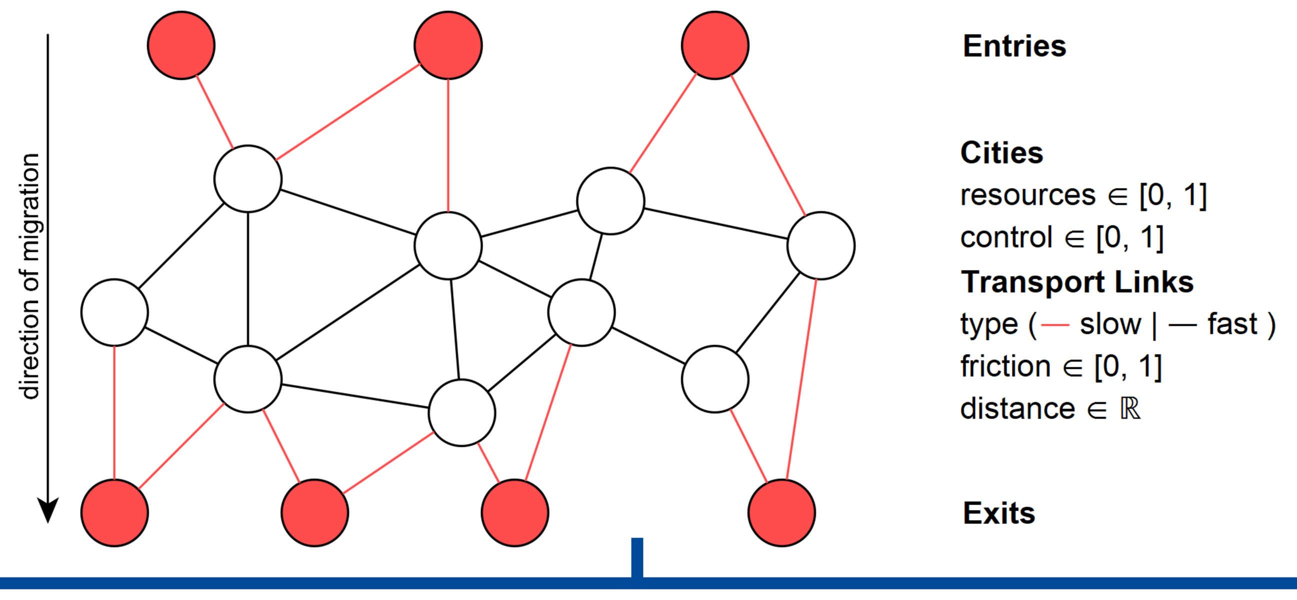
Developing Agent-Based Migration Models In Pairs

Approach:
- Committing too early to a specific model architecture, design, or language environment can later become costly.
- To reveal crucial features and trade-offs for future implementation and for the design of modeling formalisms, we have implemented two instances of the same model using different approaches.



Abstract Model:
- Formation of migration routes with limited information about the environment.
- Migrants enter the graph world on one side, and try to find a path to an exit.
- Migrants must decide to: explore, make contacts, exchange knowledge, move to a neighboring location.

Simulation Model 1: Julia

A modern general purpose language (GPL)
- Discrete time steps: iteration through all agents every time step to execute their behavior.
- Modeled processes happen every step, or have fixed probabilities.

```
# model logic:
function step_agent_move!(agent, world, par)
  agent.in_transit = true
  loc = next_step(agent, world, par)
  link = find_link(agent.loc, loc)
  link.count += 1 # update traffic counter
  costs_move!(agent, link, par)
  explore_move!(agent, world, loc, par)
  move!(world, agent, loc)
end

# corresponding scheduling logic:
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)
  agent.steps += 1 # required for analysis
end
```

Simulation Mode 2: ML3

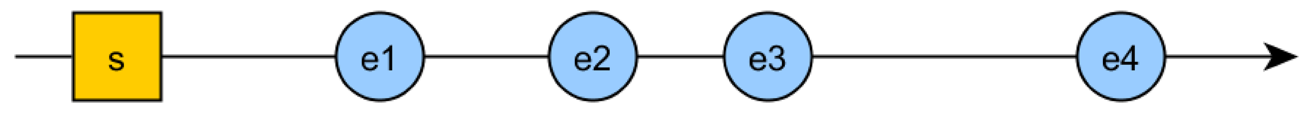
An external domain-specific language (DSL) for agent-based modeling
- Behavior modeled using stochastic rules (*who-when-what*-triplets) in continuous time.
- Decision-making based on "competing risk" approach: options with higher rates are chosen with higher probability.

```
Migrant                           # who: affected agent type
|  !ego.in_transit                #      and guard condition
@  ego.move_rate()                # when: exponential rate
-> ego.in_transit := true         # what: effect
   ego.destination := ego.decide_destination()
   ego.capital -= ego.move_cost(ego.destination)
```

Scheduling: Continuous-time Markov Chain semantics

1) System is in state *s* at some time *t* :

2) Waiting times are drawn for all events (pairs of agent and applicable rule) from an exponential distribution parameterized with the rate:



3) The earliest event is executed, yielding a new state *s*', all others are discarded:

	Julia		ML3	
✗	Model and execution are tightly interwoven.	Separation of Model and Simulator	The declarative model description is executed by a separate simulator.	✓
✓	Due to interwoven model and simulation, additions make changes in many locations necessary.	Composability and Extendability	White box composition or extension by adding behavioral rules. Changes are typically localized in one or a few rules.	✓
✓	As general purpose language, Julia has a rich set of features allowing for maximum flexibility.	Flexibility	While ML3 is very expressive compared to DSLs in other areas, many features of GPLs (e.g., structures types, data structures) are not available.	✓
✓	Extensive ecosystem: IDEs, debugger, REPL, documentation, libraries, user community, ...	Language Infrastructure	Only rudimentary tool support. Few users. Documentation mostly in form of publications and existing models.	✓
✓	No specific support for simulation, but enough veritility to implement ad-hoc solutions. Ecosystem focused on scientific computing can be exploited.	Simulation Infrastructure	Simulation experimentation supported „out of the box" via a binding to SESSL, a DSL for defining and executing simulation experiments.	✓
✓	Language designed with efficient execution in mind. Ample versatility facilitates an efficient implementation. Ca. 10 s per run.	Runtime	Lack of language features (see flexibility) leads to inefficient implementation of knowledge representation and information exchange. Ca. 1.5 h per run.	✗

Conclusions:
- Applying a DSL such as ML3 has significant advantages. However, the limitations ML3 make its application impractical for this model.
- Extension of ML3 to make it applicable would require adding many GPL features. Continued development as an internal DSL might be appropriate.
- Knowledge representation and exchange proved to be challenging both conceptually and in implementation. DSL support would be advantageous.

O. Reinhardt, M. Hinsch, J. Bijak, A. M. Uhrmacher "Developing Agent-Based Migration Models in Pairs" Winter Simulation Conference 2019 (to appear)

Oliver Reinhardt, Adelinde M. Uhrmacher
VISUAL AND ANALYTIC COMPUTING
University of Rostock | Albert-Einstein-Str. 22 | 18055 Rostock | Germany

Martin Hinsch, Jakub Bijak
SOCIAL STATISTICS AND DEMOGRAPHY
University of Southampton | University Road | Southampton SO17 1BJ | UK