DEALING WITH VAGUENESS IN AGENT-BASED MODELS: A PYTHON FUZZY LOGIC ABM FRAMEWORK

Andrei LUCHICI[1]

Abstract

Complex systems are everywhere; countless examples of behaviors fall into the complex systems paradigm, from physical and natural sciences to social and economic sciences. Given the nature of these systems, where the whole is greater than the sum of its constituents, scientists must have adequate tools for investigating complex systems. Recently, Agent-Based Models (ABM) have become a de facto tool for creating idealized computer simulations to investigate pattern formation, perform root-cause analysis, or simulate alternative scenarios within the domain of complex systems. This paper introduces a miniature framework for developing and analyzing agent-based models where agents and the environment can follow vague rules. The proposed tool is applied to a sample simulation, providing a proof-of-concept example of how Fuzzy Logic and Fuzzy Inference can model complex systems with vague rules.

Keywords: complex systems, agent-based modeling, fuzzy logic, fuzzy inference, rule-based systems, rule-based simulation, simulation

JEL Classification: C63

1. Introduction

Simulations are increasingly becoming a de facto tool for understanding complex phenomena. Traditional approaches relied heavily on induction, i.e., generating theories based on collected data, or deduction, i.e., starting from first principles and creating a general theory. However, with the advent of computers and increasing processing power and data, researchers are starting to adopt a new way of doing research, namely, simulations or generative methods [9].

Simulations can replicate complex systems and processes as closely as possible. Using computer simulations allows researchers to analyze and evaluate the effects of different parameters and variables on a given system. By allowing users to experiment and test different scenarios, simulations can be a powerful tool for understanding and predicting the behavior of complex systems [3].

Computer simulations are helpful in various fields, such as physics, engineering, economics, and biology. For example, in economics, simulations can be used to analyze the effects of various policies on the market and the economy [5]. In biology, scientists use computer simulations to study organisms' behavior and environment [6].

[1] PhD Associate Lecturer, Romanian-American University, andrei.luchici@rau.ro

Moreover, social scientists are increasingly using computational tools and simulations to understand the effects of different variables on a particular population or system. For example, ABMs model the spread of an epidemic or the effects of a particular policy on a specific group of people. Simulations are becoming increasingly important in research and development as they allow researchers to quickly and accurately model complex systems [3].

Agent-based models (ABMs) are a type of computer simulation used in social sciences and economics research to analyze the collective behaviors of autonomous agents interacting with each other in a well-defined environment. Agent-based modeling combines individual-level behaviors, micro-macro dynamics, and emergent phenomena to explain observed societal patterns. ABMs are used to understand the collective behaviors of complex systems, such as the effects of different types of social networks, the behavior of markets, and the dynamics of economic growth. ABMs have been used to explore the effects of policy interventions, such as taxes, subsidies, and regulations, and to study the formation and evolution of social norms [9].

Presently there are several frameworks and tools available for building agent-based simulations[2]. Java, C/C++, or Objective-C are the main programming languages used to build these tools. More recently, Kazil et al. proposed Mesa [10], a Python-based framework for developing ABMs. Another famous example is NetLogo created by Uri Wilensky[3]. This tool was used extensively to build ABMs for different physical, biological, and social sciences processes and problems. More recently, more tools emerged as complex systems research has gained momentum. However, most of the tools available only deal with binary logic and uncertainty, i.e., probabilities. In contrast, many of the theories available, especially in the social and biological sciences, rely on vague descriptions of the processes. Therefore, there is a real need for new tools that can work with vague terms and definitions and make computations using these vague concepts to evolve the state of the agents.

Lotfi Zadeh introduced fuzzy logic in 1965 to help computers represent vague concepts. Since then, scientists and engineers have used fuzzy logic in various applications, including decision-making algorithms, natural language processing systems, and robotics control systems [4]. Fuzzy logic can be used for many applications, including fuzzy control systems, decision-making algorithms, pattern recognition and classification systems, natural language processing systems, and robotics control systems. In fuzzy control systems, a set of rules written in terms of linguistic variables determine the system's output, unlike classical approaches, such as equation-based modeling. Through fuzzy logic, a digital computer can understand the real-world environment better and respond to changes more quickly than traditional control methods [4].

In decision-making algorithms, fuzzy logic provides a way to make decisions based on uncertain or incomplete information. For example, it could be used in an expert system that helps a doctor diagnose an illness where there may not be enough data available for a definitive diagnosis. In pattern recognition and classification systems, fuzzy logic can help computers recognize patterns more accurately than conventional methods. It also has

[2] A list of several ABM modeling software can be found here: https://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software

[3] https://ccl.northwestern.edu/netlogo/

applications in natural language processing, where it can help computers interpret ambiguous sentences more effectively than traditional rule-based approaches. Finally, it has been applied successfully in various robotics control tasks where its ability to capture uncertainty makes it especially useful when dealing with a dynamic environment [4].

By coupling ABMs and Fuzzy Logic, one can obtain a framework for modeling complex behavior in the presence of imprecise information. In this framework, ABMs are used to model the dynamics of a system, and Fuzzy Logic is used to represent the imprecise information. Using Fuzzy Logic, we can help computers represent vague concepts difficult to describe using traditional mathematical equations.

