Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

TEMPORAL LOGICS FOR DIFFUSION IN SOCIAL NETWORKS

Vitor Pereira Machado

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Valmir Carneiro Barbosa & Mario Roberto Folhadela Benevides (externo)

Rio de Janeiro
Junho de 2023

TEMPORAL LOGICS FOR DIFFUSION IN SOCIAL NETWORKS

Vitor Pereira Machado

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Valmir Carneiro Barbosa
& Mario Roberto Folhadela Benevides (externo)

Aprovada por: Prof. Valmir Carneiro Barbosa
Prof. Mario Roberto Folhadela Benevides
Prof. Priscila Machado Vieira Lima
Prof. Manuel António Gonçalves Martins
Prof. Edward Hermann Haeusler
Prof. Bruno Lopes Vieira

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2023

Pereira Machado, Vitor

Temporal Logics for Diffusion in Social Networks/Vitor Pereira Machado. – Rio de Janeiro: UFRJ/COPPE, 2023.

XIII, 101 p.: il.; $29,7$cm.

Orientador: Valmir Carneiro Barbosa

& Mario Roberto Folhadela Benevides (externo)

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2023.

Referências Bibliográficas: p. 79 – 86.

1. social networks. 2. logic. 3. model checking. I. , Valmir Carneiro Barbosa & Mario Roberto Folhadela Benevides (externo). II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

To my beloved mother and father,
whose love, guidance, and
unwavering support have made
everything, including this work,
possible.

Agradecimentos

I would like to express the deepest appreciation to my supervisor and friend Professor Mario Roberto Folhadela Benevides, for his good spirits, guidance and enthusiasm in regards to research. Without his support this work would not have been possible.

I also wish to thank professors Valmir Barbosa, Priscila Vieira, Edward Hermann and Bruno Lopes for kindly accepting to be part of my thesis defense committee.

I also extend my gratitude to professors Alexandru Baltag, Zoé Christoff, Rasmus K. Rendsvig and Sonja Smets for their inspirational work in "Dynamic Epistemic Logics of Diffusion and Prediction in Social Networks". Their research served as a solid foundation for this work.

Finally, I wish to thank my good friend Paulo Xavier for his support and companionship along the way.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

LÓGICAS TEMPORAIS PARA DIFUSÃO EM REDES SOCIAIS

Vitor Pereira Machado

Junho/2023

Orientador: Valmir Carneiro Barbosa
& Mario Roberto Folhadela Benevides (externo)

Programa: Engenharia de Sistemas e Computação

Esta tese apresenta LTL-SN, um framework baseado em lógica para modelar redes sociais usando a Lógica Temporal Linear (LTL). O LTL-SN incorpora propriedades de redes sociais de adoção, onde os agentes adotam comportamentos baseados no comportamento de seus amigos. Além disso, ele explora variações do modelo para analisar a propagação de doenças em populações usando modelos compartimentais. A tese aproveita linguagens de especificação existentes e implementações de verificação de modelos, representando redes sociais como caminhos LTL especificados com modelos nuXmv. Os resultados de pesquisas publicadas são referenciados. Fornecemos axiomatização e provas de correção e completude para cada variante lógica apresentada, incluindo os modelos compartimentais SIR e SIRS.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

TEMPORAL LOGICS FOR DIFFUSION IN SOCIAL NETWORKS

Vitor Pereira Machado

June/2023

Advisor: Valmir Carneiro Barbosa
& Mario Roberto Folhadela Benevides (external)

Department: Systems Engineering and Computer Science

This thesis introduces LTL-SN, a logic-based framework for modeling social networks using Linear Temporal Logic (LTL). LTL-SN incorporates social network properties of adoption, where agents adopt behaviors based on their friends' behavior. Additionally, it explores variations of the model for analyzing disease spread in populations using compartmental models. The thesis leverages existing specification languages and model checking implementations, representing social networks as LTL paths specified with nuXmv models. Published research findings are referenced. We provide axiomatization and proofs of soundness and completeness for each presented logic variant, including compartmental SIR and SIRS models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Earlier studies of social interactions related with diffusion of innovations date back to at least around the late 19th century [3] and early 20th century [4]. In the late 1990's Internet social network services gained traction, with networks such as Friendster, MySpace, LinkedIn, and with Facebook as one of the more recently emerging platforms. These services allow people from all over the world to interact in meaningful ways, and as such, have gained tremendous importance in popular culture in recent years.

In a broader sense, these networks are commonly defined by a set of actors (individuals, communities or societies), relations between pairs of entities, and a set of social interactions [5]. For instance, in Facebook, an entity is a person, user of the service; the relations between pairs of entities are the friendship relations and the social interactions are posts which appear on the user's timeline.

Our goal in this paper is to develop a logic based on Linear Temporal Logic (LTL), denoted LTL-SN for social network models. Semantics are standard LTL, but we are interested in models with social network properties of adoption, where an agent adopts a behavior when a given proportion of its friends exhibit that behavior.

We also explore variations of this model for the spread of diseases among populations, so called compartmental model for epidemics. Specifically we present two variations of these models with and without agent recovery.

The reasoning for pursuing this direction is the possibility to leverage existing specification languages and model checking implementations. The social networks presented here evolve as standard LTL paths, and can be specified as nuXmv models, for instance.

A portion of the research presented in this thesis has been published on the Journal of Logic and Computation, volume 32, issue 6, September 2022, pages 1088–1108 [6].

