university of groningen / faculty of science and engineering

A LOGICAL APPROACH TO UNIFY AND TRANSLATE DIFFUSION MODELS IN A MORE GENERAL FRAMEWORK

Bachelor's Project Thesis

Konstantin Rolf, s3750558, k.rolf@student.rug.nl,
Supervisor: Dr Z. Christoff

Abstract: Social influence has been a topic of increasing importance in the rise of social networks like Facebook or Instagram. Especially the propagation of opinions in such networks has been topic of interest. Different opinion diffusion models tried to capture this behavior. Additionally, epidemiology uses very similar models to look at the spread of diseases in epidemics. This study investigates diffusion models of different origins and combines them using a deterministic finite state machine (DFA). A special translation is shown that converts a weighted and biased threshold model to an undirected and unweighted threshold model. To make diffusion models more approachable, a website tool is implemented that shows the behaviors of multiple diffusion models for common types of graph typologies. The website is also capable of showing the previously mentioned model translation.

1 Introduction

The diffusion of beliefs, trends, behaviors or information has become a major research field in the past three decades. The fast growth of the internet, and especially social media, led to a rapid decrease in communication time. One of the most interesting characteristics during this development was the emergent behavior of so-called *viral events* (Pöyry, Laaksonen, Kekkonen, and Pääkkönen (2018)). The analogy came from the research in epidemics and describes the exponential spreading of virus infections through the population. Viral events have very similar characteristics by achieving exponential growth rates across the internet community. A single image or video can spread through the internet in days and reach millions of people. The recent development of algorithms by platforms like YouTube favored this development by introducing a 'trending' section that features videos that may become viral. As a result, many researchers have become interested in the dynamics of these events and tried to model them by using the approaches from epidemiology. The *Susceptible-Infected-Susceptible* (SIS) and *Susceptible-Infected-Recovered* (SIR) models are commonly used to model the behavior of diseases. The key difference between those models is that the SIR model assumes that agents become immune after infection while the SIS model allows multiple infections.

Researchers in social influence developed very similar techniques to model the decision-making process of individuals inside social groups. DeGroot (1974) proposed one of the first opinion diffusion models that simulates the influence in a group by a trust matrix that works analogous to a Markov Chain. Another type of opinion diffusion model are threshold models which try to model social influence through thresholds that determine when it is beneficial for a person to change its opinion (Christoff and Grossi (2017); Baltag, Christoff, Rendsvig, and Smets (2019); Easley and Kleinberg (2010)). Liu, Seligman, and Girard (2014) created a similar model that uses a three-state automaton to model the behavior with an additional neutral state. Another type of model was defined by Grandi, Lorini, and Perrussel (2015) who created two different versions of propositional opinion diffusion models: The majority decision model adopts a state if the agent observes a clear majority among its friends and its own opinion. The unanimity decision model adopts a state if the agent observes a unanimous consens among its friends.

The presented models which originated in the research fields of epidemiology and opinion diffusion are very similar in the sense that they are modelling the propagation of events though a network. On the other hand, they use very different implementations and model architectures to achieve this. This thesis tries to answer the question in which way these models are interchangeable in representing the same problem domain. Can those diffusion models be converted to each other or is it possible to unify them in some kind of 'super model'? Section 2 starts by explaining the models in detail and show the underlying assumptions that are made by each model. Section 3 considers a generalization of the models in form of a deterministic finite automata (DFA). Section 4 defines a translation between a weighted uni-directional threshold model and a uni-directional threshold model. Additionally, a tool is implemented that is able to simulate the models and study their characteristics in different scenarios.

2 Different Types of Models

The introduction already mentioned some models that are used in epidemiology and opinion diffusion. This section aims at giving a general overview of these models and their assumptions. More specifically, the section will provide each model a definite name that is used in the later sections to unify and translate the model definitions. Diffusion models can generally be classified by the graph over which they are defined (weighted/unweighted; directed/undirected), by the amount of states the models requires, and by some kind of transition function to update the model in discrete time intervals. The following section will showcase the models that are present in literature and also present slight modifications and generalizations of those models. This section describes the models in a more qualitative way. Section 3 will give more formal definitions of the models.

Threshold Models $M_{\textbf{t-BI}}$ and $M_{\textbf{t-UNI}}$: Threshold models were introduced as a way to model the propagation of beliefs and opinions in social networks (Easley and Kleinberg (2010)). They are defined as an unweighted undirected graph. Each agent is influenced uniformly by its direct neighbors. Threshold models are mostly described with a binary state (agents may either follow the belief or not). However, it is also possible to extend the model definition to include an arbitrary finite amount of states (a possible extension is proposed by Grandi et al. (2015) who implement a more general type of threshold model with any amount of states). This section will only consider binary threshold models with two states that will be called P and $\neg P$. Therefore, every agent propagates its state to all of its direct neighbors. An agent adopts a new belief if the amount of all its neighbors, following this belief, is greater equal than a predefined threshold. This update policy may be defined in a uni-directional way: Nodes will only change from $\neg P$ to P. However, it can also be defined in a bi-directional way in which nodes may also change from P to $\neg P$ if the threshold statement is not fulfilled. Both types of threshold models are fully deterministic with a discrete set of two states.

Uni-directional threshold models have some interesting characteristics: Once adopting a new belief, an agent will keep his belief because there is no way an agent could change back to its original belief. Hence, beliefs propagate through the network until a stable configuration is reached. It is possible to make a formal definition that describes whether a so-called cascade will reach all nodes. A cluster of density p containing a set of nodes is defined such that each node has at least a fraction of p neighbors in the set. The definition of a cascade is extended such that a cluster of size $1 - p$ stops any cascade and that a cascade of $1-p$ is the only reason that a cascade is stopped (Easley and Kleinberg (2010)). On the other hand, bi-directional threshold models may never reach stable configurations because the update policy may allow agents to change its belief multiple times. The most simple configuration for such a model would be a model with threshold 0.5 and two nodes that are connected to each other with different initial states. Such a model would periodically fluctuate between two configurations with opposite states.

In the following, the uni-directional threshold model will be denoted by $M_{\text{threshold-UNI}}$ or simply as $M_{\text{t-UNI}}$. Similarly, the bi-directional threshold model will be denoted by $M_{\text{threshold-BI}}$ or simply as $M_{\text{t-BI}}$. Special instances of the threshold model will be defined as $M_{\mathrm{t}}(t)$ where t determines the global threshold. Conclusively, $M_{\text{t-BI}}(0.5)$ indicates a bidirectional threshold model with global threshold of 0.5.