This combination can be used to simulate various types of interactions between agents in a system and examine their effects on different outcomes. For example, it can be used to model how different policies or strategies might affect the behavior of an economic system or how specific environmental conditions might lead to changes in the population dynamics of a species. In recent years, it has become increasingly crucial for modeling complex behaviors in agent-based simulations due to its ability to capture uncertainty and provide more accurate simulations of real-world environments than traditional methods alone can provide. By combining these two powerful tools, researchers can gain valuable insight into the behavior of complex systems.

This paper proposes a new Fuzzy Logic framework for creating agent-based simulations. The framework combines fuzzy logic and ABMs to create an environment where agents can interact with one another and the environment. It then uses fuzzy logic to represent the imprecise information in real-world environments, allowing for more accurate simulations of complex behavior.

2. Fuzzy Logic and Inference

2.1 Fuzzy Logic

Our brains have evolved to make computations without requiring high precision. For us, concepts like tall, thin, short, wide, and close do does not require a precise definition, i.e., an object is tall if its height is more than 1.52m. However, a digital computer will need help to solve this task because it requires a precise and objective definition of every concept. More, there are many situations where "tall" might mean 1.23m (think about toddlers versus children), or "wide" might mean 1 meter or 7 meters (think about a coffee table versus a highway lane). In other words, we, as humans, are used to doing computations with vague concepts, and we often attribute slightly different interpretations to them depending on the context. Vagueness is an issue when we want computers to perform human-like actions because they operate using a much more restrictive concept, binary logic, where something belongs to just one class or has just one meaning. Fuzzy logic helps solve these issues. It allows us to abandon the binary way of thinking about concepts and embrace vagueness where things can simultaneously belong to different classes, i.e., the glass can be both half full and half empty to some extent [4].

Linguistic variables represent the core of Fuzzy Logic. They are used to represent quantities of interest in the system under investigation. Linguistic variables allow us to define concrete and vague concepts such as speed, distance, age, or danger. When we model a system, we usually capture the essential variables on which the system's behavior depends.

A linguistic variable is defined by its terms. Depending on the granularity we require, we can have any number of terms that define that variable. For example, think about a person's age. One can describe their age as “young” or “old”. Similarly, one can describe a person as being a “child”, “teenager”, “young adult”, “adult”, or “senior”. Both descriptions are valid depending on the context in which they are used. However, more than these descriptions are required for computations because a digital computer operates with crisp values to perform any computation.[4]Therefore a digital computer cannot perform any computations using terms like "young", making it very difficult to describe the behavior of a system using vague rules. Fuzzy logic proposed a mechanism for converting the linguistic description of a variable, object, or action into a numerical counterpart using membership functions to help the computer understand vague concepts [4].

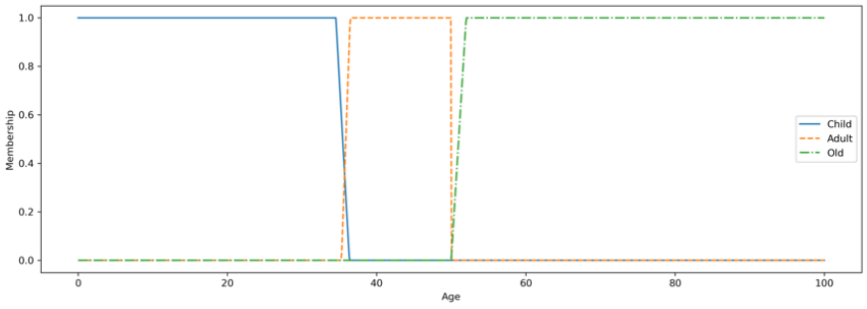


Figure 1. Sample membership functions describe the age of a person. Note that someone can be both a "child" and an "adult" with various degrees of membership.

Membership functions define the extent to which a value belongs to a given term. As mentioned before, classical logic has only two states. A variable can belong to one and only one set at a time. In fuzzy logic, membership functions define the degree of belongingness, ranging between 0 and 1, with 0 meaning that the object does not belong to that set and 1 representing a full membership. The membership function requires a universe of discourse, i.e., all the possible values that a variable can take. Returning to our example, Figure 1 contains a possible interpretation of the terms used to describe a person's age. In the example, the universe of discourse is [0,100], which are all the possible values for a person's age in years. Then, we defined five terms for this variable: child, adult (40-60), and old (60-100). As shown in the figure, each term has a corresponding membership function that defines how much each value belongs to that term. For example, if we take the value 39, it belongs more to "adult" than to "child" since its degree of membership is higher in the former case than in the latter one [4].

[4] Crisp variables represent variables that have precise numerical values. They are used in contrast with fuzzy values that describe a vague concept.

2.2 Fuzzy Inference

Fuzzy inference represents a set of tools used for making decisions using imprecise information. Fuzzy inference systems are used in various applications, such as computer vision, robotics, and decision-making. In fuzzy inference systems, the system is given a set of input variables representing some feature or property of the system (e.g., temperature) and an output variable representing the desired response (e.g., turning on the air conditioner). Applying a set of fuzzy rules to the input variables computes a final output variable that can be converted to a crisp value or reused as input for evaluating more rules. Fuzzy rules are derived from knowledge about how specific inputs should affect specific outputs and often rely on expert opinion or historical data. The output from a fuzzy inference system can be used to control physical devices such as robots, or it can be used to make decisions in business settings such as finance or marketing analysis [4].

