Multi-Agent Modelling Notation (MAMN): A multi-layered graphical modelling notation for agent-based simulations

Johannes Nguyen[1,2], Simon T. Powers[2], Neil Urquhart[2], Thomas Farrenkopf[1], and Michael Guckert[1]

[1] KITE, Technische Hochschule Mittelhessen, Friedberg, 61169, Germany
[2] School of Computing, Engineering and the Built Environment, Edinburgh Napier University, Edinburgh, EH10 5DT, UK

Abstract. Cause-effect graphs have been applied in non agent-based simulations, where they are used to model chained causal relations between input parameters and system behaviour measured by appropriate indicators. This can be useful for the analysis and interpretation of simulations. However, multi-agent simulations shift the paradigm of chained causal relations to multiple levels of detail and abstraction. Thus, conventional cause-effect graphs need to be extended to capture the hierarchical structure of causal relations in multi-agent models. In this paper, we present a graphical modelling method that we call *Multi-Agent Modelling Notation (MAMN)*, with which global aspects of the simulation as well as detailed interior mechanisms of agent behaviour can be described. We give proof of concept by showing how the logic that connects individual agent behaviour to global outcomes in a previously published simulation model can be expressed in a concise diagrammatic form. This provides understanding into what drives the model behaviour without having to study source code. We go on to discuss benefits and limitations as well as new opportunities that arise from this type of model analysis.*

Keywords: Cause-Effect Modelling · AOSE · Multi-agent Simulations.

1 Introduction and Motivation

The abstraction of a given real-world issue into a simulation model requires formalisation of causal relations and quantification of determining factors. Defining input variables and configuration parameters is an integral part of the modelling process. Multi-agent simulations try to recreate global system characteristics by modelling individual agents. System behaviour, which is often mirrored into output variables (key performance indicators), depends on both the modelled input as well as the internal mechanisms and dynamics of agent decisions. Crucially, direct and indirect consequential effects are the outcome of calculations according to mathematical functions and formulas expressed in the model. Chaining

*This research was supported by the Karl-Vossloh-Stiftung (S0047/10053/2019).

these calculations and corresponding intermediate variables reveals causal relations between input and output variables. These relations are complicated, but visualisation in a computational graph can make the internal mechanism of the model more accessible. Cause-effect graphs have already been applied for visualising aspects of simulation models [10, 12]. These approaches differ in the type of cause-effect relations modelled in the graphical representation, e.g. [12] focuses on modelling cause-effect relations between abstract events, rather than the computational aspects of key performance indicators. However, cause-effect relations between input parameters and performance indicators are of particular interest for policy-making. Current variations of cause-effect graphs typically model relations between variables on a common level of abstraction but the application of agent methods changes the paradigm from modelling chained causal relationships to multiple levels of detail and abstraction. This implies that the use of graphs working on a single level of abstraction is not appropriate and that a hierarchical approach separating individual and global perspectives would be more suitable. The semantics of a graphical notation needs to capture the internal aspects of individual agents, i.e. preferences and their decision-making behaviour, as well as their context in the computation of performance indicators at the global system level. In this paper, we propose a novel graphical modelling method that captures the hierarchical structure of causal relations between input and output variables in agent-based traffic simulations. This improves the transparency of agent-based models, by allowing the causal connections from input parameters and agent action selection functions to the result variables of the model to be clearly expressed in a manuscript. We establish a set of notation elements and define rules to represent the main logical constructs commonly used to simulate agents in route choice scenarios. Although code generation is typically a natural application for this type of graphical specification, our interest lies in finding an appropriate graph structure to improve the analysis and validation of agent-based simulations. In scope of this paper, we focus on collecting the necessary requirements for capturing the hierarchical structure of cause-effect relations in agent-based traffic simulations before further elaborating in subsequent work on how this graphical method can be leveraged for the analysis and validation of real applications. As we progress with tool implementation, we intend to assess which meta-model is best suited as a reference for this type of modelling.