The roadmap is as follows: In chapter 2 we briefly go over some of the literature that inspired or served as a basis for this work, providing an overview of the past and current tendencies in the study social networks with such frameworks. In chapter 3 we introduce LTL-SN, its language and semantics, present some simple examples of model evolution and formulas, and in chapter 4 we present some important definitions that lead on to the axiomatization and finally its soundness and completeness results in chapter 5. In the next chapters we explore variants of LTL-SN for different purposes: chapter 6 introduces a variant supporting multiple behaviors. In chapter 7 we briefly discuss model checking and its time complexity. Chapter 8 introduces a variation which explores a SIR model for the study of infectious diseases. We go on in chapter 9 expanding the SIR variation to a branching CTL-based variant; And in chapter 10 a SIRS model which allows agents to become susceptible to infections again. Finally, we present our closing words and future works in chapter 11.

Chapter 2

Background

In this chapter, we will introduce and define some key concepts and models that are commonly used in various fields such as computer science, epidemiology, and social network analysis. These concepts will be relevant for the work we develop here.

We will introduce linear and branching temporal logics, logics that deals with reasoning about events that occur over time. Next, we will define social networks and present some of the literature in logics that aims to study these systems. We will then introduce SIR compartmental models, which are widely used in epidemiology to study the spread of infectious diseases. Finally, we will discuss model checking, a technique used to verify the correctness of models against certain specifications or properties.

2.1 Kripke structures

Transition systems are representations of states in a system, and transitions between these states. These transitions can be labelled or unlabelled. Lastly, a transition system is denoted "total" when every state contains an outbound transition.

Kripke structures are a special case of transition systems: total unlabeled transition systems [7]. Each state also has a set of propositions associated with it, via a labelling function. Usually Kripke structures are represented as 4-tuples, $M = (S, I, R, L)$, such that:

- S is a finite set of states;
- $I \subseteq S$ is a set of initial states;
- $R \subseteq S \times S$ is a binary transition relation between states;
- $L : S \to 2^{AP}$ is a labelling function associating each state to a set of atomic propositions (AP).

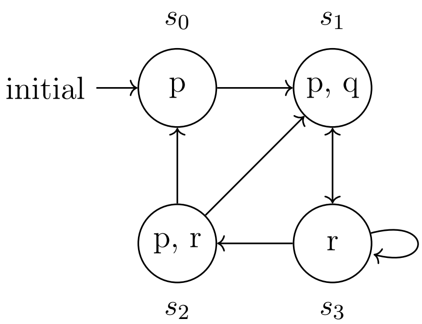


Figure 2.1: A Kripke structure represented as a bidirected graph.

Figure 2.1 is an example of a Kripke structure with the following properties:

- $S = \{s_0, s_1, s_2, s_3\}$;
- $I = \{s_1\}$;
- $R = \{\{s_0, s_1\}, \{s_1, s_3\}, \{s_2, s_0\}, \{s_2, s_1\}, \{s_3, s_1\}, \{s_3, s_2\}, \{s_3, s_3\}\}$;
- $L = \{\{s_0, \{p\}\}, \{s_1, \{p, q\}\}, \{s_2, \{p, r\}\}, \{s_3, \{r\}\}\}$.

2.2 Temporal Logics

Temporal logics are formal systems made to represent and reason about propositions qualified by time-related modalities. As such, temporal logics are modal logics, meaning they provide operators that act on propositions to generate temporal propositions. Some examples of temporal quantifiers used in these logics are "in the future p holds" or "p holds until q holds", and so on.

Some of the most common logics used for model checking systems fall under the umbrella of temporal logics, first applied to this purpose in [8]. The most commonly used being Linear Temporal Logic (LTL) and Computation Tree Logic (CTL), or the LTL superset CTL* [9].

One of the most important concepts for model checking with temporal logics is that of fairness. A run of a program or automaton is said to be “fair” if every one of its processes are always given a chance to run indefinitely into the future. That is, a process never falls into a condition of “starvation”. Many times in model checking it is convenient to employ fairness constraints [10], in which only paths that do not ignore indefinitely enabled processes are considered, and paths in which certain processes are never given a chance to run are discarded.

2.2.1 Linear Temporal Logic (LTL)

Linear Temporal Logic (LTL) was first introduced in the seminal work of [8].

LTL formulas are evaluated against paths on a Kripke structure, and represent a series of states over time (e.g.: the execution of a program, or automaton). These formulas are composed of atomic propositions, the standard classical operators and temporal operators.

Definition 1 (Language of LTL)**.** *The language of LTL is given by the following BNF:*

$$\varphi := \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid X\varphi \mid F\varphi \mid G\varphi \mid \varphi_1 U \varphi_2$$

The operators are read as "Next" (X), "Finally" (F), "Globally" (G) and "Until" (U).

The "Next" operator is a unary operator that represents the proposition that the next state in the sequence satisfies a given condition. For example, the formula Xp means "the next state satisfies p". The Next operator is often used to express statements about immediate future states.

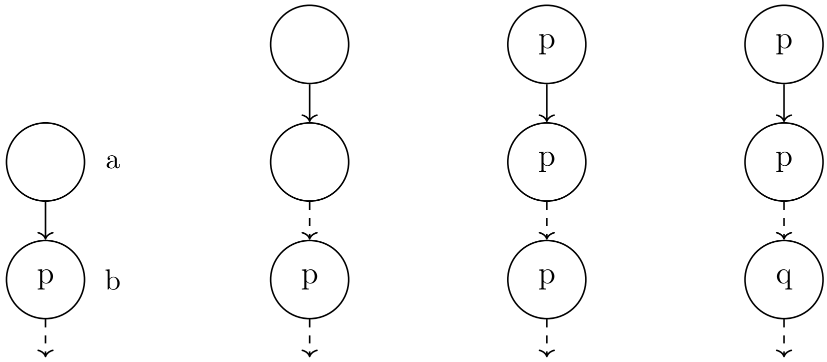
The "Finally" (also sometimes denoted "Future") operator is a unary operator that represents the proposition that a given condition will eventually become true at some point in the future. For example, the formula Fp means "p will eventually become true". The Finally operator is often used to express statements about eventualities.