A Mamdani system is one of the most common Fuzzy Inference methods [1]. This system takes a set of inputs, applies fuzzy logic operators such as AND, OR, and NOT to them, and then produces an output. In practice, there are more alternatives for performing Fuzzy inference. One common alternative is the Sugeno approach [2].

The Mamdani system first determines a set of rules that will be used to evaluate the inputs. Each rule provides information regarding how each input should affect the output. After determining the rules, the system is given a set of values for each input variable and then uses these values to determine how each rule should be applied. A Mamdani system allows for decisions or actions to be taken based on imprecise information or data that may have yet to have definitive answers [1]. A schematic representation of a Mamdani system can be observed in Figure 2.

Fuzzy inference systems can also be used in medical diagnosis and expert systems, which help automate decision-making processes within complex problem domains. Fuzzy inference has been used in medical diagnosis systems where it helps doctors identify diseases more accurately than traditional methods based on symptoms alone [7]. In business applications, fuzzy inference has been used in financial forecasting models where it helps predict stock market movements more accurately than traditional models using technical analysis alone [8].

Overall, the fuzzy inference is an essential and valuable tool for making decisions when precise information is unavailable or difficult to obtain. It provides a powerful way of automating decision-making processes by considering many different inputs while still providing reliable outputs that can be used to make decisions, making fuzzy inference a valuable tool in many applications and problem domains [4].

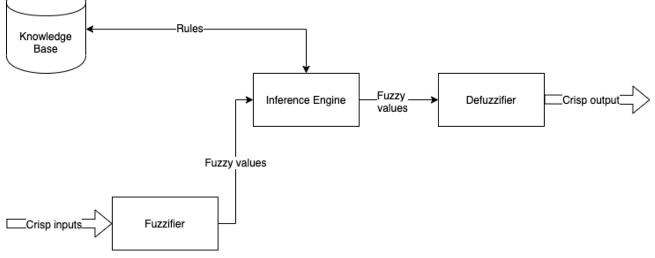


Figure 2. Overview of a generic Mamdani Fuzzy Inference System.

3. Framework Development

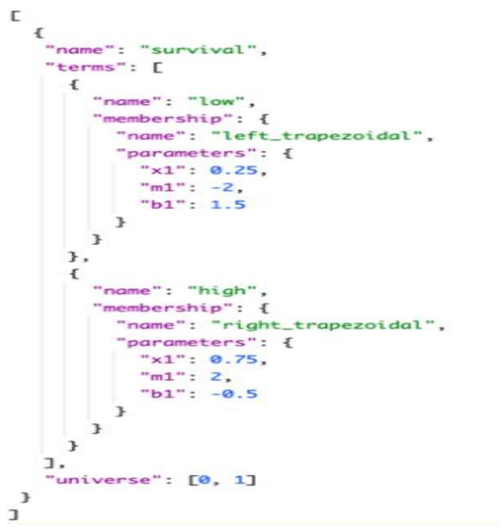
The tool described in this paper was designed as a modular framework that can easily be adapted to fit multiple scenarios. The tool's core is represented by two modules: the fuzzy module and the agent module. Additionally, a simulation module is required where the user can define special agents, provide the fuzzy rules governing their behavior, and set up the environment and other mechanics required to simulate the desired situation. Finally, figure 3 presents an overview of the tool and its main modules, which will be discussed in more detail in the subsections below.

3.1. The Fuzzy Module

The Fuzzy Module consists of five modules that provide the mechanism for defining linguistic variables and Fuzzy rules, creating, and instantiating membership functions, and performing Fuzzy inference.

The module defines a Vocabulary that represents a list of all available Linguistic Variables required for modeling the system. Each Linguistic Variable is composed of a set of Terms, with every term having a predefined Membership Function. When simulating a system, the user needs to specify the Vocabulary using a JSON configuration file. This configuration file describes the concepts available for defining the agents' behaviors (rules) and for evolving the simulation over space, time, or both. An example JSON configuration file for a vocabulary containing just one variable, SURVIVAL, that can model the survival probability of an agent based on a vague definition of survival can be found in figure 3.

Every Fuzzy variable is defined by a set of terms, for example, “low” and “high” in the example presented in Figure 3. These terms are characterized by a membership function that maps a crisp input to a corresponding membership degree (see Section 2 for more details). Finally, to complete the definition, a universe of discourse is provided.



```
[
  {
    "name": "survival",
    "terms": [
      {
        "name": "low",
        "membership": {
          "name": "left_trapezoidal",
          "parameters": {
            "x1": 0.25,
            "m1": -2,
            "b1": 1.5
          }
        }
      },
      {
        "name": "high",
        "membership": {
          "name": "right_trapezoidal",
          "parameters": {
            "x1": 0.75,
            "m1": 2,
            "b1": -0.5
          }
        }
      }
    ],
    "universe": [0, 1]
  }
]
```

Figure 3. Example JSON configuration defining a Linguistic Variable (“SURVIVAL”) with two terms, “LOW” and “HIGH", and their associated membership functions.