The following section provides an overview of related work and discusses capabilities and scope of related modelling methods. Following this, in Section 3 we introduce a new set of notation elements and define rules for our proposed graph structure. In Section 4, we demonstrate usage of our graphical notation for representing a published simulation model from the traffic domain (see [8]). This allows us to represent the core logic that connects individual agent behaviours to global performance indicators in a diagrammatic form that fits in a manuscript. This avoids readers having to look at source code or pseudocode to try and uncover the connections. We then discuss how this can be used to improve analysis of multi-agent simulations. Finally, in Section 5 conclusions are drawn and possible options for future work are indicated.

2 Related Work

Cause-effect graphs have previously been used for a number of purposes, including software testing [15], system dynamics models [14], and for management tools [11]. They are an explicit and precise formalisation of logical systems and serve as a compact visualisation. Cause-effect graphs have been used in software testing to specify test cases for combinations of input and output variables [15]. Input variables define causes, while effects are represented as output variables. A specific variation of cause-effect graphs are causal loop diagrams which is used in system theory to model mutual effects between variable entities, e.g. mutual influence between predators and prey in an ecological system (see [10]). The system theoretical view of reducing complexity of information from reality to formal systems is an essential prerequisite for building computable simulation models. Building richer simulation models typically involves modelling of more system variables. Thus, observed effects are not a direct consequence of a single variable but of multiple causative variables (or chains of variables). *Bayesian networks* are an example of cause-effect graphs that allow output variables to be linked back to possible (chains of) input causes based on probabilities. However, applying cause-effect graphs to agent models has been difficult due to causal relations being emergent result of behavioural patterns of a large set of individuals which changes the paradigm from chained causal relations to several levels of detail and abstraction.

Visualisation of multi-agent systems has focused primarily on system design by extending traditional methods from software engineering (e.g. [6, 16]). Some of these methods have been implemented as tools to guide the software developing process of multi-agent systems [5, 9] and even dealt with system design from a more behavioural perspective [4, 17]. However, these modelling methods have a primary focus on the technical design of software components, rather than the cause-effect relations of performance indicators in a simulation model. In this paper, we want to focus on exactly this type of causal relations between input parameters and performance indicators on the social-behavioural level as these are particularly relevant for policy-making. Another type of visualisation that has been applied to computer-based simulations are event graphs [12]. This type of graph focuses on modelling the relations between abstract events, which is a concept from *complex event processing*. [7] defines an event as a record of an activity in a system which is linked to other events by aggregation, time, or causal conditions. The aggregation of events into different levels of detail has led to the extension of event graphs with a hierarchical structure [13]. However, aggregation into different levels of abstraction in event processing is different from what is required for modelling causal relations of performance indicators in multi-agent systems. Global performance indicators are emergent results of the decisions of a large set of autonomous individuals, which in the graph leads to changing aggregation mechanisms depending on the context of decisions and the simulation scenario. Hence, there is a need for a new modelling method that is specifically designed for modelling the different levels of abstraction for causal relations of performance indicators in multi-agent systems.

