11/22/2105

Treating Interactions in Agent-Based Models as MODULes

# The nature of the MODULE

To be a module an algebraic structure must contain a primary set that is an additive group, G, satisfying four group axioms: closure, associativity, identity and invertibility. There is an operator ⊕ which takes two elements of the group and yields a third element and its operation satisfies these axioms.

In addition, a module contains a secondary set, R, a ring of coefficients, with a second operation, ⊗, which takes an element of R and an element of G and produces and element of G.

## An Analysis of a Generic ABM Interaction as a Module

### Elements: Following Whitehead (2014), we call the elements of our group *prehensions*. A prehension can be roughly understood as a state of affairs in the world *as seen from a particular point of view*. (In this case the world is the world of our model [see Morgan 2012], but Whitehead views this as a useful metaphysics for the actual world.)

### The operation ⊕, which we will call “prehend”, accepts two prehensions as arguments and produces a third prehension.

#### Axioms:

##### **Closure:** Every prehending involving two prehensions will produce a prehension.

##### **Associativity:** (a ⊕ b) ⊕ c = a ⊕ (b ⊕ c) In a typical agent model, this will mean that we must ensure that, say, a neighborhood can interact with a neighborhood (b ⊕ c), and then with an agent (a ⊕ (b ⊕ c)). Furthermore, this must produce an identical prehension to that produced by an agent interacting with one neighborhood and then another one ((a ⊕ b) ⊕ c).

##### **Identity:** Any prehension prehending the null prehension produces itself.

##### **Invertibility:** For any prehension, there is another prehension that combines with it to produce the null prehension.

### The operation ⊗, which we will call “intensify” (although it may also de-intensify) accepts an element of R and an element of G (a prehension), and produces an element of G.

#### Axioms:

##### a, b ∈ G:

###### (a ⊕ b)x = ax ⊕ bx

###### a(x ⊕ y) = ax ⊕ ay

### Meaning:

#### An agent’s prehension of itself is its view of its own internal state.

#### An agent’s prehension of its environment is its view of its surroundings.

#### But from the point of view of the prehension module, these prehensions are interchangeable.

#### A null prehension could arise, e.g., from the environment when an agent has no neighbors. It could arise internally when an agent has “no opinion” on the relevant parameters, e.g., a color-blind agent in our fashion model.

#### Invertibility may occur, for instance, when an agent has some internal tendency to act in some way (e.g., to move to a new neighborhood or switch fashions) but some force in the environment exactly offsets that tendency (e.g., that “authorities” establish come penalty for so acting).

#### An intensification of a prehension leaves the elements of the prehension in the same internal relationship, but they are scaled up or down relative to other prehensions. This is useful for capturing situations like the gradual dissipation of an attitude, or increasing fanaticism over time.

# The advantages of employing this abstraction

## Why bother?

### We achieve a uniform template for all models as far as how agents interact with their environment. Programming new models then becomes much easier.

### We will have taken a huge step towards enabling “fill-in-the-template” style programming of ABMs.

### We open up the possibility of using known properties of modules to identify properties of our ABM.

# a sketch of the usual action pattern

An agent gathers together a prehension of its environment, and then combines that with how it views its own state (its self-prehension) to produce a new prehension, which it adopts as its own. The new prehension may simply be adopted, or it may trigger some further step, such as a movement in space on the part of the agent.

However, while the above may be typical, our model allows the reverse: in some models (e.g., Forest Fire), it may be the environment that adopts the new prehension. Furthermore, environmental prehensions may interact directly with each other as well.

# Implementation

1. import numpy as np
3. X\_VEC = np.array([1, 0])
4. Y\_VEC = np.array([0, 1])
5. NULL\_VEC = np.array([0, 0])

8. class Prehension():
10. @classmethod
11. def from\_vector(cls, v):
12. p = Prehension()
13. p.vector = v
14. return p
16. def \_\_init\_\_(self, x=0, y=0):
17. self.vector = np.array([x, y])
19. def prehend(self, other):
20. return Prehension.from\_vector(self.vector + other.vector)
22. def intensify(self, a):
23. return Prehension.from\_vector(self.vector \* a)
25. def direction(self):
26. if self.vector[0] > self.vector[1]:
27. return X\_VEC
28. elif self.vector[0] < self.vector[1]:
29. return Y\_VEC
30. else:
31. return NULL\_VEC

# Bibliography

Morgan, Mary S. 2012. *The World in the Model: How Economists Work and Think*. Cambridge; New York: Cambridge University Press.

Schelling, Thomas C. 2006. *Micromotives and Macrobehavior*. New York: Norton.

Stepanov, Alexander A, and Daniel E Rose. 2015. *From Mathematics to Generic Programming*. Upper Saddle River, NJ [u.a.]: Addison-Wesley.

Whitehead, Alfred North. 2014. *Process and Reality*. [S.l.]: Free Press. http://rbdigital.oneclickdigital.com.