The tool presented in this paper contains the most common membership functions already defined: uniform, linear, step, triangle, trapezoidal, and sigmoid functions. Additionally, the tool has a simple interface for defining custom membership functions if needed. Each membership function is implemented as a new class derived from a base class. To define a new membership, one has to implement the method *func()*, which takes an input, x, as an argument together with the predefined values of any parameter required to evaluate the function. The JSON configuration sets the values for all additional function parameters (as can be seen in the *parameters* key in Figure 3)

In addition to the Vocabulary and Linguistic Variables, the Fuzzy module also includes the functionality for defining and evaluating Fuzzy rules. Each rule is built from an antecedent (the IF part) and a consequent (the THEN part). If more terms are needed to define the antecedent, they can be combined using standard logical operators (AND, OR, and NOT). Moreover, in the current implementation, only simple, single-value consequents are supported.

Finally, the Fuzzy module implements a mechanism for performing Fuzzy inference. This mechanism fuzzifies a crisp value for one or multiple inputs. Their fuzzy values are combined, and finally, a crisp output is produced using the Center of Gravity defuzzification method [4]. At the end of each rule evaluation, a numerical output is obtained for every consequence of the evaluated fuzzy rules. The implementation follows a Mamdani inference system, as described in Section 2.2.

The current version of the package requires the domain expert to design the knowledge base (the behavior rules) and the Vocabulary (linguistic variables). Manual or expert-driven vocabulary and rule design can be difficult, especially if there is little theoretical or experimental knowledge about the system. Moreover, this approach can lead to biased results as the user has a particular belief when designing the knowledge base.

Future versions of the package will introduce machine learning algorithms that can learn from data and automatically generate the knowledge base and Vocabulary for a given problem. Adding machine learning to this tool would eliminate the need for manual rules design, making it easier for domain experts to build and use fuzzy logic systems quickly. Additionally, this approach reduces results bias using actual data rather than subjective assumptions.

3.2. The Agent Module

The Agent module serves as a basis for defining specific agents. This module helps the user quickly define agents that abide by a set of rules and perform inferences to update the agent's state according to specific Fuzzy rules. To define an agent, one needs to provide the agent with knowledge (Fuzzy rules describing the agent's behavior), an initial state, a vocabulary (used to reason about the world), and an Inference mechanism. The user also needs to implement a *next()* method for each agent type to describe how the agent's state is updated after all the rules are evaluated. Finally, it is essential to note that the Agent module cannot learn that only some agents can operate with classic rules or thresholds. This implementation decision allowed agents to deal with fuzzy concepts first. Subsequently, other versions will expand the agent's capabilities to work with classic rules and thresholds.

3.3. The Simulation Module

To develop an agent-based simulation, one usually starts with a theory of behavior in a prescribed setting. Next, one can follow one of two paths, one led by domain experts or one by available data about the system.

The first approach relies heavily on experts to determine the linguistic variables required to describe the behavior, the agent types, and other parameters that might influence the system. Following the expert opinions, one proceeds to develop a set of Fuzzy rules and implement the simulation logic. One may be interested in creating a post-processing pipeline for analyzing and understanding the simulation results.

The second approach uses data about the system. One uses data collected about the system to discover the linguistic variables required to describe the system using a clustering or dimensionality reduction algorithm. Subsequently, one can also infer the membership functions for each linguistic term using the available data via another clustering procedure followed by a statistical modeling step whereby one tries to determine the probability of an instance (observation) belonging to a given linguistic term. Next, one can determine the Fuzzy rules driving the system and implement the simulation logic. Finally, a post-processing pipeline for analyzing and understanding the simulation results should be developed similarly to the previous approach.

Once the simulation is built and running, it is time to validate it against the initial theory of behavior to assess its accuracy. Once validated, one can start running experiments with different combinations of parameter values to observe their effect on the system's behavior or performance. Using fuzzy logic and agent-based simulation models can be an effective way to model complex systems. By leveraging the power of fuzzy rules and agents to represent parameters, behaviors, and interactions in a system, one can create a powerful tool for predicting population dynamics. Moreover, using data-driven approaches to determine the linguistic variables and membership functions allows one to develop practical simulations that accurately capture real-world behavior quickly.

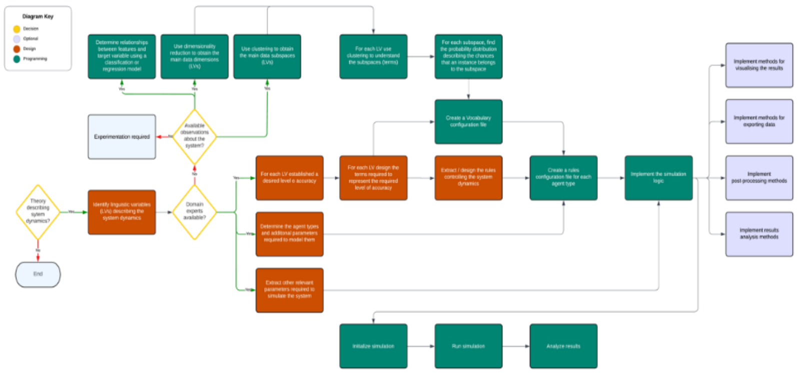


Diagram Key
Decision
Optional
Design
Programming
Determine relationships between features and target variable using a classification or regression model
Use dimensionality reduction to obtain the main data dimensions (LVs)
Use clustering to obtain the main data subspaces (LVs)
For each LV use clustering to understand the subspaces (terms)
For each subspace, find the probability distribution describing the chances that an instance belongs to the subspace
Yes
Yes
Yes
Experimentation required
No
Available observations about the system?
Create a Vocabulary configuration file
No
Yes
For each LV established a desired level o accuracy
For each LV design the terms required to represent the required level of accuracy
Extract / design the rules controlling the system dynamics
Create a rules configuration file for each agent type
Implement the simulation logic
Theory describing sytem dynamics?
Yes
Identify linguistic variables (LVs) describing the system dynamics
Domain experts available?
Yes
Determine the agent types and additional parameters required to model them
No
End
Yes
Extract other relevant parameters required to simulate the system
Implement methods for visualising the results
Implement methods for exporting data
Implement post-processing methods
Implement results analysis methods
Initialize simulation
Run simulation
Analyze results

Figure 4. Diagram showcasing the process for developing an agent-based simulation.