The "Globally" operator is a unary operator that represents the proposition that a given condition holds globally, i.e., it holds in all future states. For example, the formula Gp means "p holds in all future states". The Global operator is often used to express statements about invariance or safety conditions.

And finally, the "Until" operator is a binary operator that represents the proposition that a certain condition holds until another condition becomes true. For example, the formula pUq means "p holds until q is true". The Until operator is often used to express statements about future states that satisfy a particular condition.

Figure 2.2 demonstrates models that satisfy each of the temporal operators in LTL.

Definition 2 (Satisfaction of LTL formulas)**.** *Given a model $\mathcal{M} = (S, R, V)$ and a path $\pi = s_0, s_1, s_2, \ldots$ such that $s_i R s_{i+1}$ for all i. The satisfaction of formulas in*



(a) Xp (b) Fp (c) Gp (d) pUq

Figure 2.2: Example paths where each of the LTL temporal operators hold. Notice that the paths are infinite, and dashed lines mean an arbitrarily long path. Figure 2.2c represents a path where p holds in all states, and figure 2.2d a path where q holds from the first state where it appears and onwards.

LTL is defined as follows:

$$
\begin{array}{lll}
\pi, i \models p & \textit{iff} & p \in V(s_i) \\
\pi, i \models \neg\varphi & \textit{iff} & \pi, i \not\models \varphi \\
\pi, i \models \varphi_1 \wedge \varphi_2 & \textit{iff} & \pi, i \models \varphi_1 \textit{ and } \pi, i \models \varphi_2 \\
\pi, i \models X\varphi & \textit{iff} & \pi, i+1 \models \varphi \\
\pi, i \models \varphi U \psi & \textit{iff} & \textit{there exists a } n \geq 0 \textit{ such that} \\
 & & \pi, n \models \psi \textit{ and } \pi, j \models \varphi \textit{ for all } 0 \leq j < n
\end{array}
$$

Other operators, such as F and G can be defined in terms of these:

- $F\varphi \equiv \top U \varphi$;
- $G\varphi \equiv \neg F \neg\varphi$.

The complexity of the model checking problem in LTL has a complexity of PSPACE-complete with an algorithm complexity of $O(|TS| \cdot exp(|\varphi|))$ where $|TS|$ is the size of the transition system model, and $|\varphi|$ is the size of the formula [11]. The algorithm complexity is linear over the size of the transition system, however even for a reasonably small model the state explosion problem is still an issue [12]. This means that model checking LTL naïvely is a very complex task for all but the most simple use cases, and symbolic model checking is commonly employed. Despite the difficulties in model checking LTL, it is considered to be the most useful for practical use, due to it being more appropriate for behavioral instead of structural specifications, and also inherently supporting fairness [13].

Two properties that are commonly considered during model checking are "fairness" and "invariance". Consider the expression GFp: it states that for every state in the path, there is always some state in the future in which p holds. Consider that p

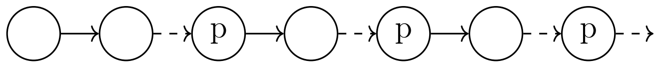


Figure 2.3: This figure provides intuition for a fair path, in which there is always a state in the future where p holds. In other words, the “absolute fairness” or “impartiality” statement GFp holds.

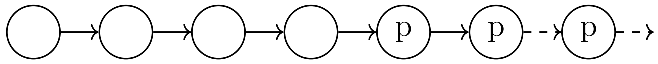


Figure 2.4: This figure provides intuition for an invariant path, where FGp holds. In other words, in some state in the strict future, p holds and holds in all subsequent states.

represents a given process being executed, for example. This expression, commonly denoted “absolute fairness” [14] (see figure 2.3), captures the fact that process p is always given a chance to run and the system will never reach a situation of starvation.

Another important property is denoted “invariance” (see figure 2.4), captured with the expression FGp [13]. It means that, at some point in the future, p will always hold. It can be used to represent a desirable property in a system, in which one wishes to assure such property will always eventually be true.

2.2.2 Computation Tree Logic (CTL)

First proposed and used in [15] to model concurrent programs, Computation Tree Logic (CTL) is a formal logic used to reason about the behavior of systems with a tree-like structure, such as computer programs, hardware systems, and communication protocols.

Definition 3 (Language of CTL)**.** *The language of CTL is given by the following BNF:*

$$\begin{aligned}\varphi := &\top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid AX\varphi \mid AF\varphi \mid AG\varphi \mid \varphi_1 AU \varphi_2\\ &\mid EX\varphi \mid EF\varphi \mid EG\varphi \mid \varphi_1 EU \varphi_2\end{aligned}$$

The path quantifier prefixes in operators are read as “Eventually” (E) and “Always” (A). Temporal operator suffixes X, F, G and U are read the same as their LTL counterparts.