3 Method

We have developed a new graphical notation to model the hierarchical structure of cause-effect relations between input and output variables in multi-agent simulations. We focus on modelling the main logical constructs commonly used to simulate agents in route choice scenarios. Balke and Gilbert [1] have given an overview of established agent architectures used in literature for modelling decision behaviour. For this work, we focus our method on modelling agent behaviour according to the commonly used *Belief-Desire-Intention (BDI)* model based on [3]. The BDI model allows internal agent aspects to be abstracted into separate mental-levels [3] providing a uniform basis for the comparison of agent behaviour [2] and thus facilitates the analysis of simulations. A distinctive property of multi-agent simulations is their focus on the modelling of individuals and their actions. Output variables in such simulations typically describe the system behaviour using performance indicators on the global level, whereas input variables model internal details of agents on the individual level. Based on this, we model the different levels of a simulation as separate graphs and establish a modelling method that allows for their conjunction through appropriate notation elements. Formally, let $G = (V, E)$ be a directed labeled graph with vertices V and edges E. $V = N \cup F$ is a heterogeneous set of vertices with N the set of `variable` nodes and F the set of `functional` nodes. Vertices $v \in V$ are modelled using geometric shapes (see Figure 1). In particular, vertices $n \in N$ serve as variables that contain either `primitive` (rectangles) or `complex` information (circles). This type of vertex is used to define input parameters and performance indicators of the simulation, as well as relevant intermediary variables that are produced during the computation of performance indicators. The outline of these variable nodes $n \in N$ indicates whether n is a `single` variable (solid) or a `collection` of variables (dotted). Vertices $f \in F$ are used to model aggregations of functional sequences as well as termination of the simulation. For example, route selection of traveller agents can be complex and is usually implemented using established, externally implemented algorithms, e.g. Dijkstra's algorithm or A*. As we are interested in modelling cause-effect relations of input and output variables, functional nodes $f \in F$ serve as an abstraction of the implemented logic used to compute output variables. This abstraction allows cause-effect relations between variables on the same level to be modelled as input/output chains while at the same time details of functions are shifted to a sub-level as a separate graph. Functional sequences that are moved to a sub-level are indicated using a `step-into conjunction` node (downward pointing triangle) on the upper level. Algorithms or mathematical formulas for computing intermediary or output variables are modelled as an ellipse using the `function` node. In addition to this, we have added one more notation element (hexagon) to model an `iterative loop` with a termination clause. A special type of nodes in N and F are *mental-level* nodes N_M and F_M which are based on the basic BDI model [3] and are used to represent internal aspects of agents. `Beliefs` are variables N_M that contain information about the current internal state of an agent as well as perceived information about its surrounding environment. This

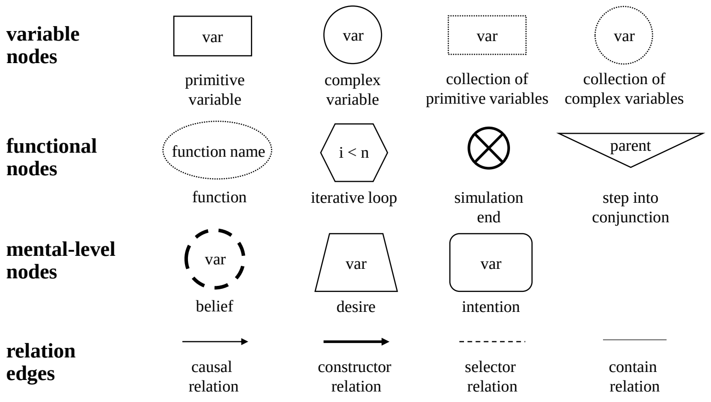

Fig. 1. Notation elements.

is modelled using a circle element with a bold dashed border. In addition to this, `desires` and `intentions` are functional nodes F_M that model agent behaviour. `Desires` define goals of an agent to maintain or achieve a certain state. In the context of mobility, this can also be referred to as *travel purpose*. Mobility of individuals typically is a necessary means for pursuing personal objectives, such as travelling to work or going to shop for groceries. In our notation, we model these desires using a trapezium. To achieve a desired goal agents have to perform actions or a series of subsequent actions. In the BDI model, this is referred to as `intentions`. We model this in our notation using a rectangle with rounded corners. Vertices of a graph are linked through edges $e \in E$ which are pairs (v_1, v_2) with $v_1, v_2 \in V$. Furthermore, $E = E_{Causal} \cup E_{Constructor} \cup E_{Selector} \cup E_{Contain}$ (see Figure 1). Edges $e \in E_{Causal}$ define `causal relations` for which v_2 is causally dependent on v_1. This type of edge can only link variable nodes and functional nodes in alternation i.e. if v_1 is a variable node $v_1 \in N$, then v_2 must be a functional node $v_2 \in F$. In this case, v_1 can be interpreted as input to v_2. Otherwise, if $v_1 \in F$, then v_2 must be a variable node $v_2 \in N$ which is computed by v_1. Causal relations with v_1 being a `collection` node $v_1 \in N$, are defined as *for each* relations, meaning that for each item $i \in v_1$ a functional sequence $v_2 \in F$ will be executed. Apart from this, edges $e \in E_{Constructor}$ can be used to model a `constructor relation` in which v_2 is created from v_1. In this case, v_1 must be a functional node $v_1 \in F$ that results in a variable node $v_2 \in N$. Edges $e \in E_{Selector}$ model a relation in which one item is being `selected` from a collection which serves as input to a function. Thus, v_1 must be a `collection` node $v_1 \in N$ and v_2 a functional node $v_2 \in F$. Finally, the last type of edges $e \in E_{Contain}$ defines a relation in which a complex variable node $v_1 \in N$ `contains` the information of $v_2 \in N$. In the case that v_1 is a `collection` of complex variables, e represents a *for each* relation, meaning that each item in v_1 holds its own information v_2. This concludes definitions for our proposed graph structure. In the next section, we give an example of how this can be applied to a simulation model from the traffic domain.