The current implementation of the simulation module aims to help the user create an Environment for the agents, define the system's parameters under investigation, and prescribe additional rules for updating parameters because of the agent's behavior. In addition, the Simulation module also implements the methods required to visualize the system's evolution over time and/or space. It is important to note that the simulation module has two options for updating the states of the agents. First, one can set up the simulation to update the agent states in the order in which they were added (sequential update). Second, one can set up the simulation to update agent states randomly, picking one agent without replacement until all the agents update their states. Finally, one can select a simultaneous state update. Subsequent versions of the package will also support a simultaneous update mode, whereby all agents operate in parallel at every step. Finally, the current simulation module supports only time-based simulations, i.e., agents evolve from one simulation step to another without any knowledge of their position, proximity to other agents or environment objects, or connection to other agents (network). Different simulation versions will eliminate this restriction, allowing users to develop more comprehensive and realistic simulations.

4. Example Simulation: Population Dynamics

The current framework was used to simulate the dynamics of a simple population consisting of two classes of agents (“Conscious” and “Selfish”). Each agent class has two subtypes or gender. The main idea behind this sample simulation is to investigate the dynamics of two populations. The first population is conscious of the environment and guides its behavior depending on the available resources trying to prolong the population's lifespan for as long as possible. The second population is selfish about the environment and does not consider the available resources too much to update their behavior. Also, the selfish population contributes to replenishing the available resources.

Agents work with five concepts or linguistic variables: AGE, REPRODUCE, SURVIVAL, PRODUCE and CONSUME. The AGE variable determines the degree to which an agent belongs to three classes, CHILD, YOUNG, or OLD. This computation is done based on a crisp age value. Next, the REPRODUCE variable determines the chances (LOW or HIGH) of an agent reproducing during a given simulation time step. The agent class, available recourses, and age influence an agent's reproduction chances.

Moreover, the SURVIVAL variable encodes an agent's chances (LOW or HIGH) to survive during a given iteration. The agent's type, available resources, and age also influence an agent's survival chances. Finally, the PRODUCE and CONSUME variables guide how many resources the agent will consume and produce based on the agent's class and age. It is essential to notice that although the agents belong to two gender classes (MALE and FEMALE), this information does not influence the agent's behavior. However, the agents will reproduce if at least two agents, one MALE and one FEMALE, have a reproduction change greater than 0.5. The simulation ends when no agents are left or when the agents consume all the available resources.

The agent's behavior is encoded using two sets of rules, one set defines the behavior of "Conscious" agents, and one defines the behavior of "Selfish" agents. The main differences between the two types of agents are that "Conscious" agents always produce many resources and reproduce less if the available resources are "Low". The exact rules used to represent the agent's behavior can be found in the example simulation online[5]. Finally, it is essential to note that the membership functions for each term are all hypothetical at this stage as this work aims to develop a working framework, not to simulate population dynamics accurately. To make an accurate simulation, one needs to follow a systematic methodology for creating the Vocabulary, the Rules, and any other agent behavior, as presented in Section 3.

4.1. Results

The population dynamics simulation presented in this paper uses six linguistic variables described by two or three terms (see Section 3 for more details about the variables). Several membership functions describe the terms for every linguistic variable to highlight the framework's capabilities introduced in this work. The linguistic variables and their corresponding terms are shown in Figure 5. The membership functions that describe every linguistic variable are chosen to illustrate the framework's capabilities.

Three simulations were performed using the same setup (Vocabulary, knowledge base, environment) to investigate the survival rate of a population of 50 agents composed of selfish and conscious agents given different ratios of agents of each type. Based on the results of the simulations, a population where selfish agents are more prevalent than conscious agents do not manage to survive for a long time.

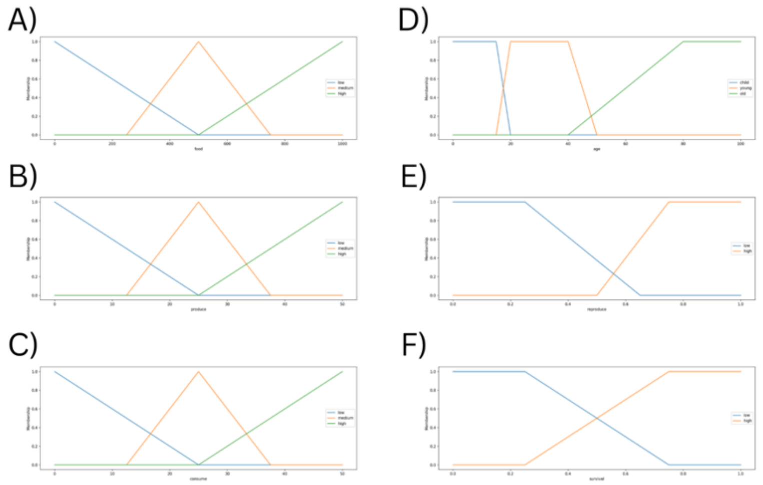


Figure 5. Definition of all linguistic variables used for the population dynamics simulation. A) Membership functions describing the food produced by each agent. The membership function has three levels, "low", "medium", and "high”. B) Membership functions describing the available food quantity. The membership function has three levels, "low", "medium", and "high”. C) Membership functions describing the food consumed by each agent. The membership function has three levels, "low", "medium", and "high”. D) Membership functions describing the agent's age. The membership function has three levels, "child", "young", and "old”. E) Membership functions

