Petri Nets as Modeling Tool for Emergent Agents

Margo Bergman[1]

[1] Penn State Worthington Scranton, 120 Ridge View Drive, Dunmore, PA 18410
mwb12@psu.edu
http://www.personal.psu.edu/mwb12

Abstract. Emergent agents, those agents whose local interactions can cause unexpected global results, require a method of modeling that is both dynamic and structured. Petri Nets, a modeling tool developed for dynamic discrete event system of mainly functional agents, provide this, and have the benefit of being an established tool. We present here the details of the modeling method here and discuss how to implement its use for modeling agent-based systems.

1 Introduction

Petri Nets have been used extensively in the modeling of functional agents, those agents who have defined purposes and whose actions should result in a known outcome. However, emergent agents, those agents who have a defined structure but whose interaction causes outcomes that are unpredictable, have not yet found a modeling style that suits them. A problem with formally modeling emergent agents that any formal modeling style usually expects to show the results of a problem and the results of problems studied using emergent agents are not apparent from the initial construction. However, the study of emergent agents still requires a method to analyze the agents themselves, and have sensible conversation about the differences and similarities between types of emergent agents. We attempt to correct this problem by applying Petri Nets to the characterization of emergent agents. In doing so, the emergent properties of these agents can be highlighted, and conversation about the nature and compatibility of the differing methods of agent creation can begin.

1.1 Petri Nets

Petri Nets are a graphical modeling tool used mainly to analyze manufacturing processes. The main strength of using Petri Nets lies in the fact that they can handle concurrency of events. For complex modeling the ability to allow several events to occur simultaneously and still analyze their effects on each other is a necessity. The classic Petri Net consists of four objects: *places, transitions, directed arcs* and *tokens*. A *place* is a state of existence for a model. Consider a traffic light which has three states, each of which indicates a different situation; red says stop, yellow indicates caution, and green allows forward motion. Each of these three states would be considered a place in a Petri Net. Places are usually denoted by a circle. *Transitions* are

the means by which the different places are reached. This would be the light changing from red to yellow, yellow to green, etc. You must go through a transition in order to reach a place. Transitions take the form of a square, or a straight line. The *directed arc* links the places to the transition. If one must change from red to yellow, there would be an arc linking the place "red" to the transition "changing from red to yellow," and another linking this transition to the place "yellow." The *token* is the means by which the Petri Net is made active. It indicates at which *place* in the Petri Net the current process is, and allows for restrictions on the activity of the processes. If there were two traffic lights at an intersection the tokens would indicate which is green, which is red, and ensure that only one was green at any given time. Below is the traffic light example shown as a Petri Net (1).

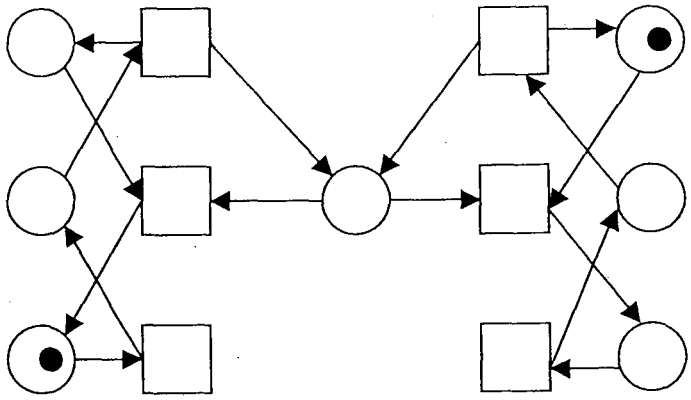

Fig. 1. This represents two traffic lights. The tokens in Green 1 and Red 2 show that the left traffic light is green currently, and the right one, red. Notice that both tokens have to be in spot X for the light to change. This is true of many actual traffic systems, where all lights at an intersection are briefly red before one turns green

This process of putting the Petri Net into action is referred to as *firing*. Each token allows for a single firing, which causes a token to move from place, through a transition, into another place. In order for a particular transition to be *enabled*, all of the places who have a directed arc leading to that transition must have a token. In the case above, the X place creates the situation where both lights must be red for one to turn green.

The model above is an example of a Petri Net that describes a well-defined system, with predictable results, and no emergent properties. Below we describe how this tool can be applied in a situation where emergence generates the interesting result.

2 Emergent Agent Modeling

We create a formal model of a classic agent-based model, the Schelling model of spatial segregation. In this model, there are two types of agents. Each agent has a threshold level for the number of similar agents they wish to have in their 'neighborhood', although no agent has a particular preference for segregation. When chosen by random, each agent takes an accounting of the percentage of each agent type in their neighborhood, and if the percentage of dissimilar agents is too high, they will move to another location. Although simple, the model's results stem from the emergent properties of the heterogeneous agents. Although this is not a detailed model, there are still many choices a researcher must make when programming the simulation, such as the type of neighborhood the agents live in, how the thresholds are determined, the method of location switching, etc. Each model has characteristics in common, however, and it is these characteristics that should be included in formal specification of the model. Below is a Petri Net model of the basic characteristics that should be included in every Schelling simulation, regardless of the individual choices made by the researchers.

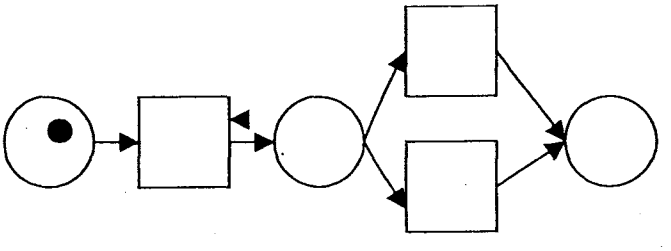

Fig. 2. A Petri Net of the Schelling model of spatial segregation [2]. Here, an agent has a certain threshold of similar agents they want in their "neighborhood". If that percentage falls beneath their threshold, they will choose to move. The Petri Net shows the basic model, without requiring knowledge of the specific parameters

Given this model of the process that the researcher is trying to analyze, and the specifics of the choices that she made in the design process, the original results should be replicable. In addition there is no need for every researcher to utilize the same programming language or software package in order to understand the workings of the model. Petri Nets are dynamic which makes them ideal for analyzing the structure of

agent-based models, whose results usually rely on the dynamic interactions of their component parts.

3 Conclusion

These Petri Net models do not replace the agent-based model itself. The emergent nature of many agent-based results still requires a full computational simulation to be created. However, they do provide a method by which two modelers can discuss a single problem without being distracted by the particulars of their individual models. Since Petri Nets are mathematically based, issues of the efficiency of the model can also be analyzed. Finally, there is already an established body of work in the field of Petri Nets, which prevents agent-based modelers from having to invent new systems of analysis. Just as economics and other fields adapted calculus for their own uses, agent-based modelers in all disciplines can use this technique.

References

1. http://tmitwww.tm.tue.nl/staff/wvdaalst/Petri_nets/pn_tutorial.html July 2004
2. Schelling, TC. Micromotives and Macrobehavior. W. W. Norton & Company, 1978