Weighted Threshold Models $M_{\textbf{wt-BI}}$, $M_{\textbf{wt-UNI}}$: The definition of a simple undirected and unweighted threshold model $M_{\text{t-BI}}$ or $M_{\text{t-UNI}}$ can be extended to a more general case of weighted and directed threshold model $M_{\text{wt-BI}}$ or $M_{\text{wt-UNI}}$. The previous definition assumed equal influence across all neighbors. The requirement can easily be removed by allowing any positive weight with the constraint that the sum of the incoming weights adds up to one. Similarly, it is possible to remove the assumption of a global threshold and turn it into a node-specific threshold. Lastly, the graph must be transformed into a directed graph. The definition of an undirected graph in the original threshold model $M_{\text{t-BI}}$ or $M_{\text{t-UNI}}$ can be implicitly seen as a directed graph already because the nodes value incoming connections differently depending on the amount of incoming nodes. The transition function works exactly the same as an unweighted threshold model except that each neighbor node is valued according to the weight of its edge with a node-specific threshold.

DeGroot Models $M_{\textbf{deg}}$: DeGroot models were proposed by DeGroot (1974) in 1974 and tried to apply the findings in stochastic theory and specifically Markov Chains to opinion diffusion. The DeGroot model is a time-homogeneous Markov Chain that models social decision-making processes by the usage of a stochastic trust matrix T that represents how much agents value the opinions of other agents. Therefore, each agent may be influenced from every other agent with a unique weight. The state of the model at time point t is represented as a vector p_t containing one value in the range $[0, 1]$ for each agent. A single discrete time step may be computed by multiplying the trust matrix with the vector p_t holding the current state: $p_{t+1} = M * p_t$. The limit of the trust matrix M ($\lim_{n\to\infty} M^n$) can be used to compute the stable configuration for every initial agent configuration (if there is one). DeGroot models are fully deterministic with a continuous state. In the following, DeGroot models will be denoted by M_{deg}.

Threshold Automata $M_{\textbf{ta}}$: The threshold automata model was introduced by Liu et al. (2014) and is defined over an undirected and unweighted graph. The transition function uses a three-state automata with a set of four input symbols. Nodes may either follow the belief Bp, have a disbelief $B\neg p$, or follow no belief $\neg Bp \wedge \neg B\neg p$ which is abbreviated as Up. The input symbols are defined as weak and strong influence in both directions to model the influence from neighboring nodes. A node is strongly influenced to believe in p if and only if all of its neighbors and at least one neighbor are believing in p (Liu et al. (2014) assume a threshold of 100% for their model). The same applies to strong influence in $\neg p$. A weak influence is given when at least one of its neighbors believes in p and the amount of agents disbelieving in p is not greater than a certain threshold (conservatively 0% is assumed). The same applies to weak influence in $\neg p$.

The finite state machine uses a simple transition table to update the state according to the axioms of influence. A strong influence in p or $\neg p$ causes the state to switch to Bp or $B\neg p$ respectively. A weak influence in p and a missing strong influence in p or $\neg p$ causes the state to switch from Bp to Up. Similarly, a weak influence in $\neg p$ and a missing strong influence in $\neg p$ causes the state to change from $B\neg p$ to Up. The finite state machine described by Liu et al. (2014) can be seen in Figure 2.1. Models may reach a stable limit distribution or may also fluctuate periodically between states. The threshold automata model will be denoted as M_{ta} from here on.

Probabilistic Models $M_{\textbf{sir}}$, $M_{\textbf{sis}}$: The probabilistic models, namely the SIS and SIR models, were invented to model the propagation of infectious diseases through the population (Easley and Kleinberg (2010)). Both models are defined as an unweighted and directed graph with the only difference in the amount of states. The SIS model contains the *susceptible* (S) and *infectious* (I) state. The SIR model contains a third state which is called *removed* (R). The general behavior of the model may be described by three simple rules that distinguish the current state of the node (Easley and Kleinberg (2010)):

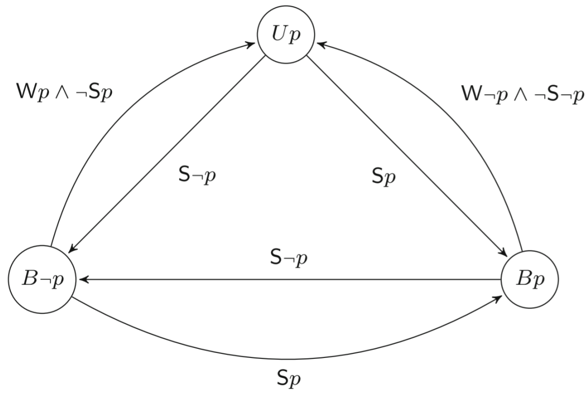


Figure 2.1: Transition table of a finite state machine by Liu et al. (2014).

- Susceptible: The node does not have the disease, but may be infected at any future time step.
- Infectious: The node has the disease. It stays in this state for a fixed number of time steps. It may infect any susceptible neighbor node with a probability of p during every time step.
- Removed: After the node was infected, it will either become susceptible again (SIS model) or gain resistance to all future infections (SIR model) which is the same as removing the node from the graph.

The behavior of diseases in these models may be classified by the R factor. It describes how many other nodes are on average getting infected by an infected node. Diseases that have a factor of $R < 1$ are gradually dying out. SIS models with a factor $R \geq 1$ can sustain themselves. Any disease in the SIR model will eventually die out because there is only a finite amount of nodes that can be infected. The SIS model will be denoted by $M_{\text{sis}}(p, t_i)$ and $M_{\text{sir}}(p, t_i)$ where p gives the probability of an infection spreading to a neighbor node and t_t gives the time the node stays infected.