Definition 4 (Satisfaction of CTL formulas)**.** *Given a model $\mathcal{M} = (S, R, V)$ and a* *state $s \in S$. The satisfaction of formulas in CTL is defined as follows:*

$$
\begin{array}{lll}
\mathcal{M}, s \models p & \textit{iff} & p \in V(s) \\
\mathcal{M}, s \models \neg\varphi & \textit{iff} & \mathcal{M}, s \not\models \varphi \\
\mathcal{M}, s \models \varphi_1 \wedge \varphi_2 & \textit{iff} & \mathcal{M}, s \models \varphi_1 \textit{ and } \mathcal{M}, s \models \varphi_2 \\
\mathcal{M}, s \models EG\varphi & \textit{iff} & \textit{for some path } \pi = s_0, s_1, s_2, \ldots, s_i, \ldots \textit{ such that } s_0 = s \\
 & & \textit{we have for all } \mathcal{M}, s_i \models \varphi \\
\mathcal{M}, s \models \varphi EU\psi & \textit{iff} & \textit{for some path } \pi = s_0, s_1, s_2, \ldots, s_i, \ldots \textit{ such that } s_0 = s \\
 & & \textit{there exists an } i \textit{ such that } \mathcal{M}, s_i \models \psi \\
 & & \textit{and for all } j < i \textit{ we have that } \mathcal{M}, s_j \models \varphi \\
\mathcal{M}, s \models EX\varphi & \textit{iff} & \textit{for some } s' \textit{ such that } sRs' \textit{ we have } \mathcal{M}, s' \models \varphi
\end{array}
$$

Some operators can be defined in terms of others [16]:

- $AX\varphi \equiv \neg EX\neg\varphi$
- $EF\varphi \equiv E[\top U\varphi]$
- $AG \equiv \neg EF\neg\varphi$
- $AF \equiv \neg EG\neg\varphi$
- $\varphi AU\psi \equiv \neg((\neg\psi EU\neg(\varphi \vee \psi)) \vee EG\neg\psi)$

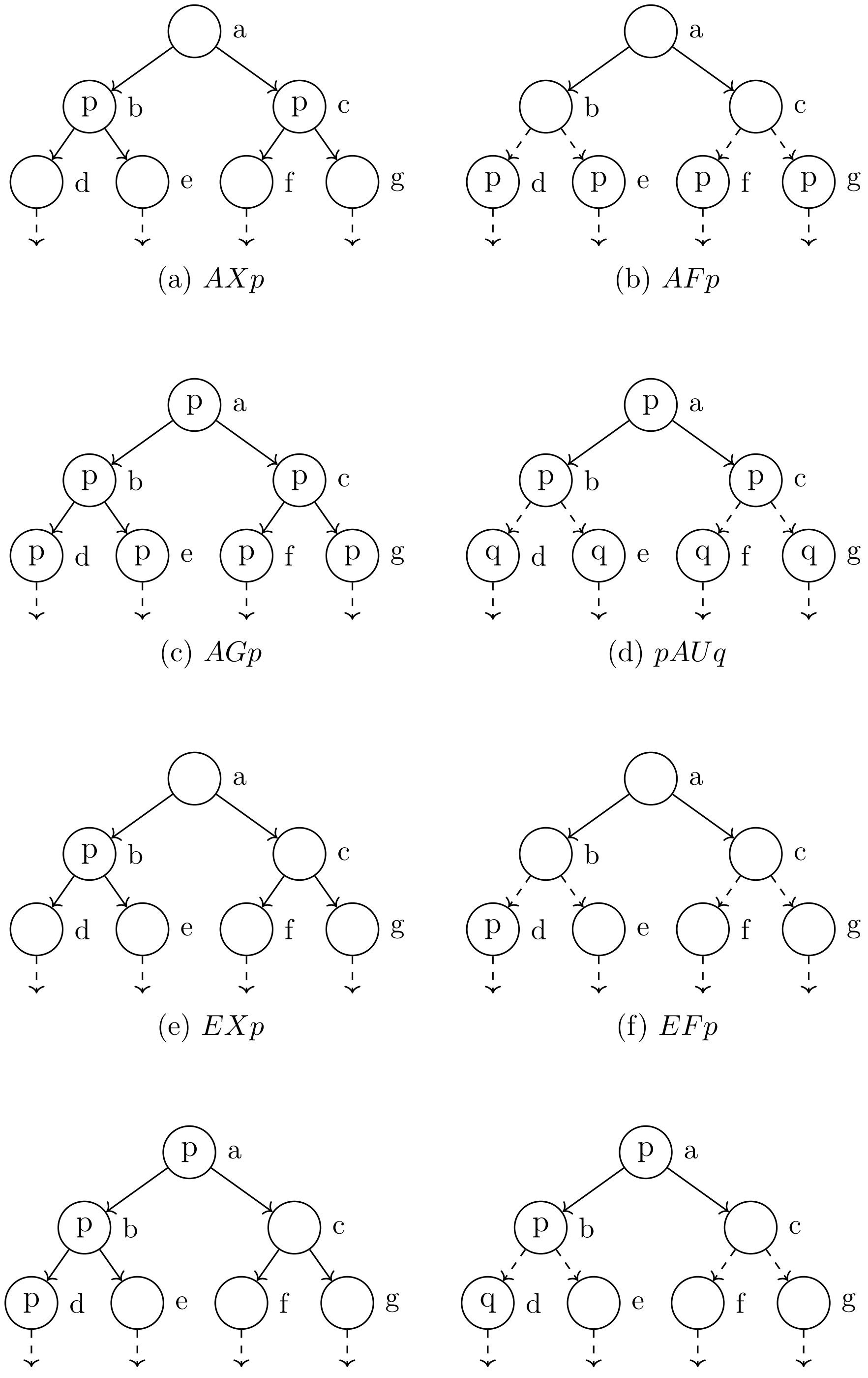
Figure 2.5 demonstrates models that satisfy each of the path quantifiers and temporal operators in CTL.

