## Generic Programming and Agent-Based Modeling

Generic programming is inspired by the idea of abstract algebras in mathematics. It involves abstracting algorithms from the data structures that they employ. Its foremost implementation at present is the C++ standard template library, but, in fact, Python, with its dynamic typing system, is an ideal language for employing generic programming techniques.

Agent-based modeling (ABM) is ripe for the employment of this technique, as there are certain patterns of interaction that occur again and again in ABMs. In particular, agents in an ABM typically:

1) Exist in an environment.

2) Get called upon to act in a loop.

3) Survey their environment for cues as to what to do.

4) Act based on the results of 3).

This talk will present efforts to realize generic programming in the context of the Indra ABM system, written in Python. Implementing generic programming in Python relies heavily on the use of iterators, and decorator, map, filter, reduce and lambda features of Python. This talk will show examples of how these are used to create generic algorithms.

In mathematics, there was a process of generalization that moved through things such as:

Common measures

Natural numbers

Integers

Rational numbers

Polynomials

Real numbers

Complex numbers

Groups

Rings

Fields

I think it important to note how this happened: no one sat down and devised an abstract algebra of rings and fields, and then said, "So farmers: I think you can derive something from this to measure your fields." No, people started out measuring fields and buildings and so on, and gradually, over a period of thousands of years, abstracted higher-level constructs from these concrete applications.

Similarly in programming, it seems to me that it is rarely a successful strategy to try to start by defining some grand abstraction, and then building one's system around it. Rather, successful generic programming comes from building concrete applications, and then painstakingly seeking out commonalities found in different applications and abstracting them.

There is a trade-off: the resulting code is going to be harder for people not familiar with generic programming and similar abstractions to understand. But this is OK, because it enables skilled software engineers to create a “fill-in-the-template” system for the use of social scientists who are not highly skilled programmers, and will never (we hope!) need to peer under the hood of the system and cope with the added obscurity a highly generic system can create.

Generic library modules:

|  |  |  |
| --- | --- | --- |
| \_\_init\_\_.py | grid\_env.py | spatial\_agent.py |
| agent\_pop.py | menu.py | spatial\_env.py |
| display\_methods.py | node.py | thing.py |
| entity.py | plane\_agent.py | user.py |
| env.py | plane\_env.py | utils.py |
| grid\_agent.py | prop\_args.py |  |