[5] https://github.com/aluchici/fuzzy-logic-framework

describing the agent's reproduction rate. The membership function has two levels, "low" and "high". F) Membership functions describing the agent's survival rate. The membership function has two levels, "low" and "high”

In Figure 6, panels A) to C), a population where 75% of the agents were "selfish" survived for only three rounds, with the available resources declining rapidly. In contrast, when the proportion of selfish agents is less than that of the conscious agents, the population can survive for a more extended period. Panels D) to F), a population of 50% selfish and 50% conscious agents survived for the whole duration of the simulation. Moreover, the available resources have increased over time. It is interesting to note the apparent oscillatory behavior of the available resources as time evolves. More experiments need to be performed to investigate if this behavior is consistent for populations where the number of conscious and selfish agents is the same.

Furthermore, in panels G) to I), a population of 25% selfish agents also survived the simulation. It displayed a similar behavior as a population with an equal number of agents of each type. In this case, it is interesting to note that the number of agents varies, with the conscious agents increasing over time while the selfish agents appear to stabilize. However, to prove this observation, more experiments need to be conducted.

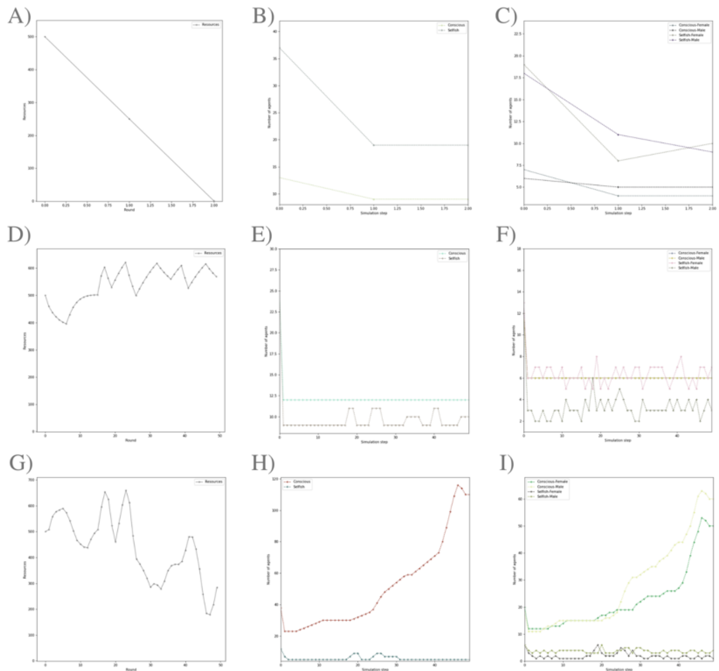


Figure 6. Three example simulations where agents are initialized with a random survival and reproduction rate for the first round of the simulation. A) – C) Evolution of the population of 50 agents. Initial resources are set to 500, representing 50% of the maximum amount. Agents are split into 25% Conscious and 75% Selfish, with a 50% ratio of Female agents for each agent type. D) – F) Evolution of the population of 50 agents. Initial resources are set to 500, representing 50% of the maximum amount. Agents are split into 50% Conscious and 75% Selfish, with a 50% ratio of Female agents for each

agent type. G) – I) Evolution of the population of 50 agents. Initial resources are set to 500, representing 50% of the maximum amount. More, agents are split into 75% Conscious and 75% Selfish, with a 50% ratio of Female agents for each agent type.

It is important to note that every simulation presented in Figure 6 initializes a new population of agents of different ages with random chances of survival and reproduction in the first round. It could be that 50% of the agents may disappear during the first round could explain the sharp decrease in the number of agents. Also, this can contribute significantly to the chances of survival. Therefore, additional simulations were performed.

More simulations were performed to explore the influence of random survival and reproduction rates in the first round of the simulation. For this batch, all agents survive the first round. The results of some of the simulations are presented in Figure 7. Panels A) to C) show different behavior compared to the case presented in Figure 6. Specifically, one can observe that more than a 50% ratio between Conscious and Selfish agents is needed to ensure the population's survival. Panels D) to F) and G) to I) show that when all agents survive the first round, the population does not survive for the whole simulation duration, even if 75% or 85% of the agents were Conscious.