Propositional Opinion Diffusion $M_{\textbf{maj-POD}}$, $M_{\textbf{u-POD}}$: These type of models were created by Grandi et al. (2015) and are based on the research in binary opinion diffusion models and judgement aggregation theory. The models originated from a different background, but the resulting models are related to the threshold model $M_{\text{t-BI}}$. The model is defined as an undirected unweighted graph with an arbitrary amount of states. At each discrete time step the agents aggregate all of their neighbor opinions (voting) and change their own belief according to the transition function. Grandi et al. (2015) define the majoritarian propositional opinion diffusion (maj-POD) to update the agent's belief when it notices a strict majority in the beliefs of its neighbors including its own opinion. They also define the unanimous rule (u-POD) to update the agent's belief if and only if all of the agent's influencers share the same opinion. The model by majority model will be denoted by $M_{\text{maj-POD}}$ and the unanimity model by $M_{\text{maj-POD}}$.

2.1 Assumptions

Table 2.1 generalizes the assumptions that are made by the presented models. The assumptions of each model can be presented as a list of the factors: deterministic (the outcome can be solely determined by the initial configuration), weighted (the edges of the graphs are valued), reflexive (the current state of the node directly influences the next state), irreflexive (the next state is independent of the current state), equal influence (all incoming edges to a node are valued equally in the transition function), states (the amount of states that the model uses). The usual definitions of reflexiveness and irreflexiveness are defined on the basis of recurrent edges in a graph. This thesis uses a more abstract approach that defines them on the fact that the current state has an influence on the next state. For example, the SIS and SIR models are used in an irreflexive graph. However, it is not possible to determine the next state of a node by only looking at the neighbor nodes because the node behaves differently depending whether it is infected, susceptible or recovered. This means that there is some kind of implicit reflexive connection in the SIS and SIR model that makes the next state dependent on the current state.

2.2 Jargon

Literature (Easley and Kleinberg (2010); DeGroot (1974); Baltag et al. (2019); Christoff and Grossi (2017); Grandi et al. (2015); Liu et al. (2014)) uses interchangeable terminology to describe diffusion models. This paragraph aims at providing a unified jargon that will be used in the following sections. The network of nodes that is frequently described as *network*, *social network*, *agent group* is used as graph. The graph's entities, which may be described as *nodes*, *agents* or *actors*, will be denoted as nodes. The entities that connect two nodes together will be described as edges. The state of an agent, which is frequently described by *opinion*, *(social) belief*, *information* or *status*, will be called state.

3 Unified Model

3.1 Automata Theory

This section will consider the similarities between the models and describe how they can be seen as subsets of a more general kind of model. The following paragraph will focus be on the definition and transition functions of a single node. The goal is finding a general transition function that is capable of modelling the transition functions of all diffusion models defined in Table 2.1. The first step towards a more general function is classifying the transition function of each model by the minimum amount of computational complexity needed to simulate them. The automata theory distinguishes between four different types of machines that grow in computing power and capabilities: Combinational Logic (CL), Finite-State Machines (FSM), Pushdown Automatons (PD) and Turing Machines (TM).

Table 2.1: Characteristics of all diffusion models: (1) deterministic, (2) weighted, (3) reflexive, (4) irreflexive, (5) equal influence, (6) states, (Y) Yes, (N) No, ($\mathbf{I}^1$) the model is implicitly weighted because equal influence may give an edge different weight depending on the amount of neighbor nodes, (K) variable number of states. See Section 2.1 for definitions of the assumptions.

	1	2	3	4	5	6
$M_{\text{t-BI}}$	Y	I^1	N	Y	Y	2
$M_{\text{t-UNI}}$	Y	I^1	Y	N	N	2
$M_{\text{wt-BI}}$	Y	Y	N	Y	N	2
$M_{\text{wt-UNI}}$	Y	Y	Y	N	N	2
$M_{\text{deg.}}$	Y	Y	N	N	N	$[0, 1]$
M_{ta}	Y	I^1	Y	N	Y	3
M_{sis}	N	I^1	Y	N	Y	2
M_{sir}	N	I^1	Y	N	Y	3
$M_{\text{u-POD}}$	Y	I^1	N	Y	Y	K
$M_{\text{maj-POD}}$	Y	I^1	Y	N	Y	K

Bounds: Each machine adds some computational power that allows accepting additional languages that cannot be accepted by a machine of lower class. Hence, the natural upper bound for a general model is a TM as it is the highest order of automata type. However, this is merely a useful definition as it just describes that each model may be simulated by a modern computer (a computer is assumed to have TM capabilities even though it is in principle bounded in its memory). A more convenient approach is finding the lower bound that is still able to represent all transition functions. The advantage in this approach is that the model does not get too abstract to make meaningful conclusions.

Combinational Logic: The first step is checking which models can be represented by the lowest class of automata (Combinational Logic). CL is a time-independent logic that uses the logical combinators of propositional logic to determine the state of the automata. Time-independence describes that the state of the automata is completely determined by the input. Hence, CL automatas are not capable of storing any state (or using the state to determine the next state) because this would violate the assumption of time-independence. Even with this restriction, it is still possible to build simple diffusion models which can be represented by CL automata. Examples for this are the $M_{\text{t-BI}}$ and $M_{\text{wt-BI}}$ models which are defined by a single equation that is independent of the state of the node. On the other hand, the $M_{\text{t-UNI}}$ and $M_{\text{wt-UNI}}$ models do not satisfy this requirement because they behave differently depending on the current state. Another example is the $M_{\text{u-POD}}$ model that checks for a unanimous consens in the neighboring nodes which is not dependent on the current state of the node. In general, the $M_{\text{t-BI}}$, $M_{\text{wt-BI}}$ and $M_{\text{u-POD}}$ models are the only models that can be completely represented by CL. More precise, all models that are classified as being irreflexive in Table 2.1 can be represented by CL because the definition of irreflexiveness captures the assumptions of time-independence in CL. All other models may require some kind of internal state (reflexive connections) to determine the next state. The 'may' is important in this case because certain edge cases of models are still reducible to CL. For example, a DeGroot model without reflexive connections is time-independent. However, the DeGroot model cannot be represented by CL generally.

