Pure functional programming in Agent-Based Simulation

Jonathan Thaler

University of Nottingham, Ningbo, China

AIOP Seminar 2019

The Metaphor

- "[..] object-oriented programming to be a particularly natural development environment for Sugarscape specifically and artificial societies generally [..]"
- agents map naturally to objects (North et al 2007)

Outline

- What is Agent-Based Simulation (ABS)?
- What is *pure* Functional Programming (FP)?
- How can we do ABS + FP?
- ABS + FP = Type Safety
- ABS + FP = Software Transactional Memory
- ABS + FP = Property-Based Testing
- ABS + FP = Dependent Types
- Conclusions

What is Agent-Based Simulation (ABS)?

Example

Simulate *the spread of an infectious disease in a city.*
What are the ***dynamics*** *(peak, duration of disease)?*

1. Start with population → Agents
2. Situated in City → Environment
3. Interacting with each other → Local interactions
4. Creating dynamics → Emergent system behaviour
5. Therefore ABS → Bottom-up approach

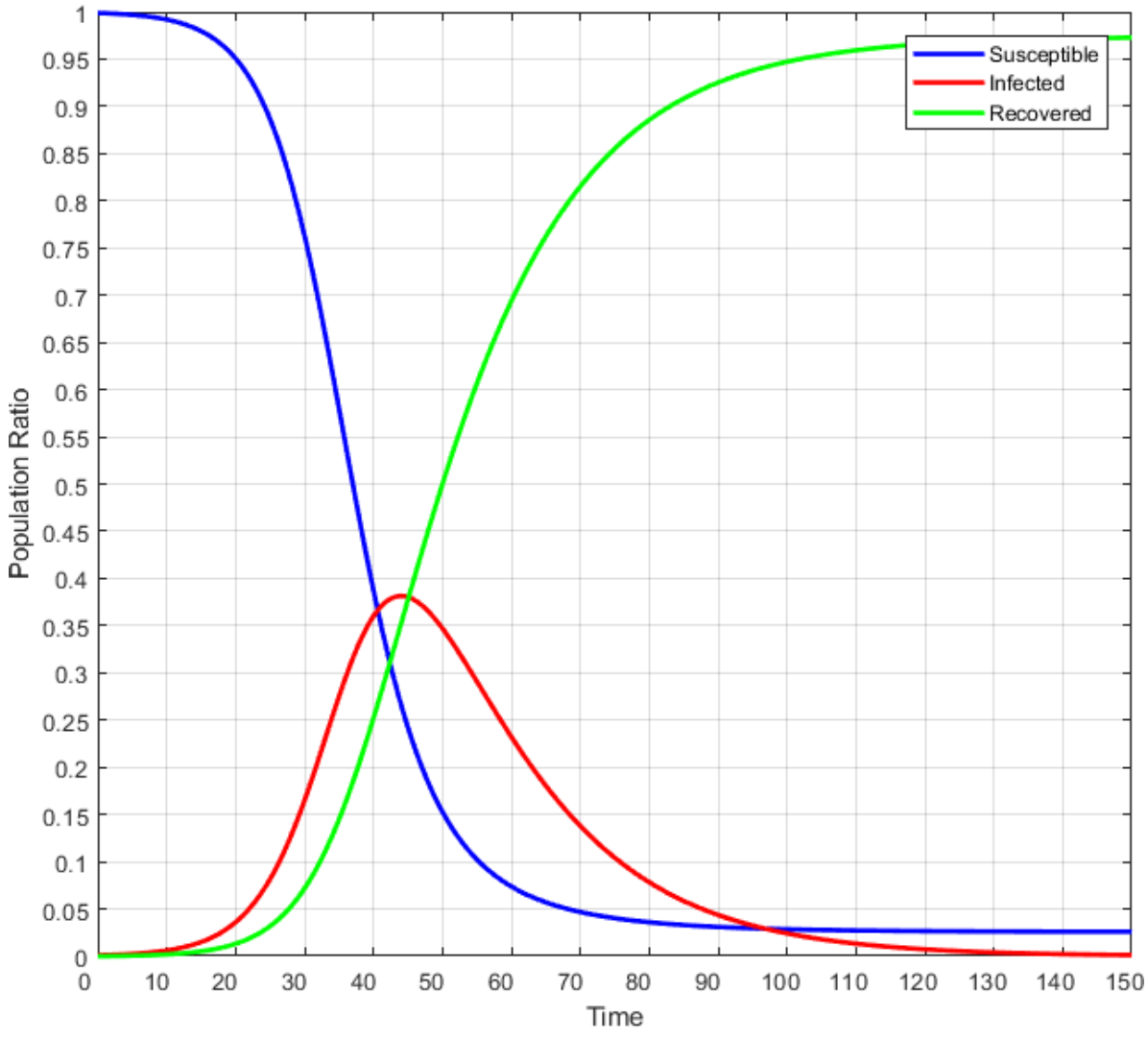
SIR Model

- Population size $N = 1,000$
- Contact rate $\beta = 5$
- Infection probability $\gamma = 0.05$
- Illness duration $\delta = 15$
- 1 initially infected agent

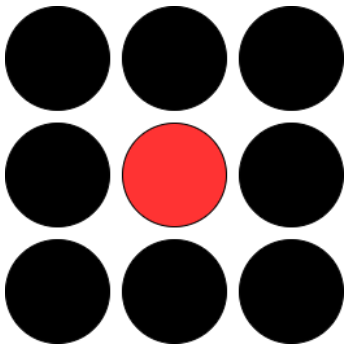
System Dynamics

Top-Down, formalised using Differential Equations, give rise to dynamics.

