Probabilistic Relational Agent-Based Models

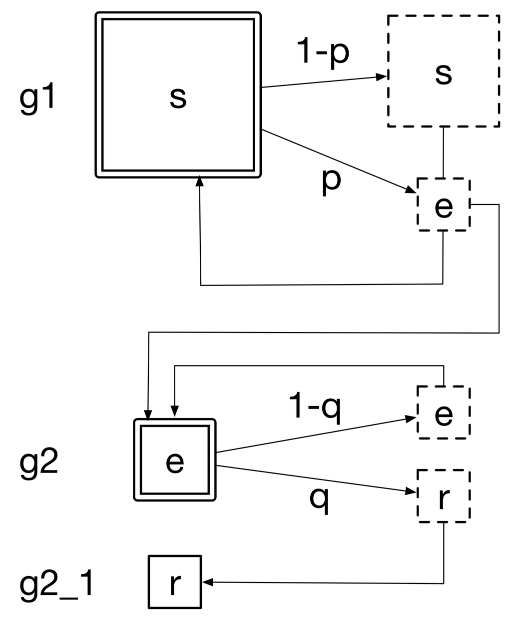
Tomasz D. Loboda
Paul R. Cohen
University of Pittsburgh

1. Motivation

- What
  - Accelerate modeling of all kinds
  - Target language for semi-automated construction of probabilistic relational agent-based models (PRAMs)
- How
  - By combining agent-based models with probability theory
  - Effectively arriving at a mass redistribution system

2. Elements of a PRAM

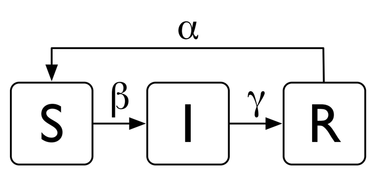
- Entities
  - Groups
    - Mass (e.g., 500 agents)
    - Attributes (e.g., age, sex, flu)
    - Relations (e.g., school, hospital)
  - Sites
    - Locations agents can be @
- Rules
  - Animate mass redistribution



3. Modeling Levels

- Domain — Invoke domain-specific models (e.g., SIRS)
- Class — Invoke a class of processes or models (e.g., MC)
- Rule — Write rules directly

- Example: The SIRS model
  - β - transmission rate
  - γ - recovery rate
  - α - immunity loss rate (α = 0 implies life-long immunity)



3.1. Domain Level (example in Python)

```
SIRSModel('flu', β=0.05, γ=0.50, α=0.10)
```

3.2. Class Level (example in Python)

```
β, γ, α = 0.05, 0.50, 0.10
transition_matrix = {
    's': [1 - β,      β,  0.00],
    'i': [ 0.00, 1 - γ,      γ],
    'r': [     α,  0.00, 1 - α]
}
TimeInvMarkovChain('flu', transition_matrix)
```

3.3. Rule Level (more elaborate example in pseudo-code)

```
rule_flu_progression():
    if group.feature.flu == 's':
        p_inf = n@_{feature.flu == 'i'} / n@    # n@ - count at the group's location
        dm      p_inf -> F:flu = 'i', F:mood = 'annoyed'
        nc  1 - p_inf
    if group.feature.flu == 'i':
        dm 0.2 -> F:flu = 'r', F:mood = 'happy'
        dm 0.5 -> F:flu = 'i', F:mood = 'bored'
        dm 0.3 -> F:flu = 'i', F:mood = 'annoyed'
    if group.feature.flu == 'r':
        dm 0.1 -> F:flu = 's'     # dm - distribute mass
        nc 0.9                    # nc - no change

rule_flu_location():
    if group.feature.flu == 'i':
        if group.feature.income == 'l':
            dm 0.1 -> R:@ = group.rel.home
            nc 0.9
        if group.feature.income == 'm':
            dm 0.6 -> R:@ = group.rel.home     # R:@ - location the group is at
            nc 0.4
    if group.feature.flu == 'r':
      dm 0.8 -> R:@ = group.rel.school
      nc 0.2
```