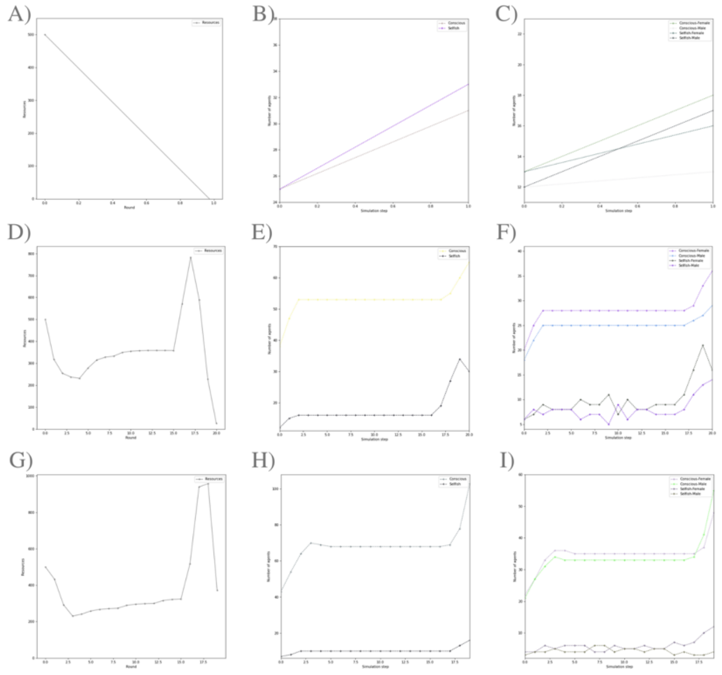


Figure 7. Three example simulations where agents are initialized with a guaranteed survival for and random reproduction rate for the first round of the simulation. A) – C) Evolution of the population of 50 agents. Initial resources are set to 500, representing 50% of the maximum amount. Agents are split into 50% Conscious and 50% Selfish, with a 50% ratio of Female agents for each agent type. D) – F) Evolution of the population of 50 agents. Initial resources are set to 500, representing 50% of the maximum amount. More, agents are split into 75% Conscious and 25% Selfish, with a 50% ratio of Female agents for each agent type. G) – I) Evolution of the population of 50 agents. Initial

resources are set to 500, representing 50% of the maximum amount. Agents are split into 85% Conscious and 15% Selfish, with a 50% ratio of Female agents for each agent type.

Furthermore, another batch of simulations was conducted. For this new batch, the survival rate was a random value between 0 and 1. The reproduction rate was fixed at 1 for the first round of the simulation. It can be observed that a population consisting of an even number of Conscious and Selfish agents can survive for the entire simulation duration. Moreover, a similar behavior as in Figure 6 is observed in panels A) to C). Surprisingly, when the ratio of Conscious agents increased to 75% or 85%, respectively, the population could not survive for the entire duration of the simulation. Instead, the population quickly disappeared as the number increased (see panels D) to F) or G) to I)).

From these simulations, several interesting behaviors emerge. First, in several simulations, for example, in Figures 7 and 8, panels D) and G), the population rapidly disappeared (all available resources were consumed) even if there were considerably more Conscious agents than Selfish agents. These results indicate a potential tipping point for the number of agents and/or the ratio between conscious and selfish agents.

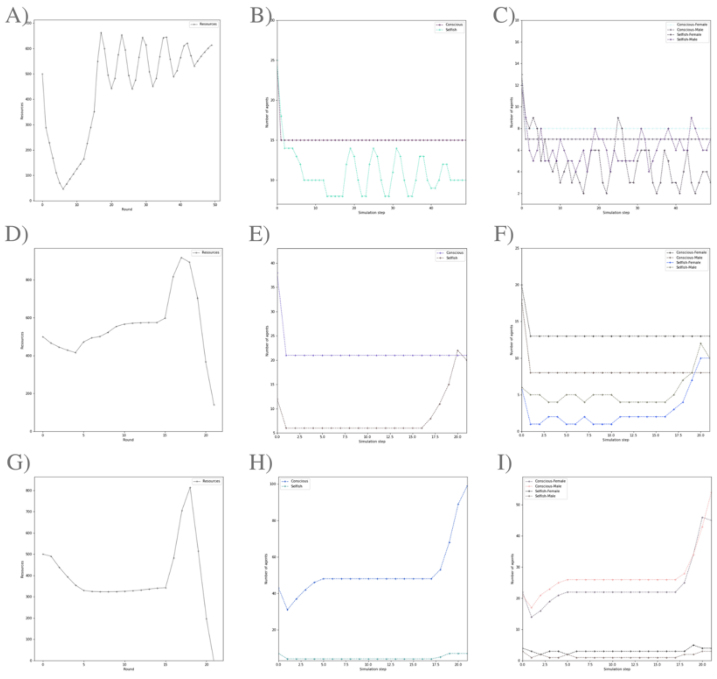


Figure 8. Three example simulations where agents are initialized with a guaranteed reproduction and random survival chance for the first round of the simulation. A) – C) Evolution of the population of 50 agents. Initial resources are set to 500, representing 50% of the maximum amount. Agents are split into 50% Conscious and 50% Selfish, with a 50% ratio of Female agents for each agent type. D) – F) Evolution of the population of 50 agents. Initial resources are set to 500, representing 50% of the maximum amount. More, agents are split into 75% Conscious and 25% Selfish, with a 50% ratio of Female agents for each agent type. G) – I) Evolution of the population of 50 agents. Initial

resources are set to 500, representing 50% of the maximum amount. Agents are split into 85% Conscious and 15% Selfish, with a 50% ratio of Female agents for each agent type.