Finite-State Machine: The next step is checking which transition functions can be represented by the next higher class of automata. FSMs are especially interesting because they are the lowest class of automata that may use internal state in their transition function. FSMs can be classified into the three different subcategories of deterministic finite automatons (DFAs), non-deterministic finite automatons (NDAs) and probabilistic automatons (PAs). DFAs are regularly defined as a five tuple $D = (Q, \Sigma, \delta, q_0, F)$ where Q determines the set of possible states; Σ determines the alphabet of possible inputs; δ determines the transition function that maps a state $q \in Q$ and an input symbol $\sigma \in \Sigma$ to the next state; $q_0 \in Q$ is the initial state; and $F \subset Q$ determines the set of accepting states. The definition of a DFA requires that each transition in δ is unique and that each transition must be triggered by an input symbol. A simple example of a DFA can be seen in Figure 3.1. NDAs are a more general case of DFAs where these restrictions are not given. That means that the transitions in δ do not need to be unique and that the system can change state without input symbols. PAs are an even more generalized version of NDAs where transitions between states can have specific probabilities. Following the generalizations, every DFA is a NDA and every NDA is a PA, but not the other way around. However, the additional generalization does not add any power to the machine. Any language that can be accepted by a PA or NDA can also be accepted by a DFA. That means that every NDA and PA can also be represented by a regular DFA. For example, any NDA can be represented by a DFA by applying the powerset constructions even though an n state NFA can result in a DFA with up to 2^n states (Rabin and Scott (1959)).

Liu et al. (2014) (the M_{ta} model) already used a three-state DFA in conjunction with quantifier logic

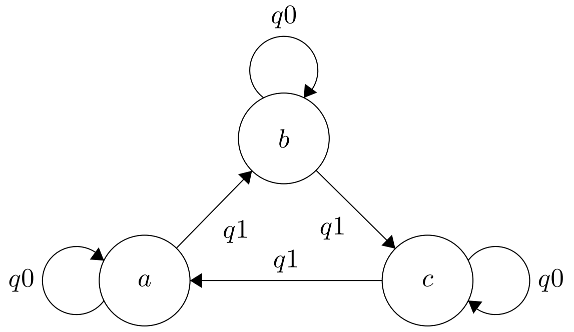


Figure 3.1: Example of a simple FSM that rotates between three states on input q_1 or repeats the current state on q_0.

to build a transition function for the nodes in their model. The input to the FSM are simple threshold rules that can be determined by the states of the neighboring nodes (Wp, $W\neg P$, Sp and $S\neg P$). Similarly, the $M_{\text{t-UNI}}$, $M_{\text{wt-UNI}}$ and $M_{\text{maj-POD}}$ models can be represented by DFAs in the same manner. This follows from the fact that each model has a finite amount of states where the next state can be solely determined by the current state and the inputs from the neighboring nodes. Also, all transitions are unique and all transitions are require an external input symbol. The M_{sir} and M_{sis} models do not fulfill this requirement because they have probabilistic transitions and can change infection state without external input symbols. However, they can still be represented by PAs which are generally reducible to DFAs.

Conclusion: In summary, all transition functions that were discussed so far can be represented by DFAs. The only exception to this is the DeGroot model which was not mentioned till now. The major problem of the DeGroot model is the continuous state space which is defined as: $Q = \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$. Hence, it is not possible to build a DFA because the system would need an infinite amount of states which violates one of the basic assumption of a DFA. However, it is still possible to represent DeGroot models as DFAs by relaxing the requirement of a perfect translation or unification. The infinite continuous state space can be represented by a finite amount of intervals. With intervals getting smaller, the amount of states needed to describe the DeGroot model increases to infinity. Non-perfect translations/unifications are not the priority of this thesis, which is the reason why they will not be discussed any further. Table 3.1 gives a summary of this section by listing the lower bound automaton for each model.

3.2 Unified Model M

So far, the section only considered the transition function of a single node. In Table 3.1 it was shown that the most powerful automata needed is a DFA (without considering the DeGroot model). If we use the definition of a DFA as general transition function and start looking at the full graph of nodes, the whole structure becomes very similar to the definition of a cellular automaton. A cellular automaton is a (possible infinite) set of DFAs which is ordered in a grid-like pattern where the single DFAs use their neighbors as input. A graph does not necessarily fulfill the definition of a grid-like pattern, but the collection of nodes connected by edges works similar to a set of interconnected DFAs that use their neighboring DFAs as input.

As the next step, a general model definition is given that will be applied to all diffusion models. The model is created as a five-tuple $M = (G, Q, Q_0, \Sigma, \delta)$ with $G = (V, E)$ as a graph containing a set of nodes $V = \{v_1, v_2, ...\}$ and a set of weighted edges $E = \{(v_1, v_2, w_1), (v_2, v_1, w_2), ...\}$. The graph is directed such that $(v_1, v_2, w) \neq (v_2, v_1, w)$. By definition, the graph will not contain two edges that differ only in the weight (the graph is not a multigraph). The second value in the model definition, Q, describes the set of finite states that is used in the DFA. The initial state of the model is described as Q_0 and is a vector containing the initial configuration such that each element of $Q_0 \in Q$. Σ gives the set of input symbols to the DFA. Lastly, the transition table σ describes a relation from the set of states and set of input symbols to other states.

Table 3.1: Lower bound automaton for each model: Combinational Logic (CL), Finite-State Machine (FSM), Pushdown Automaton (PD) and Turing Machine (TM). The M_{deg} model requires a superset of the DFA.

Model	Lower Bound
$M_{\text{t-BI}}$	CL
$M_{\text{t-UNI}}$	DFA
$M_{\text{wt-BI}}$	CL
$M_{\text{wt-UNI}}$	DFA
M_{sis}	PA = DFA
M_{sir}	PA = DFA
M_{ta}	DFA
$M_{\text{deg}}.$	DFA $\subset$
$M_{\text{maj-POD}}$	DFA
$M_{\text{u-POD}}$	CL

A few general functions will be defined to note special subsets of nodes and edges that will be commonly used in the next step. The functions $Source(e)$, $Destination(e)$ and $Weight(e)$ are used to access the members of the edge tuple $e = (v_s, v_d, w)$. The functions $in(v) = \{e | e \in E \wedge Destination(e) = v\}$ and $out(v) = \{e | e \in E \wedge Source(e) = v\}$ refer to the set of incoming and outgoing edges for a node v. The magnitude operator $|a|$ is used to count the number of members in a certain set. Therefore, $|in(v)|$ and $|out(v)|$ refer to the amount of incoming and outgoing edges from node v. These definitions are commonly called the 'indegree' and 'outdegreee' of node v. With this logic, it is already possible to define the requirements of the models in Table 2.1 in a more formal way:
Additionally, a helper function $State(v)$ will be defined that is used to access the state of node v. With that, it is possible to define a function that returns the set of all neighboring input nodes that follow a certain state. The function takes the node v and a state q as parameters and returns the set of all incoming connections that also have this state. Hence, $|ifx(v, \neg P)|$ would count the amount of neighbor nodes with incoming edges that have state $\neg P$.

$$ifx(v, q) = \{e \in in(v) \mid State(Source(e)) = q\} \tag{3.1}$$

With these definitions, all diffusion models in Table 2.1 (except to the DeGroot model) can be considered as specific instances of the general model M. The following section will present the generalization in more detail.