SIR Model Dynamics

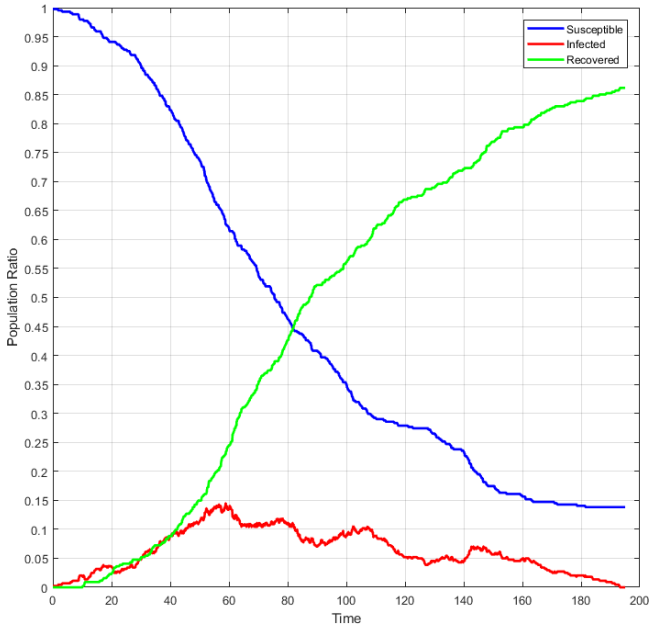


Defining Spatiality

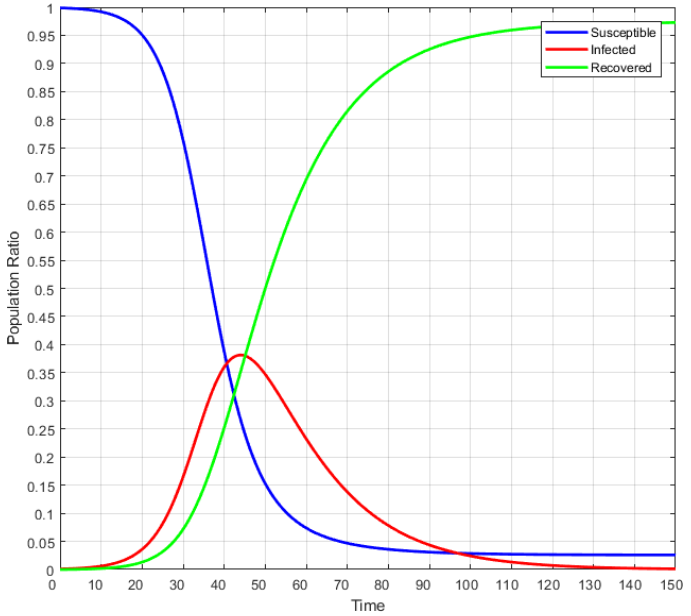


Moore Neighbourhood

Spatial Dynamics



Agent-Based



System Dynamics

Spatial Visualisation

TODO: explain monads: declare type of side-effects statically at compile-time, continuations and closures (which are simple

objects with only 1 method)

How to implement ABS?

Established, state-of-the-art approach in ABS

Object-Oriented Programming in Python, Java,... TODO: the concept of agents maps particularly well to agents

We want (pure) functional programming

Purity, explicit about side-effects, declarative, reasoning, parallelism, concurrency, property-based testing,...

How can we do it?

Functional Reactive Programming

Arrowized Functional Reactive Programming (AFRP)

- Continuous- & discrete-time systems in FP
- Signal Function
- Events
- Effects like random-numbers, global state, concurrency
- *Arrowized* FRP using the *Dunai* library

Monadic Stream Functions (MSF)

Process over time

$$\mathit{SF}\ \alpha\ \beta \approx \mathit{Signal}\ \alpha \to \mathit{Signal}\ \beta$$

$$\mathit{Signal}\ \alpha \approx \mathit{Time} \to \alpha$$

Agents as Signal Functions

- Clean interface (input / output)
- Pro-activity by perceiving time

FRP combinators

Dynamic change of behaviour

```
switch :: SF inp (out, Event e)
       -> (e -> SF inp out)
       -> SF inp out
```

Stochastic event source

```
occasionally :: RandomGen g
             => g -> Time -> b -> SF a (Event b)
```

Random number stream

```
noiseR :: (RandomGen g, Random b)
       => (b, b) -> g -> SF a b
```

Infinitesimal delay (1 step)

```
iPre :: a -> SF a a
```

ABS + FP = Type Saftey

- Purity guarantees reproducibility
- Enforce and guarantee update semantics

ABS + FP = Software Transactional Memory

- Concurrency using Software Transactional Memory (STM)
- Lock free!
- Tremendous performance improvement
- Substantially outperforms lock-based implementation
- STM semantics retain guarantees about non-determinism

STM Semantics

- With Haskell typesystem can be explicit in side-effects: STM only
- Guarantees that the non-determinism comes only from concurrency within STM and nothing else

ABS + FP = Property-Based Testing

- Express specifications directly in code and generate random test cases
- Stochastic nature of Property-Based TEsting and ABS should be perfect match

ABS + FP = Dependent Types

- Types as first class citizen: compute them at compile time
- Types can contain values e.g. the size of a list
- Programs become proofs
- Totality a central concept

Conclusion

- The direction is towards an simulation which is more likely to be correct with the ultimate goal being a correct-by-construction implementation (up to some specification).
- A correct-by-construction implementation does NOT relief us from actually running the simulation!
-

Thank You!