The model checking problem in CTL has complexity of PTIME with an algorithm complexity of $O(|TS| \cdot |\varphi|)$ where $|TS|$ is the size of the transition system model, and $|\varphi|$ is the size of the formula [11]. While CTL might look like a good alternative to PSPACE-complete LTL, it is generally less useful in practice. [9] says CTL is "unintuitive and hard to use, it does not lend itself to compositional reasoning, and it is fundamentally incompatible with semi-formal verification".

Lastly, it is also worth noting that, contrary to LTL, the concepts of fairness cannot be directly expressed in CTL. Since fairness is very useful restriction in real life applications, some model checkers that do not include LTL or CTL* specifications still provide the ability to define and enforce fairness constraints for CTL-only specifications. Some of these model checkers are NUSMV/NUXMV, PRISM and its-ctl.

2.2.3 CTL*

CTL*, introduced in [17], is a superset of CTL and LTL as shown in figure 2.7, and allows mixing LTL and CTL formulas.



(a) AXp (b) AFp

(c) AGp (d) $pAUq$

(e) EXp (f) EFp

(g) EGp (h) $pEUq$

Figure 2.5: Example paths where each of the CTL path quantifiers and temporal operators hold

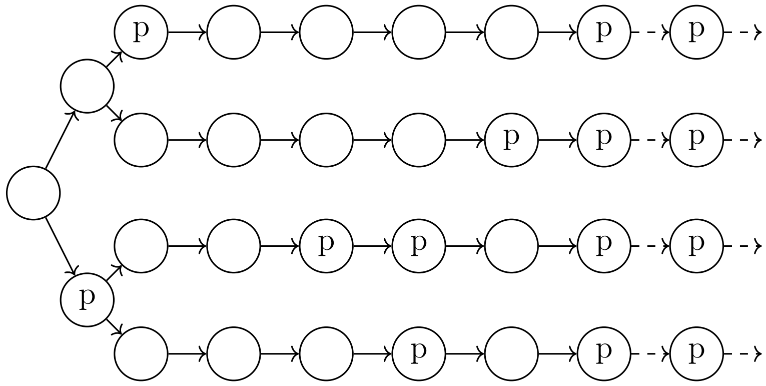


Figure 2.6: A model satisfying the formula $AFAGp$. Notice this is subtly different, and strictly stronger, than the LTL invariance formula FGp shown in figure 2.4.

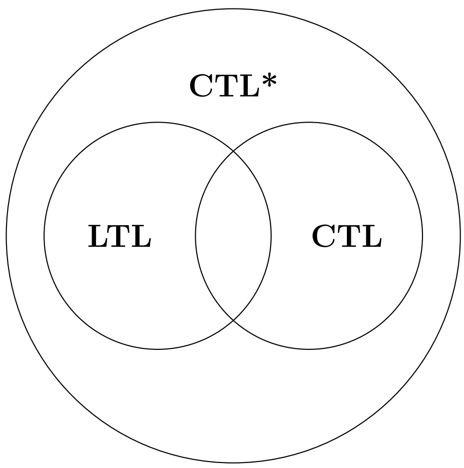


Figure 2.7: The relationship between the expressiveness of LTL, CTL and CTL*

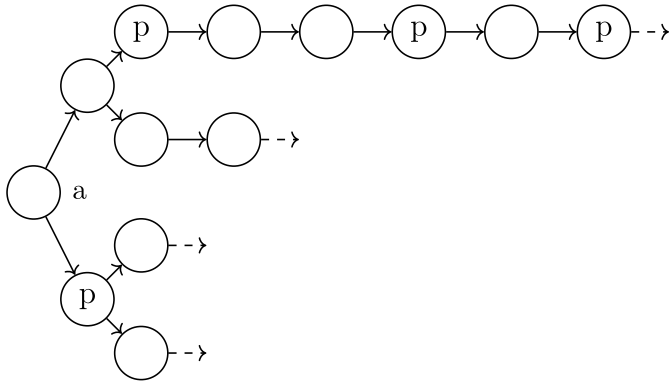


Figure 2.8: The uppermost path represents a path in which there is always a future state where p holds. Therefore, it is the case that $\mathcal{M}, a \models E(GFp)$. The statement GFp is usually denoted "absolute fairness" or "impartiality" and is expressible in LTL [1]. The statement $E(GFp)$, however, is under the path quantifier E and is expressible only in CTL*.

Definition 5 (Language of CTL*)**.** *The language of CTL* is given by the following BNF:*

$$\varphi := \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid A\phi \mid E\phi$$
$$\phi := \varphi \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid X\phi \mid F\phi \mid G\phi \mid \phi_1 U \phi_2$$

Figure 2.8 shows an example of a formula that is in CTL* and outside of both LTL and CTL, and a model where the formula holds.

The complexity of the model checking problem in CTL* is dominated by the LTL complexity [18], that is, it is PSPACE-complete with an algorithm complexity of $O(|TS| \cdot exp(|\varphi|))$ where $|TS|$ is the size of the transition system model, and $|\varphi|$ is the size of the formula [11]. This is deceiving, however, as the algorithms are notoriously more difficult to implement as evidenced by the general lack of availability of CTL* model checkers [13], though at least one proposal exists [19].