Figure 3.2: Requirements used to describe diffusion models.

1. **Undirected Graph:** For all $e_x = (v_1, v_2, w_x) \in E$ there exists one $e_y = (v_2, v_1, w_y) \in E$. Note, that an undirected graph does not assume that the weight for both connections is the same ($w_x \neq w_y$).
2. **Reflexive Graph:** The next state of all nodes can be determined only by the inputs from neighbor nodes and the current state.
3. **Irreflexive Graph:** The next state of all nodes can be determined only by the inputs from neighbour nodes.
4. **Stochastic:** For every $v \in V$:
   4.1 input: $\sum_{e \in in(v)}$ Weight$(e) = 1$
   4.2 output: $\sum_{e \in out(v)}$ Weight$(e) = 1$
5. **Equal influence:** For every $v \in V \wedge e \in in(v)$: Weight$(e) = 1/|in(v)|$. Equal influence across inputs implicates a stochastic input (4.1).

Bi-directional Threshold Model $M_{\textbf{t-BI}}$: The model requires an undirected graph (requirement 1, see Figure 3.2), equal influence across all neighbors (requirement 4, see Figure 3.2) and irreflexivity(requirement 3, see Figure 3.2). The threshold model is binary state which are denoted by P and $\neg P$ which yields $Q = \{P, \neg P\}$. We can define the transition function δ in equation 3.3 by defining an axiom S that checks whether the threshold function is fulfilled. Hence, the list of possible input symbols is $\Sigma = \{S, \neg S\}$.

$$\delta(v) = \begin{cases} P, & \text{if } S(v) \\ \neg P, & \text{otherwise} \end{cases} \tag{3.2}$$

The axiom S can be written as a proportion of the neighbors following P and $\neg P$ (equation 3.3) or by a summation of the states (equation 3.4). The latter one requires that the states P and $\neg P$ are numerically equivalent to 1 and 0.

$$S(v) = \frac{1}{|in(v)|} |ifx(v, P)| \geq t \tag{3.3}$$

$$S(v) = \frac{1}{|in(v)|} \sum_{e \in in(v)} State(Source(e)) \geq t \tag{3.4}$$

It was already shown that this model can be represented by CL (Section 3.1). It is also possible to create a simple DFA from equation 3.2 that is independent of the state:

$$\begin{tikzcd} P \arrow[loop left, "S"] \arrow[r, bend right, "\neg S"'] & \neg P \arrow[loop right, "\neg S"] \arrow[l, bend right, "S"'] \end{tikzcd} \tag{3.5}$$

Uni-directional Threshold Model $M_{\textbf{t-UNI}}$: The uni-directional threshold model is the same as the bi-directional threshold model except for the update policy which does not allow models to switch back from P to $\neg P$. This model may not be represented by CL because the model behaves differently depending on whether it is in state P or $\neg P$. The following DFA with the axiom in equation 3.3 or equation 3.4 describes this model.

$$\begin{tikzcd} P \arrow[loop left, "S \vee \neg S"] & \neg P \arrow[loop right, "\neg S"] \arrow[l, bend right, "S"'] \end{tikzcd} \tag{3.6}$$

Weighted bi-directional threshold model $M_{\textbf{wt-BI}}$: The weighted threshold model can be represented by the same DFA as shown in 3.5 but with a slightly adapted axiom $S(v)$ shown in equation 3.7 where the weight of the incoming edges is taken into account. The node-specific threshold is called t_v.

$$S(v) = \frac{\sum_{e \in in(v)} Weight(e) * State(Source(e))}{|in(v)|} \geq t_v \tag{3.7}$$

Weighted uni-directional threshold model $M_{\textbf{wt-UNI}}$: This model uses the same axiom as the weighted bi-directional threshold model (equation 3.7) with the DFA 3.6.

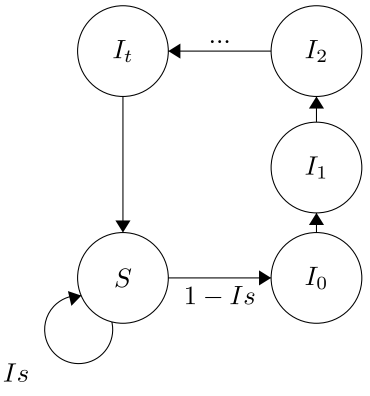
SIS Model $M_{\mathbf{sis}}$: The probabilistic SIS model is highly dependent on the amount of time steps t_I for which a node stays infected. The easiest way to model such a transition is by creating a state for each infection step. Hence, a model with t_I infection steps may be represented by a state machine with $t_I + 1$ states such that $Q = \{S, I_1, I_2, ..., I_{t_I}\}$. A checkmark for infected neighboring nodes can be defined by combining all neighbor nodes that are in one of the infected states.

$$\begin{aligned} Infected(v) = \{x \mid x \in ifx(v, I_1) \vee \\ x \in ifx(v, I_2) \vee \ ... \ \vee x \in ifx(v, I_{I_t})\} \end{aligned} \tag{3.8}$$

The probability of a node staying susceptible Is can then be calculated using equation 3.9. Consequently, the counter probability $1 - Is$ gives the probability that the node becomes infected.

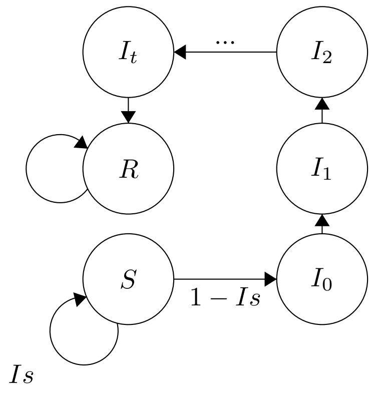
$$Is = (1 - p)^{|Infected(v)|} \tag{3.9}$$

Using this equation, it is possible to create a transition function seen in equation 3.10. It becomes clear that the model is a PA by looking at the two possible choices in the transition table when the node is in state S. The model also requires state transitions without external input which is a requirement by a DFA. However, the given PA can be reduced to a DFA.