3.4. Rule Level (ODEs support in development; Python)

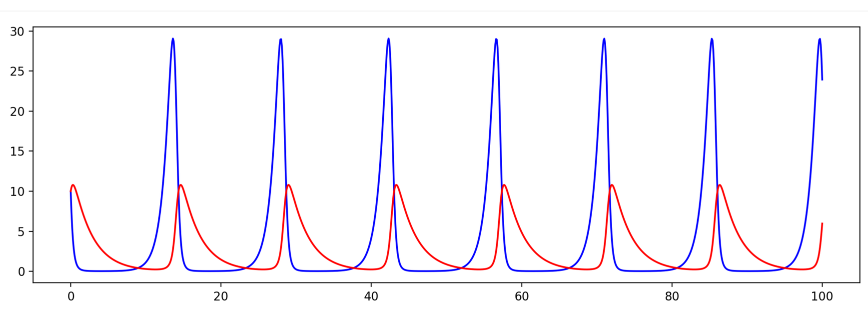
```
α = 1.1   # baboon
β = 0.4
δ = 0.1   # cheetah
γ = 0.4

def f_lotka_volterra(t, state):
    x,y = state
    return [x * (α - β*y), -y * (γ - δ*x)]

r = ODESystem(f_lotka_volterra, ['x', 'y'], dt=0.1)

Simulation().add([r, Group(n=1, attr={'x':10, 'y':10})]).run(1000))  # initial (10,10)

h = r.get_hist()  # time series of computed values
plt.plot(h[0], h[1][0], 'b-', h[0], h[1][1], 'r-')     # red-predator; blue-prey
```



4. Semi-Automated Construction

- Static and dynamic rule analysis
  - Identify essential group attributes and relations
- Automatic population generation from rel. databases