2.3 Social Networks

Social network analysis has been used to study a wide range of social phenomena, including the spread of information, the formation of social groups, the emergence of collective behavior, and the diffusion of innovations. Graph-based methods have also been used to identify key individuals and groups within a network, to detect communities or clusters of nodes with similar properties, and to predict the future behavior of the network. Overall, social network studies using graphs have proven to be a valuable tool for understanding the complex dynamics of social systems.

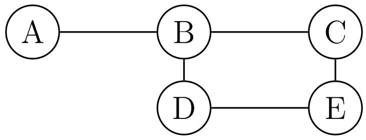


Figure 2.9: This diagram shows a network where agents A, B, C, D, and E are represented as circles. The edges between the nodes represent the connections between the agents. For example, A is connected to B, and B is connected to A, C and D.

"Classical" social networks studies are usually characterized by graph structures (see figure 2.9), composed of nodes, which represent the actors (eg.: people); and the edges, which represent relations or interactions between the nodes (eg.: friendship).

Formal studies of social networks in graph theory, called sociometry, began with [20] as a means to explore how the psychological state of people is related to the structural features of what was termed the 'social configurations' [21].

There are a few logic frameworks which attempt to emulate social network structure and dynamics, and we will now briefly go over some of the most important ones to provide an overview of the formalisms used to study social networks.

The works of [22, 23] deal with how a group of agents should reach consensus by assessing the probabilities that each agent assigns to a given proposition, and that informed disagreements are therefore irrational behavior. These are seminal works in social networks theory.

2.3.1 Literature on logics for social networks

In this section we review some relevant literature for the study of social networks in logics, with a focus in models of diffusion for these networks.

Threshold models are some of the most studied in logics for social networks, such as in [24–27]. In these, agents are influenced by their neighbours into adopting a behavior depending on the proportion of neighbours who already adopted.

In [24] the focus is on establishing conditions for adoption of a "product" by agents, and whether the process has a unique outcome or not, or an unavoidable one. Agents have neighbours and are always aware of their product adoptions. They have a threshold θ, and will adopt a product if more than θ of its neighbours have adopted it or not.

The logic in [25] focuses on the influence of groups of agents over other groups, with formulas such as $A \triangleright B$ meaning "if the group of agents A has adopted, that eventually leads to the adoption by group B", and a sound and complete axiomatization.

In [26] the case for recalcitrant agents in a network with unequal weights among agents is studied. Recalcitrant agents are completely immune to the behavior diffusion (or opinion about a product, in the context presented).

In [27] a different approach to threshold models is taken, by constructing the dynamics using action models and product update. The work proposes an action model approach to the updates, which consists of applying the product between the network graph and a graph that encodes the decision rules, and is uniformly followed by all agents.

A survey of both threshold-based and non threshold-based models is conducted in [28], with concepts such as ranked friends, reliability of agents, and plausibility of outcomes.

A more game-theoretic approach is taken in [29], where agents can choose when their information should be public or private, in what the authors call "games of influence". Agents try to achieve goals that are expressed as LTL formulas, such as $FGop(A, p)$, meaning a goal that "eventually and henceforth, agent A will have an opinion p".

In [30] a First-Order Computational Tree Logic (FO-CTL) is presented for SIR models (Susceptible, Infectious, or Recovered), which model the spread of diseases. The presence of a single connection in the network is sufficient to spread a disease to an agent. It is shown that under certain conditions, such as a finite agent set, the model checking problem is decidable.

Paper [31] investigates the phenomenon of peer pressure, by introducing a model where a person's preferences are changed in response to the preferences of a "peer group". Relationships among agents are defined within the usual "Facebook logic" [32] definition: connections between pairs of agents that are both symmetric (friendship is always reciprocal) and irreflexive (an agent is not a friend of itself).

Private and public opinions are considered under the dynamics of social influence in [33]. It provides rules for "strong" and "weak" influence, and also rules for when an agent will express its true inner opinions or a different external opinion. Agents are capable of evolving their epistemic knowledge to infer the inner opinions of others, by discarding states which are contradictory to the rules of social influence.

In [2] a "minimal" threshold-model based logic is presented, in the sense that it does not use static modalities or hybrid logic for its diffusion process. The authors go on to introduce an epistemic version of this logic, where agents must have sufficient information about the behaviors of their neighbours before adopting the behavior. The network diffusion dynamics and axiomatization presented in this work serve as basis for the work we develop here, which we expand with the new LTL operators and provide a sound and complete axiomatization for this expanded system.

2.4 SIR infection models

In this section we briefly introduce a compartmental model for epidemics denoted "SIR". Chapters 8 and 9 will explore variations of the logics frameworks presented earlier that are based on these models.

Compartmental models for epidemics have figured in the literature since 1916 [34], first introduced as sets of differential equations for what is today referred to as the SIS (Susceptible-Infectious-Susceptible) model. These compartmental models are usually named after their compartment classes, indicating the evolution of an individual over time in such models. Later [35] proposed the SIR (Susceptible-Infectious-Recovered) model, where infected agents are infectious for one round, and then become recovered and cannot be infected nor spread the disease any further. The presence of a single connection in the network is sufficient to spread a disease from one agent to another. In this section we will focus on SIR models in particular.