(3.10)

SIR Model $M_{\mathbf{sir}}$: The SIR can be defined in a similar way as the SIS model. The model uses the same fixed number of time steps t_I in the infection process, but the node transitions to a recovered state afterwards. Hence, the model contains $t_I + 2$ states such that $Q = \{S, R, I_1, I_2, \ ... \ , I_{t_I}\}$. The transition function for the SIR model is shown in equation 3.11. The finite state machine is exactly the same as the PA described in M_{sir} except that the machine moves to state R after it cycled through all infection states.



(3.11)

Propositional Opinion Diffusion $M_{\mathbf{maj\text{-}POD}}$, $M_{\mathbf{u\text{-}POD}}$: The model is defined as an undirected graph (requirement 1, see Figure 3.2) with equal influence (requirement 5, see Figure 3.2). The definition is not limited to a binary state and may contain any finite amount of states Q. Liu et al. (2014) describe two different approaches to model the opinion diffusion:

$M_{\mathbf{maj\text{-}POD}}$: The majoritarian case where a node changes its belief if there exists a strict majority in the beliefs of the neighbor nodes and its own belief. The definition requires that the model is reflexive such that each node is connected to itself. For each state x, an axiom S_x (equation 3.12 gives an example with three states) is introduced that specifies whether the majority case is fulfilled. All axioms are mutually exclusive because there may only be a single majority (or no majority) at any point in time.

$M_{\mathbf{u\text{-}POD}}$: The unanimity case describes the rule where a node changes its belief if there is an unanimous consens among the neighboring nodes. This definition is irreflexive (requirement 3, see Figure 3.2) because the next state is not dependent on the current state. For each state x, an axiom S_x is introduced (equation 3.13 gives an example with three states) that specifies whether there is a unanimous consens among the neighbors.

$$
\begin{aligned}
S_0 =&|ifx(v,P_0)| > |ifx(v,P_1)| \\
&\land |ifx(v,P_0)| > |ifx(v,P_2)| \\
S_1 =&|ifx(v,P_1)| > |ifx(v,P_0)| \\
&\land |ifx(v,P_1)| > |ifx(v,P_2)| \\
S_2 =&|ifx(v,P_2)| > |ifx(v,P_0)| \\
&\land |ifx(v,P_2)| > |ifx(v,P_1)| \\
Up =&\neg S_0 \land \neg S_1 \land \neg S_2
\end{aligned}
\tag{3.12}
$$

$$
\begin{aligned}
S_0 =&|ifx(v,P_0)| = |in(v)| \\
S_1 =&|ifx(v,P_1)| = |in(v)| \\
S_2 =&|ifx(v,P_2)| = |in(v)| \\
Up =&\neg S_0 \land \neg S_1 \land \neg S_2
\end{aligned}
\tag{3.13}
$$

With these axioms it is possible to build a DFA (equation 3.14). The DFA is the same for the unanimous case and the majority case. Table 3.1 describes that the lower bound for the majority model is a DFA and for the unanimity model it is CL. This does not become clear when looking at the axioms or at the DFA. It is based on the fact that the definition of $M_{\text{maj-POD}}$ is reflexive while the definition of $M_{\text{u-POD}}$ is irreflexive.

$$\begin{tikzcd}
P_0 \arrow[loop above, "Up \lor S_0"] \arrow[rr, bend left, "S_1"] \arrow[dr, bend left, "S_2"] & & P_1 \arrow[loop above, "Up \lor S_1"] \arrow[ll, bend left, "S_0"] \arrow[dl, bend right, "S_2"'] \\
& P_2 \arrow[loop below, "Up \lor S_2"] \arrow[ul, bend left, "S_0"] \arrow[ur, bend right, "S_1"'] &
\end{tikzcd} \tag{3.14}$$

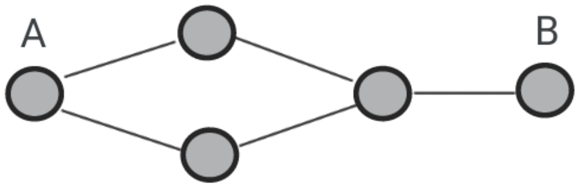


Figure 4.1: A one-way bridge in $M_{t=0.5}$ that transfers the state from node A to B but not from B to A.

4 Conversion of $M_{\textbf{wt-BI}}$ to $M_{\textbf{t-BI}}(t = 0.5)$

The previous sections showed that it is possible to unify the model definitions by using a more general concept of a DFA. Another approach is the exact translation between models. Is it possible to represent the same behavior of a certain model by a different model? For some cases it is certainly possible. Easley and Kleinberg (2010) already discussed a basic translation between the SIS and the SIR model. The following section will show a translation of a uni-directional, directed and weighted threshold model with unique thresholds ($M_{\text{wt-UNI}}$) to an uni-directional, unweighted and undirected threshold model with a global threshold $M_{\text{t-UNI}}(t = 0.5)$. First, it is important to note that the DFAs are the same in both cases. The only difference between those models is the way in which they aggregate the neighbor opinions. The weighted threshold model gives each connection a specific weight and has a node-specific threshold.

Adding intermediate nodes that emulate the behavior of the weights and node-specific thresholds enables the conversion between those models. The core concept of the conversion is a construct of three nodes that works as an one-way bridge (see Figure 4.1) in $M_{\text{t-BI}}(t = 0.5)$. It allows a state transfer from node A to B, but blocks the transfer from node B to A. This is due to a cluster that blocks the cascade coming from side B: The node on the left of B will never change its opinion because it has two neighbors from side A that prevent the node from reaching the 0.5 threshold. A cascade coming from node A does not hit this blockade and reaches node B.

