NetLogo for Building Prototype Multiagent Systems

José M Vidal

Department of Computer Science and Engineering
University of South Carolina

October 25, 2011

Abstract

We introduce NetLogo and its uses for building Multiagent Systems.
Motivation

- Mitch Resnick developed StarLogo as a tool for teaching kids the “distributed mindset.”
  - How do ants gather food?
  - Why do traffic jams happen?
  - How do termites build their nests?
  - Why is there day and night?
Motivation

- Mitch Resnick developed StarLogo as a tool for teaching kids the “distributed mindset.” [?]  
  - How do ants gather food?
  - Why do traffic jams happen?
  - How do termites build their nests?
  - Why is there day and night?

- Kids got all of them wrong.
Motivation

- Mitch Resnick developed StarLogo as a tool for teaching kids the “distributed mindset.” [?]
  - How do ants gather food?
  - Why do traffic jams happen?
  - How do termites build their nests?
  - Why is there day and night?

- Kids got all of them wrong.
- Of course, adults are not much better.
  - Why is Microsoft successful?
  - Why is the economy so bad/good?
  - What does company/country X think?
Resnick argues that it is hard for humans to avoid falling back on a centralized explanation.

He showed that once students get first-hand experience with emergent phenomena they can understand how complex processes can arise from simple interactions.

But, how?
Solution

- Resnick argues that it is hard for humans to avoid falling back on a centralized explanation.
- He showed that once students get first-hand experience with emergent phenomena they can understand how complex processes can arise from simple interactions.
- But, how?
- He created a programming environment: Starlogo.
NetLogo History

- Starlogo is based on Logo
- StarLogo added thousands of turtles to Logo and patches of ground which can hold stuff.
- It ran on the Connection Machine.
- Uri Wilensky, director of the Center for Connected Learning and Computer-Based Modeling extended MacStarLogo and created StarLogoT. Based on what he learned from StarLogoT, Wilensky designed a new language called NetLogo (Wilensky, 1999).
- We will be using NetLogo.
Play is necessary for understanding. Emergent behaviors are often unexpected.
Why NetLogo

1. Play is necessary for understanding. Emergent behaviors are often unexpected.

2. The smaller the program, the more likely it is that its results will be useful to others (model docking problem [Axtell et al., 1996]). Netlogo is succinct.
Why NetLogo

1. Play is necessary for understanding. Emergent behaviors are often unexpected.
2. The smaller the program, the more likely it is that its results will be useful to others (model docking problem [Axtell et al., 1996]). Netlogo is succinct.
3. It greatly reduces the tweak-test-analyze cycle.
Why NetLogo

1. Play is necessary for understanding. Emergent behaviors are often unexpected.
2. The smaller the program, the more likely it is that its results will be useful to others (model docking problem [Axtell et al., 1996]). Netlogo is succinct.
3. It greatly reduces the tweak-test-analyze cycle.
4. It has great expressive power.
Why NetLogo

1. Play is necessary for understanding. Emergent behaviors are often unexpected.
2. The smaller the program, the more likely it is that its results will be useful to others (model docking problem [Axtell et al., 1996]). Netlogo is succinct.
3. It greatly reduces the tweak-test-analyze cycle.
4. It has great expressive power.
5. It is fun to use.
You can download and install it in your machine.
Better to use the enclosed JVM if you can.
A .nlogo file is just plain text so it can be read by any installation.
There is a beta 3D (OpenGL) version.
Overview

- Turtles, Patches, Observer.
You can have different breeds of turtles:

breeds [mailmen letters]
They are like classes.

You can

ask mailmen [set color red]

An alternate notation is

set [color] of (turtle 43) red
Data Types

- Numbers: reals and ints.
- Strings: with many manipulating functions.
- Lists
- That's it! You can use lists as arrays, hash tables, etc.
Turtle Variables

- breed
- color
- heading
- hidden?
- label
- label-color
- pen-down?
- shape
- size
- who (read-only): the number of the turtle
- xcor
- ycor
Turtle Commands

- `forward` to move. `right` and `left` to turn relative to current heading.
- `set heading towards (turtle 22)` to point towards 22. `towardsxy x y` also works.
- `distance (turtle 1)` reports the distance to that turtle. `distancexy x y` also works.
- `[who] of turtles` reports a list with the numbers of all the turtles. It is a very useful command.
- `other-turtles-here` reports all the other turtles in this patch.
Agentsets

- A set of agents.
- Usually generated by a turtles with [...] 
- Also generated by an in-radius, other-turtles-here.
- Manipulated with any, max-one-of, min-one-of, random-one-of, one-of.
- ask takes an agentset as argument.
- Within a with you can use both myself to refer to the turtle doing the command. For example
  ask turtles with [color = [color] of myself] [....] 
- self is the same as turtle who.
Patch Variables

- pcolor
- plabel
- plabel-color
- pxcor
- pycor
Patches

- They can execute commands. You can ask patch 0 0 [set pcolor red]
- They should generally handle the pheromones or other environmental state. They represent the state of the world that is not agents.
- They are useful for representing obstacles (walls) in a multiagent system.
- Can sprout turtles and diffuse pheromones.
Procedures

- Are defined as
  ```netlogo
to do-something
  ...
end
```

- Procedures can also take arguments.
  ```netlogo
to do-something [x y]
  ...
end
```

- Procedures that return a value are called reporters and they are defined as
  ```netlogo
to-report is-it-raining?
  ...
  report true
end
```
You can use

map
to apply a reporter to a list and produce a new list.

show map [? * ?] [1 2 3]
=> [1 4 9]

foreach
loops over instances in a list:

foreach [1.1 2.2 2.6] [ show ? + " -> " + round ? ]
=> 1.1 -> 1
=> 2.2 -> 2
=> 2.6 -> 3
Advanced Operators

- **filter**
  reports a new list with only some of the members of the given list:

  ```
  show filter [? < 3] [1 3 2]
  => [1 2]
  ```

- **reduce**
  primitive lets you combine the items of a list into a single result:

  ```
  show reduce [?1 + ?2] [1 2 3]
  => 6
  ```

- **sort-by**
  sorts a list in any order you specify:

  ```
  show sort-by [?1 < ?2] [3 1 4 2]
  => [1 2 3 4]
  ```
Extension API allows one to define new NetLogo commands using Java.

Good for CPU-intensive computations (backpropagation), defining complex operations (matrix multiplications), or using external libraries (SOAP).

Don’t use in PS.
Read the manual that is installed with NetLogo, especially the primitives dictionary. There are many more primitives than what I showed here.

Do Problem Set 0 to get acquainted with the language.

You can see more MAS models we have created at http://jmvidal.cse.sc.edu/netlogomas/