SIR models are described by a set of ordinary differential equations [36]. For simplicity, we present the model here without birth and death rate terms:

$$\begin{cases} \frac{dS}{dt} & = -\frac{\beta IS}{N}, \\ \frac{dI}{dt} & = \frac{\beta IS}{N} - \gamma I, \\ \frac{dR}{dt} & = \gamma I \end{cases}$$

where

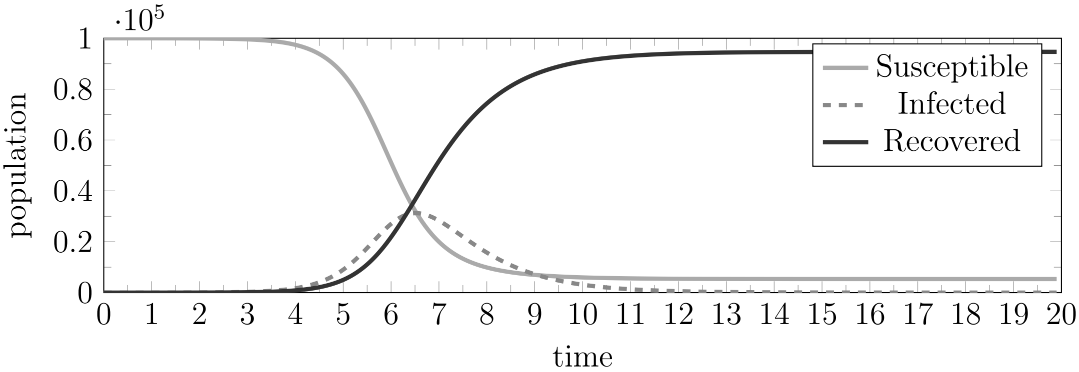
- S is the number of susceptible individuals;
- I is the number of infected individuals;
- R is the number of recovered individuals;
- $N = S + I + R$ is the total population;
- β is the infection rate per unit of time of an individual;
- γ is the rate of recovery per unit of time of an individual.

That is, for an infection rate $\beta = 2$, one infected individual will infect other two individuals on every elapsed unit of time. A recovery rate of $\gamma = 1$ means that, on every elapsed unit of time, one infected individual will recover from infection.

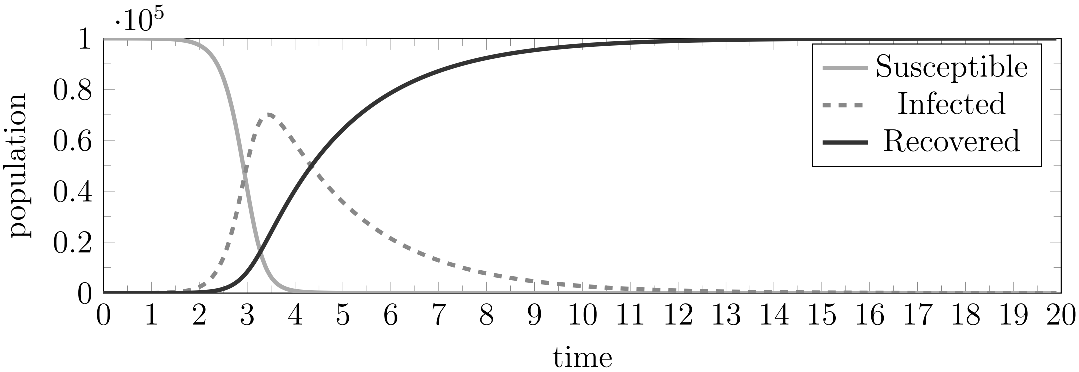
Figure 2.10 demonstrates the evolution over time of an initial population of $N = 100{,}000$ individuals, with a single infected individual at the start.

There has been some work on SIR models in the field of logics.

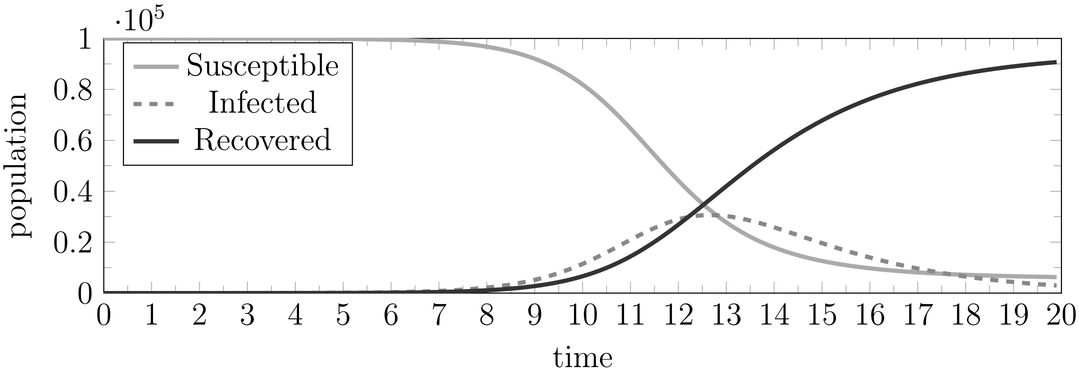
In [37], these models are studied in a sound and complete hybrid logic, and a tableau system is also provided. In this approach, agents are discrete and infections



(a) This case demonstrates a moderately infectious disease with a reasonably quick recovery time. A mild peak of infections occurs, however some agents never become infected. Population $N = 100{,}000$ and a single infected individual at the start. Infection rate is $\beta = 3$, and recovery rate $\gamma = 1$.



(b) This case demonstrates an extremely infectious disease with a slower recovery time. Many agents are quickly infected, and take a long time to recover. Almost every agent is infected. Population $N = 100{,}000$ and a single infected individual at the start. Infection rate is $\beta = 5$, and recovery rate $\gamma = 0.5$.



(c) This case demonstrates a disease that spreads very slowly, and with a slow recovery time. The peak of infections becomes extremely flattened over time. Population $N = 100{,}000$ and a single infected individual at the start. Infection rate is $\beta = 1.5$, and recovery rate $\gamma = 0.5$.