These structs are used to emulate the behavior of the weighted and directed edges. Because the structs only allow a one-sided transfer they work the same way as directed edges. It is possible to simulate the influence of the weights by adding any number of structs that fulfills the relation to the other inputs and the threshold. For example, if a node A has a threshold of 2/4 and receives an input with weight 1/4 from B and another input with weight 3/4 from C, this could be represented by one struct from B to A, three structs from C to A, and two nodes emulating the threshold. More generally, the least common divisor of the threshold and all incoming weights divided by the weight of the current edge yields the amount of blockers needed to emulate the connection. This approach is repeated for all weights. Lastly, the graph needs to be balanced because the structs do not only exhibit an influence on the target node B but also on the source node A. Each struct adds an additional influence of two $\neg P$ nodes to A. This influence needs to be balanced by adding two counter nodes with state P to ensure that the model behaves in the same way. The following algorithm describes the steps in more detail:

1. For each node in $M_{\text{wt-BI}}$ consider the weights of all incoming edges e_{in} and the node's threshold t. Let k be the least common multiple of this set (the set contains fractional values; all values are expanded with a value that turns them into them integers).
2. For this node replace each incoming edge $e = (v_{\text{source}}, v_{\text{destination}}, w)$ with $N = k/w$ constructs of three nodes v_1, v_2, v_3 with belief $\neg P$ where v_{source} is connected to v_1; v_1 to v_2 and v_3; v_2 and v_3 to $v_{\text{destination}}$ (an example can be seen in Figure 4.1 where A is v_{source} and B is $v_{\text{destination}}$).
3. Add $2k/w$ number of counter nodes to v_{source} that follow the belief of P.
4. Add t/w nodes with belief of $\neg P$ that are connected to $v_{\text{destination}}$ that simulate the threshold.

5 Implementation

A substantial part of the project is the development of a website that is capable of simulating the diffusion models and the conversion between models. The goal of the implementation is an open-source accessible tool that creates a more natural understanding of the differences and similarities between the models. Therefore, the tool is implemented as a freely accessible website that can simulate the diffusion behaviors of all models listed in this thesis. In general, the behavior of each model differs significantly for different graph topologies which is the reason that there are multiple predefined graph topologies available. The implementation and the available set of topologies is based on the NetworkX framework (Hagberg, Schult, and Swart (2008)) which is available under the 3-clause BSD license. A list of all available graph topologies is given in the appendix A. The visual implementation is based on the Plotly Dash framework (Plotly Technologies Inc. (2015)) which integrates interactive graphs on websites (MIT license). The source code for the visual tool is also published under MIT license and can be accessed at https://github.com/KonstantinRr/graph-translator. A detailed guide on running the project can be found in the README file.

The diffusion website can be used to show the diffusion behavior of the presented models in different topologies. In the following section, a few examples of diffusion behavior are shown. Figure 5.1 shows the limit of the diffusion behavior of a threshold model $M_{\text{t-BI}}(0.5)$ with a threshold of 0.5. It becomes visible that the limit distribution ends up in clusters having the same state. It can also be observed that the edges of clusters are always in positions with low density connections to other clusters. Especially interesting is the cluster of four nodes at the top right of the graph. These nodes switch periodically between states at every tick because each node has a majority of neighbors with a different opinion. These nodes are the only nodes in the distribution which are unstable. Figure 5.3 shows a similar limit distribution for the $M_{\text{maj-POD}}(s = 5)$ model. The same topology used in the previous example was initialized with random states s_0, s_1, s_2, s_3, s_4. The stable state shows clear edges between clusters at positions with a low density.
Figure 5.2 shows the limit distribution of the model $M_{\text{sir}}(p = 0.05, t_i = 20)$ on the same graph. Each node has a probability of 0.05 to get infected for each neighbor node at the time step. The node stays

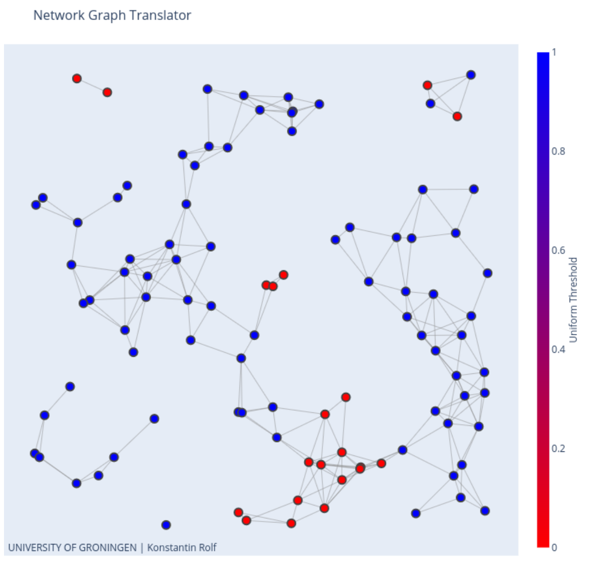


Figure 5.1: Example limit distribution of $M_{\textbf{t-BI}}(0.5)$ on a random geometric graph with $m = 100$ and $c = 0.25$. There is a clear distribution into clusters with a periodically changing cluster of four nodes at the top right.

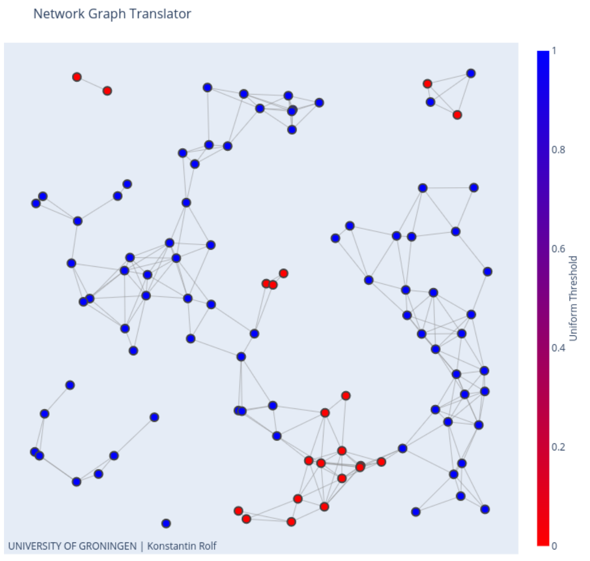


Figure 5.2: Example limit distribution of $M_{\textbf{sir}} = 0.05$ on a random geometric graph with $m = 100$ and $c = 0.25$. All nodes except two nodes have become infected and recovered.

infected for 20 steps and may infect other nodes. The Figure shows that all nodes except two nodes became infected and are now recovered.