4 Use Case

As an example, we discuss the MAMN graph for the simulation model presented by the authors in [8]. This model is agent-based and was designed for measuring environmental impact of traffic caused by individuals and their travel behaviour. From a global perspective, input parameters of the simulation determine population size, potential home and supermarket locations as well as persona for agent characteristics on the individual level. The causal relations between these input parameters, agent behaviours, and key performance indicators such as average distance travelled are expressed in ca. 119.100 lines of source code. We now show how these can be expressed graphically in a succinct manner using MAMN. Based on notation elements presented in Figure 1, population size is modelled as a `primitive variable` whereas relevant locations and agent persona are `collections of complex variables` (see Figure 2). Relevant details of complex variables such as the attributes of agent persona (age, gender, etc.) are modelled as variable nodes and linked to the persona node using a `contain relation`. Information from input parameters is used to generate a population of shopping agents. This process is modelled using an `iterative loop` as a functional node with an outgoing `creator relation`. The resulting population of shopping agents can be represented as a `collection of complex variables`. Shopping agents in the simulation follow the BDI model and therefore details about their knowledge and behaviour are mapped to the corresponding mental-level nodes. Agents are assigned a persona and a start location as `beliefs` which is specified using the `contain relations`. Furthermore, agent behaviour is motivated by their purpose of travel to purchase groceries which is expressed using a `desire` node together with the associated `contain relation`. In order to satisfy their desire, agents perform a series of actions (`intentions`) initiated by `causal relations`. Agents determine modes of travel and supermarkets to be visited in *pre-journey planning*. This information is used to create a *journey* which is modelled using a `constructor relation` that results in a `complex variable`. Details about decision processes in *pre-journey planning* are moved to a separate graph (see Figure 3). This is indicated on the top-level using a conjunction

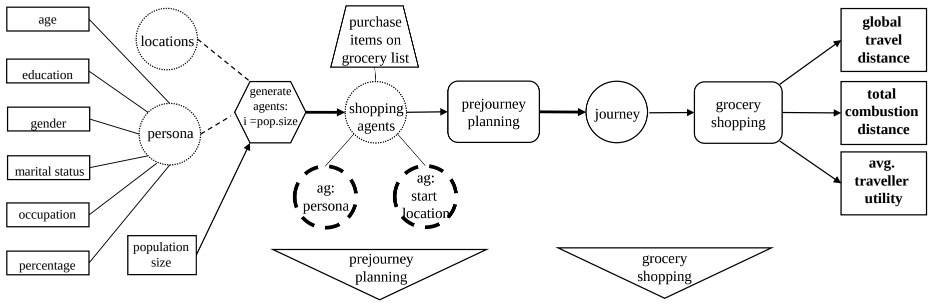

Fig. 2. MAMN graph from the global perspective.

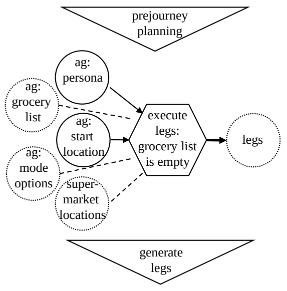

Fig. 3. MAMN graph for pre-journey planning.