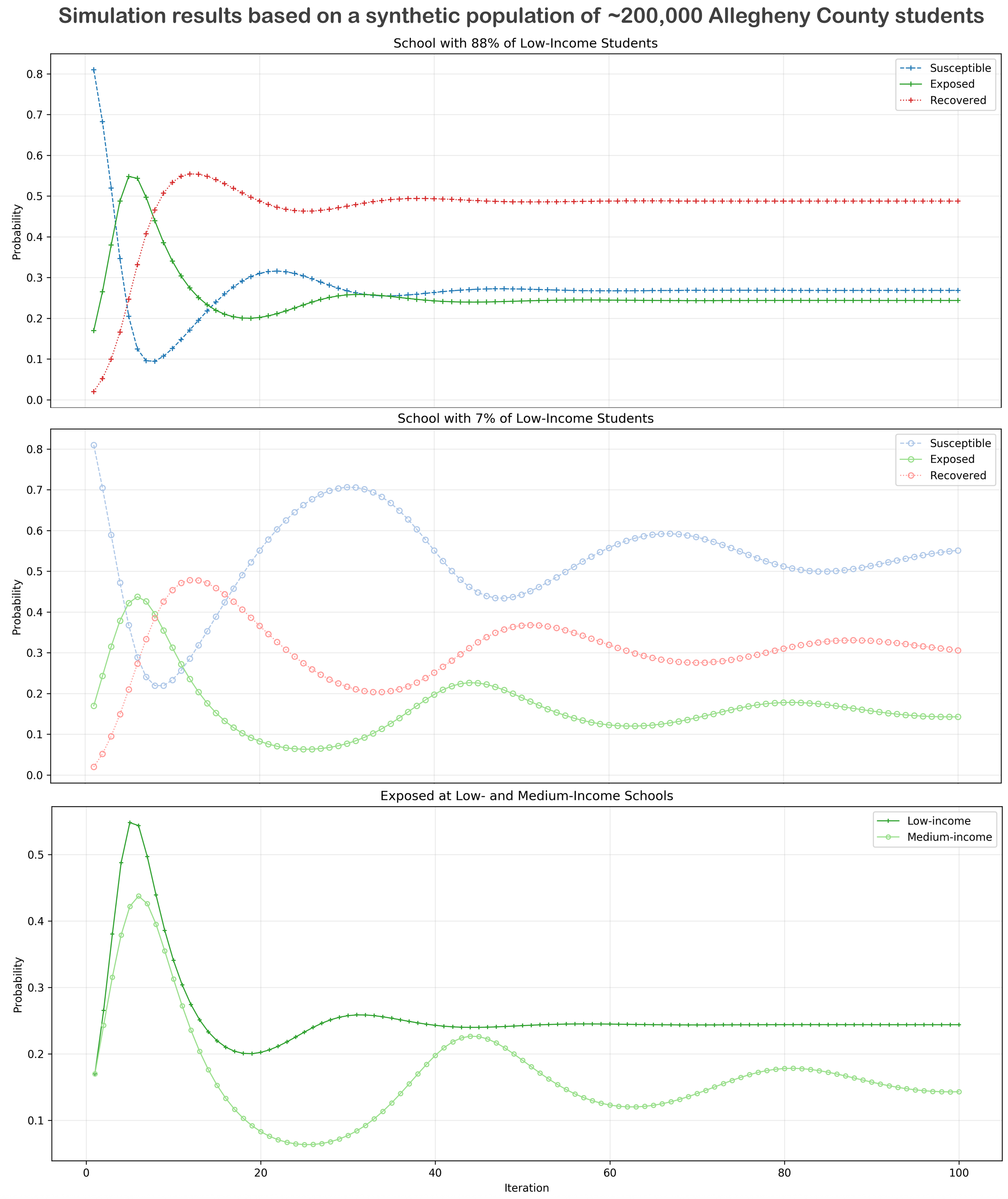
5. Examples of Supported Rules/Models

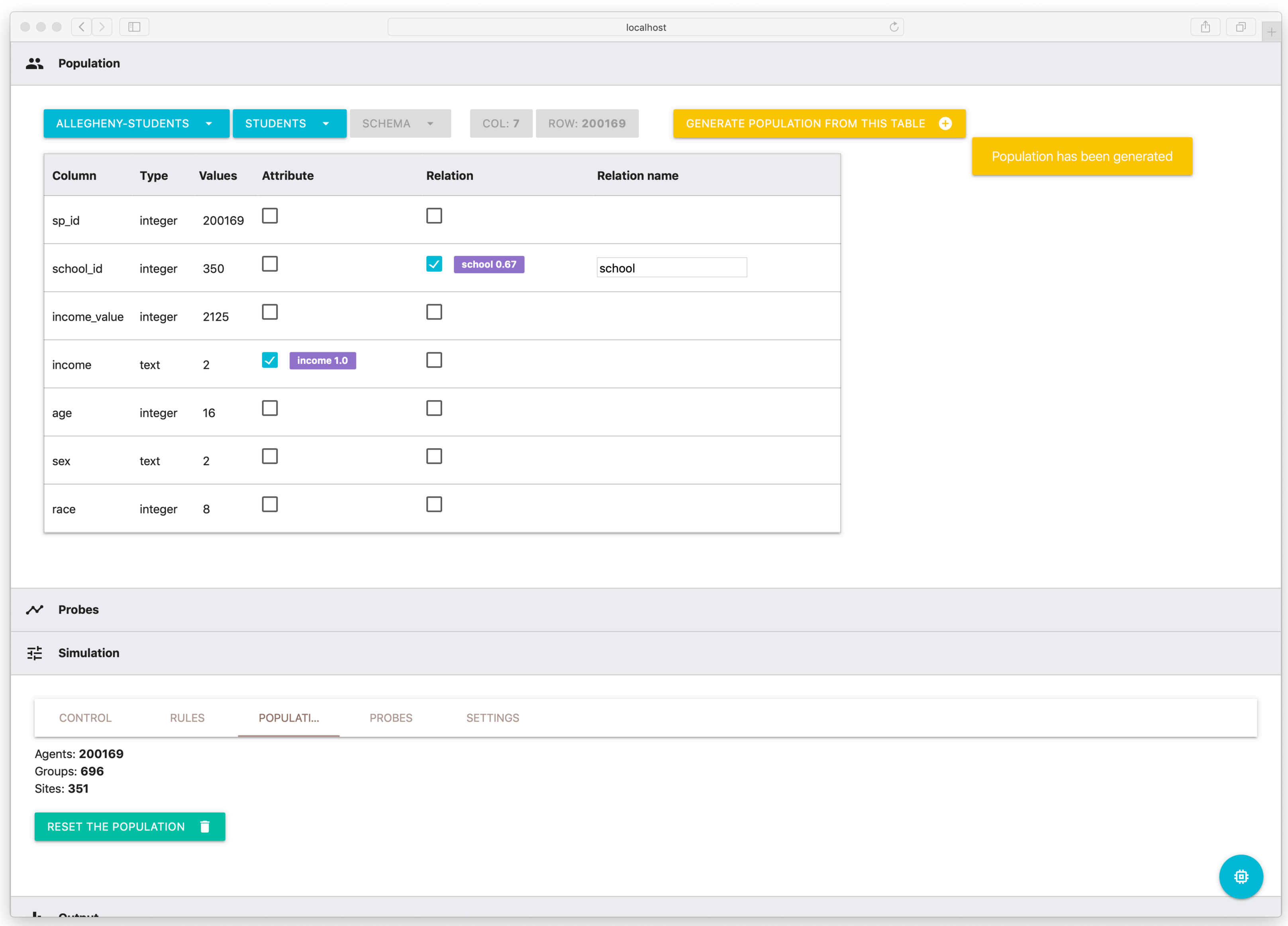
- Fundamental stochastic processes
  - Poisson point process
- Probabilistic
  - Finite state-space time-(in)variant Markov chain
- Epidemiological
  - SIS, SIR, SIRS
- Segregation model

6. On-Going Efforts

- Theoretical work on the equivalence between PRAMs and other model types
- Investigating the relationship between PRAMs and dynamical systems specified via ordinary differential equations
- Accounting for continuous group features and continuous-time simulations
- Extending the definition of population
- Allowing changes to the total population mass
- Ensuring proper amalgamation of different models within the same simulation
- Investigating the pedagogical value of PRAMs

Simulation results based on a synthetic population of ~200,000 Allegheny County students





GitHub /momacs/pram