6 Discussion

In the previous sections it was shown how different diffusion models work (see Section 2) and that they can be represented by deterministic finite automatons (see Section 3.1). The DeGroot model was the only model that required higher order complexity which cannot be represented by a DFA. Section 3.2 then defined a set of DFAs as nodes connected by edges in a more formal way and presented the DFA transition tables for multiple diffusion models. This showed that it is indeed possible to unify the model definitions by finding the lower bound automaton that has enough complexity to represent the models. Section 4 introduced a translation between the $M_{\text{wt-UNI}}$ and $M_{\text{t-UNI}}$ to show that it is also possible to translate specific instances of models into each other. The given translation worked by adding nodes that emulate the behavior of the weights and thresholds.

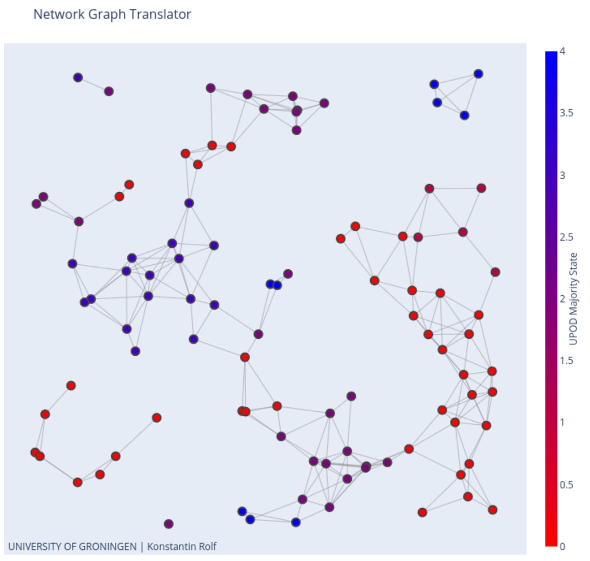


Figure 5.3: Example limit distribution of $M_{\textbf{maj-POD}\ s=5}$ on a random geometric graph with $m = 100$ and $c = 0.25$ and initial random state. The nodes form clusters that have the same state. Similar to Figure 5.1 all edges of the clusters are in low-density regions.

This thesis showed that all models are equivalent on a more abstract level. As a next step, the similarities and differences between the DFAs could be investigated in more detail. Specifically, it should be possible to combine the DFAs presented in this thesis to create a single DFA that is able to simulate all models. Another approach is the implementation of the DeGroot model in automata logic. This could happen through the usage of a more powerful automata (at least a push down automaton) or by using simplifications to reduce the state space. The research in this domain may give rise to a better understanding of the connections between diffusion models. Models from epidemiology and opinion diffusion combined yield a better understanding why certain events in social media behave similar to viral events.

References

Alexandru Baltag, Zoé Christoff, Rasmus K. Rendsvig, and Sonja Smets. Dynamic Epistemic Logics of Diffusion and Prediction in Social Networks. *Studia Logica*, 107(3):489–531, June 2019. ISSN 1572-8730. doi: {10.1007/s11225-018-9804-x}.

Zoé Christoff and Davide Grossi. Stability in Binary Opinion Diffusion. In Alexandru Baltag, Jeremy Seligman, and Tomoyuki Yamada, editors, *Logic, Rationality, and Interaction*, Lecture Notes in Computer Science, pages 166–180, Berlin, Heidelberg, 2017. Springer. ISBN 9783662556658. doi: {10.1007/978-3-662-55665-8}.

Morris H. DeGroot. Reaching a Consensus. *Journal of the American Statistical Association*, 69(345): 118–121, March 1974. ISSN 0162-1459. doi: 10.1080/01621459.1974.10480137.

David Easley and Jon Kleinberg. *Networks, crowds, and markets: reasoning about a highly connected world*. Cambridge University Press, New York, 2010. ISBN 9780521195331. OCLC: ocn495616815.

Umberto Grandi, Emiliano Lorini, and Laurent Perrussel. Propositional Opinion Diffusion. pages 989–997. ACM, 2015.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

Fenrong Liu, Jeremy Seligman, and Patrick Girard. Logical dynamics of belief change in the community. *Synthese*, 191(11):2403–2431, July 2014. ISSN 1573-0964. doi: 10.1007/s11229-014-0432-3.

Plotly Technologies Inc. Collaborative data science, 2015. URL https://plot.ly.

Essi Ilona Pöyry, Salla-Maaria Laaksonen, Arto Ilmari Kekkonen, and Juho Ilmari Pääkkönen. *Anatomy of viral social media events*. University of Hawai'i at Manoa, 2018. ISBN 9780998133119. doi: 10.24251/hicss.2018.272. URL https://helda.helsinki.fi/handle/10138/237224.

Micheal O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, April 1959. ISSN 0018-8646. doi: 10.1147/rd.32.0114.

A Appendix

Full list of supported graph topologies created by Hagberg et al. (2008).

- Balanced Tree(r, h): Create the balanced r-ary tree of height h.
- Barbell Graph(m_1, m_2): Creates the Barbell Graph: Two complete graphs connected by a path.
- Binomial Tree(n): Creates the Binomial Tree of order n.
- Circular Ladder Graph(n): Creates the circular ladder graph CL_n of length n.
- Cycle Graph(n): Returns the cycle graph C_n of cyclically connected nodes.
- Dorogovtsev-Goltsev-Mendes Graph(n): Creates the hierarchically constructed Dorogovtsev-Goltsev-Mendes graph.
- Empty Graph$()$: Returns the Empty Graph with n nodes and zero edges.
- Full Rary Tree(r, n): Creates a full r-ary tree of n vertices.
- Ladder Graph(n): Creates the Ladder graph of length n.
- Lollipop Graph(m, n): Creates the Lollipop Graph; K_m connected to P_n.
- Null Graph$()$: Creates the Null graph with no nodes and no edges.
- Path Graph(n): Creates the Path graph P_n of linearly connected nodes.
- Star Graph(n): Creates the star graph of order n.
- Trivial Graph$()$: Creates the Trivial Graph with one node (with label 0) and no edges.
- Turan Graph(n, r): Creates the Turan Graph.
- Wheel Graph(n): Creates the Wheel Graph.
- Random Geometric(n, c): Creates a Random Geometric Graph in the unit cube of dimensions dim.
- Paley Graph(p): Returns the Paley $(p-1)/2-$regular Graph on p nodes.
- Grid 2D Graph$(m, n, periodic)$: Creates the two-dimensional Grid Graph.