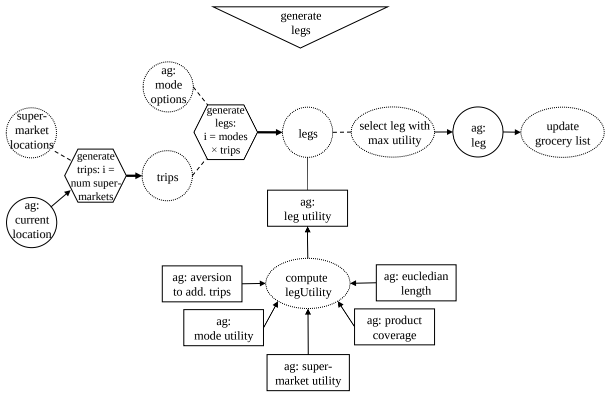

Fig. 4. MAMN graph for processing the shopping list.

element positioned at the bottom of the graph (see Figure 2). Agents can then use information from pre-journey planning to go grocery shopping. Again, details of this activity are moved to a separate graph (see Figure 5). Performance indicators of the system are obtained based on movements of agents during their grocery shopping. For example, environmental impact is measured using performance indicators on aggregated travelled distances. *Global travel distance* measures the sum of total distances travelled by all agents whereas *combustion distance* only considers distances caused by modes of travel that produce exhaust fumes. Another indicator included in the simulation model refers to the *average traveller utility.* This indicator is used during the experimentation to measure effects of different traffic policies on individuals. The simulation model therefore comprises these three output variables that can be modelled as `primitive variable` nodes. Functional nodes *pre-journey planning* and *grocery shopping* have been modelled as separate sub-level graphs. *Pre-journey planning* produces a `collection` of *legs* which determines a planned *journey* (see Figure 3). A *leg* is the combination of a *mode of travel* and the *next supermarket* [8]. In particular, agents determine legs based on their *persona, start location, surrounding supermarkets, mode options* and items on their *grocery list.* The process of generating the *collection of legs* requires a functional node with an outgoing `constructor relation`. Agents may need to process several legs until all items on their grocery list are purchased. Thus, we use an `iterative loop` with a termination clause depending on remaining items on their grocery list. Details of this loop are once more moved to a sub-level (see Figure 4). In the sub-level, the agent first computes the set of all possible legs L for which applies $L = T \times M$ with M the set of available mode options and T the set of potential trips. L, M and T can therefore be modelled as `collections of variable nodes`. A trip $t \in T$ is a combination of a start location and a destination. Thus, T is the Cartesian product of the current location of the agent and the set of unvisited supermarkets. Furthermore, for each leg $l \in L$ a leg utility $u_a(l)$ is computed based on attributes of agent a and the leg. The agent then chooses a leg with maximum

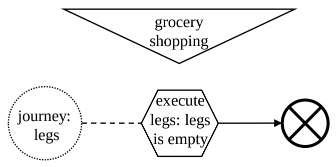

Fig. 5. MAMN graph for grocery shopping.

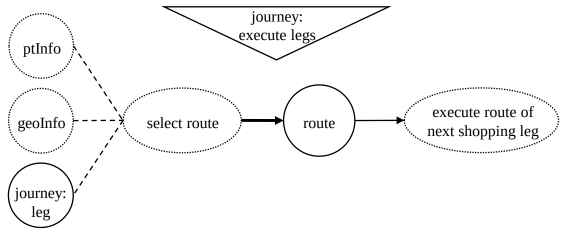

Fig. 6. MAMN graph for route selection.