5. Conclusions

This paper presented a Fuzzy Logic ABM framework implemented in Python to aid researchers in exploring complex systems. The framework builds upon established methods and extends them by implementing a Fuzzy Logic and Inference module. With the help of this module, vague concepts can be employed to create a set of rules guiding the agent's behavior, interactions, and environment evolution.

Moreover, a population dynamics simulation is presented to illustrate the proposed framework. This simple ABM consists of two agent types that consume, produce resources, reproduce, or die. A set of Fuzzy rules drives the agent's behavior and what the agent does depends solely on the available environmental resources. Although simple, the simulation yielded some exciting situations requiring closer investigation.

In conclusion, Fuzzy logic and inference represent a viable method for encoding an intelligent agent behavior and allowing machines to operate with vague definitions. However, this approach requires more testing and validation in various settings.

References

[1] Ebrahim II MAMDANI, S ASSILIAN – *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller* – International Journal of Man-Machine Studies, volume 1 no. 1 page 1-13. ISSN 0020-7373. January 1975

[2] Tomohiro TAKAGI, Michio SUGENO – *Fuzzy Identification of Systems and Its Applications to Modelling and Control* – IEEE Transactions on Systems, Man, and Cybernetics, volume SMC-15 no. 1 page 116-132. ISSN 0018-9472. January – February 1985

[3] Nigel GILBERT – *Agent-Based Models* – Pages 1-47, 68-74. ISBN 978-1-4129-4964-4. Sage Publications, Inc. 2008

[4] S. N. SIVANANDAM, S. N. DEEPA – *Principles of Soft Computing* – Pages 5-9, 251-264, 295-305, 311-320, 347-355, 356, 363-364. ISBN 978-81-265-2741-0. Wiley India Pvt. Ltd. 2011

[5] W. Brian ARTHUR – *Foundations of Complexity Economics* – Nat Rev Phys, volume 3, pages 136-145. ISSN 25225820. January 2021

[6] Jessica S. YU, Neda BAGHERI – *Agent-Based Models Predict Emergent Behavior of Heterogeneous Cell Populations in Dynamic Microenvironments* – Frontiers in Bioengineering and Biotechnology, volume 8:249 ISSN 2296-4185. June 2020

[7] Hossein AHMADI, Marsa GHOLAMZADEH, Leila SHAHMORADI, Mehrbakhsh NILASHI, Pooria RASHVAND – *Diseases diagnosis using fuzzy logic methods: A*

systematic and meta-analysis review – Computer methods and programs in biomedicine, volume 161, Pages 145-173. ISSN 01692607. April 2018

[8] Yong ZHANG, Weilong LIU, Xingyu YANG – *An automatic trading system for fuzzy portfolio optimization problem with sell orders* – Expert Systems with Applications, volume 187. ISSN 0957-4174. January 2022

[9] Robert AXELROD – *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration* – Pages 3-9, 40-68. ISBN 0-691-01567-8. Princeton University Press. 1997

[10] Jackie KAZIL, David MASAD, Andrew COOKS – *Utilizing Python for Agent-Based Modeling: The Mesa Framework* – Pages 308-317. ISBN 978-3-030-61255-9. Springer International Publishing. October 2020

Bibliography

von ALTROCK C., *Fuzzy Logic and NeuroFuzzy Applications in Business and Finance*. ISBN 0-13-591512-0. Prentice Hall Inc. 1997.

ARTHUR W. R., *Foundations of Complexity Economics*, Nat Rev Phys, volume 3, pages 136-145. ISSN 25225820. January 2021

AXELROD R. – *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration* – ISBN 0-691-01567-8. Princeton University Press. 1997

CEDERMAN L.-E., *Emergent Actors In World Politics: How States and Nations Develop and Dissolve*, ISBN 0-691-02149-X. Princeton University Press. 1997

GILBERT N., *Agent-Based Models*, ISBN 978-1-4129-4964-4. Sage Publications, Inc. 2008

JANG J.-S. R., SUN C.-T., MIZUTANI E., *Neuro-Fuzzy and Soft Computing A Computational Approach to Learning and Machine Intelligence*. ISBN 0-13-261066-3. Prentice-Hall Inc. 1997.

KARRAY F. O., DE SILVA C., *Soft Computing and Intelligent Systems Design: Theory, Tools, and Applications*, ISBN 0-321-11617-8. Pearson Education Limited. 2004

KAZIL J., MASAD D., COOKS A., *Utilizing Python for Agent-Based Modeling: The Mesa Framework,* ISBN 978-3-030-61255-9. Springer International Publishing. October 2020

MAMDANI E., ASSILIAN S., *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller*. International Journal of Man-Machine Studies, volume 1 no. 1 page 1-13. ISSN 0020-7373. January 1975

SIVANANDAM S. N., DEEPA S. N., *Principles of Soft Computing*, ISBN 978-81-265-2741-0. Wiley India Pvt. Ltd. 2011

TAKAGI T., SUGENO M., *Fuzzy Identification of Systems and Its Applications to Modelling and Control*. IEEE Transactions on Systems, Man, and Cybernetics, volume SMC-15 no. 1 page 116-132. ISSN 0018-9472. January – February 1985

YU J. S., BAGHERI N., *Agent-Based Models Predict Emergent Behavior of Heterogeneous Cell Populations in Dynamic Microenvironments*. Frontiers in Bioengineering and Biotechnology, volume 8:249 ISSN 2296-4185. June 2020