utility and updates its grocery list. Computation of the leg utility as well as the selection process are both modelled using `function nodes`. In this context, `function nodes` are the abstract representation of mathematical formula. This concludes the graph structure for *pre-journey planning*. Grocery shopping can be modelled as an `iterative loop` that processes all legs from the planned journey (see Figure 5). The simulation is terminated upon completion of this loop. Analogous to the previous examples, details of this loop are modelled in a separate sub-level graph (see Figure 6). The agent uses information on timetables for public transport as well as geographic map data to determine a route for the leg to be processed and then repeats the process for the next leg. Formalisation of cause-effect relations between input and output variables as an MAMN graph serves as a compact representation of the simulation model. This allows complex scenarios to be presented in a transparent and comprehensible manner that can be presented inside a manuscript. In particular, the representation as MAMN graphs offers insight into simulation models from a *social-behavioural* perspective, which is currently not supported by existing modelling notations and visualisations such as UML-based approaches as they focus on a more *technical* view of the simulation (e.g. system design). Event graphs may be positioned somewhere in between the *social-behavioural* and *technical* view, but are not specialised to model the hierarchical structure of cause-effect relations between input parameters and output indicators in multi-agent simulations. However, the social-behavioural perspective together with a focus on cause-effect relations is highly relevant for transportation planners using simulations based on individual traffic participants. Thus, with our MAMN method we address the lack of a modelling technique for the social-behavioural perspective of agent-based simulation models. Leveraging this, there are new opportunities to improve the development and evaluation process of agent-based simulations. Graph structures can be utilised in a bi-directional process to either transfer a theoretical simulation model into a concrete implementation as an executable piece of code (*forward engineering*) or to visualise information from a given implementation (backward engineering) which can be used to increase transparency and explainability of a system. As stated previously our interest lies in the second manner for which we establish MAMN as a basis for subsequent work on building tools for validation and analysis of agent-based simulations.

5 Conclusion and Future Work

Cause-effect graphs can provide insight into how simulation output is computed, but have previously lacked the concepts to capture the hierarchical structure of cause-effect relations in multi-agent simulations. In this paper, we have presented a new graphical method to model cause-effect relations from input parameters on the individual level to performance indicators at the global system level. This is achieved by shifting details of functional relations to a sub-level as a separate graph and through the use of appropriate conjunction elements. For future work, we will improve our graph by adjusting it to an appropriate meta-model and work on tools to improve analysis and validation of agent-based simulations.

References

1. Balke, T., Gilbert, N.: How do agents make decisions? a survey. Journal of Artificial Societies and Social Simulation **17**(4) (2014)
2. Brafman, R., Tennenholtz, M.: Modeling agents as qualitative decision makers. Artificial Intelligence **94**(1-2), 217–268 (1997)
3. Bratman, M., Israel, D., Pollack, M.: Plans and resource-bounded practical reasoning. Computational Intelligence **4**(3), 349–355 (1988)
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems **8**, 203–236 (2004)
5. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: Aspecs: an agent-oriented software process for engineering complex systems. Autonomous Agents and Multi-Agent Systems **20**(2), 260–304 (2010)
6. Gonçalves, E., Cortés, M., Campos, G., Lopes, Y., Freire, E., da Silva, V., de Oliveira, K., de Oliveira, M.: Mas-ml 2.0: Supporting the modelling of multi-agent systems with different agent architectures. Journal of Systems and Software (2015)
7. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley (2002)
8. Nguyen, J., Powers, S., Urquhart, N., Farrenkopf, T., Guckert, M.: Modelling the impact of individual preferences on traffic policies. SN Computer Science **3** (2022)
9. Padgham, L., Winikoff, M.: Prometheus. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 1 - AAMAS '02. pp. 174–185. Springer, ACM Press (2002)
10. Schaub, H.: Simulation als entscheidungshilfe: Systemisches denken als werkzeug zur beherrschung von komplexität. Entscheiden in kritischen Situationen (2003)
11. Schoeneborn, F.: Linking balanced scorecard to system dynamics (2003)
12. Schruben, L.: Simulation modeling with event graphs. Communications of the ACM **26**(11), 957–963 (1983)
13. Schruben, L.: Building reusable simulators using hierarchical event graphs. In: Winter Simulation Conference Proceedings, 1995. pp. 472–475. IEEE (1995)
14. Sterman, J.: Business dynamics. McGraw-Hill, Inc. (2000)
15. Ufuktepe, D., Ayav, T., Belli, F.: Test input generation from cause–effect graphs. Software Quality Journal pp. 1–50 (2021)
16. Wagner, G.: The agent–object-relationship metamodel: towards a unified view of state and behavior. Information Systems **28**(5), 475–504 (2003)
17. Wooldridge, M., Jennings, N.R., Kinny, D.: The gaia methodology for agent-oriented analysis and design. Autonomous Agents and multi-agent systems **3** (2000)