Figure 2.10: Multiple settings showcasing the evolution of SIR models.

obey an actual network structure connecting agents to each other, much like a social network.

In [38] and a section of [39] fuzzy logic approaches to SIR epidemic models are explored. Epidemic and pandemic models are represented as Petri nets in [40], and such models are then simulated in the "PetriNuts" platform tools.

2.5 Formal Model Checking

In certain mission critical applications, such as air traffic control, electronic commerce, communication networks, and others, hardware and software stability and predictability is crucial. Either due to life-threatening risks or economic impacts, finding errors in these applications during their operation is extremely undesirable, and can have serious consequences.

In light of this problem, statically checking program flows has been a point of study for quite some time, with the seminal papers in automatic model checking first appearing in 1981 developed independently by Edmund M. Clarke and E. Allen Emerson [41], and by Jean Queille and Joseph Sifakis [42].

The most common methods used to assess the behavior of a given system include "simulation", "testing", "deductive verification" (or a mathematical proof) and "model checking" [16].

Simulation and testing both consist of verifying if, for a given set of inputs, the system produces the expected output. Commonly, "simulation" refers to running an abstract model of the system, and "testing" to running on an actual implementation of the system [43, 44].

Deductive verification is based on axioms and rules to achieve a static proof of some property or behavior of a system. It requires a specification of the system to be tested in a formal system, and will usually require a specialist to execute the checking, which can be extremely time and resources-consuming, and error-prone [45].

The last method, model checking, will be our focus for the remainder of this work. Model checking consists of, at first glance, exhaustively checking the state space of a system to determine whether some property is true or false on the system.

As one might guess, naïvely checking the entire space of states of a given system can quickly become an intractable problem. There are, however, techniques such as Binary Decision Diagrams (BDDs) [46] and Bounded Model Checking (BMC) [47] to intelligently reduce the search space and improve speed of verification for certain types of models. These techniques are collectively referred to as "symbolic model checking".

During the modelling phase, the system's variables and transitions must be modelled in the formal logic being used, usually as a finite model; In the specification phase, the properties that must be checked are devised and written as formulas in the logic; Finally, in the checking phase the specifications are tested one by one, and a sequence of steps (known as "traces") are provided as counterexamples for the false ones [48].

Chapter 3

Linear Temporal Logic for Social Networks

In this chapter we introduce the Linear Temporal Logic for Social Networks (LTL-SN), define its model, language and semantics, a system for which we will present a sound and complete axiomatization in chapter 4.

The reasoning behind using a Linear Temporal Logic for this work is twofold. When looking at the spread of behaviors in a social network, we are mostly interested in the dynamics of spreading. Such dynamics are governed by the states of each of the agents in the network, and the connections between them which serve as the medium for information propagation over time. Temporal logics are a well established family of logic systems for modelling such dynamics that evolve over time, therefore their application in this context makes much sense.

Moreover, LTL provides a natural way to express causality and correlation between events in the network. It can be used to specify temporal patterns, such as "if event A happens, then event B must happen within a certain time period", or "event A and event B are correlated and tend to happen together". This allows for the modelling of complex social phenomena such as information diffusion, group formation, and the emergence of collective behavior.

The second reason to favor the use of LTL is the extensive backlog of model checking tools available. Tools such as nuSMV, nuXmv, mCRL2, PAT and LSTmin are all freely available.

The logic presented in this work is largely inspired by the dynamics introduced in [2], ported over to LTL which allows one to write succinct formulas reasoning about paths. We also borrow much inspiration from [37], based on which we explore another type of diffusion applied to infection spread models, later in chapters 8, 9 and 10.

LTL-SN is effectively a special case of LTL models, that give rise to the particular type of behavior propagation we are interested in capturing. As such, the standard set of LTL axioms is not complete, a gap that we fill with reduction axioms which allows us to prove completeness by a reduction of the logic into its static part.

3.1 Language and semantics

In this section we present the language of LTL-SN, define certain properties of its models and its semantics. The language presented in definition 6 is very similar to standard LTL, and they read and behave like such. The usual operators X for "Next", F for "Finally", G for "Globally" and U for "Until" are all available. The propositional variables, however are restricted to the concepts related to the shape of social networks and the behaviors agents exhibit. These are expressed, respectively, by the operators $N_a b$ and β_a, for given agents a and b.

Definition 6 (Language)**.** *The language of LTL-SN is given in Backus-Naur Form (BNF) as follows:*

$$\varphi := \top \mid N_{ab} \mid \beta_a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid X\varphi \mid \varphi_1 U \varphi_2$$

where $a, b \in \mathcal{A}$, a finite set of agents.

The remaining operators can be defined via abbreviations: $\bot \equiv \neg\top$, $\varphi \vee \phi \equiv \neg(\neg\varphi \wedge \neg\phi)$, $\varphi \to \phi \equiv \neg(\varphi \wedge \neg\phi)$, $F\varphi \equiv \top U \varphi$ and $G\varphi \equiv \neg F \neg\varphi$. Intuitive readings of the operators are the same as previously introduced in definition 1.

In this chapter we model the evolution of a single behavior in a social network, but it is easy to expand this formalism for multiple behaviors, which we will present in chapter 6.

Before we proceed to the models used in this logic, we must first define the concept of behavior sets. Behavior sets represent the state of a behavior in the network, by containing the agents which currently exhibit such behavior.

Definition 7 (Behavior sets)**.** *A behavior set B is the set of agents that exhibit the behavior of the network. The set of all behavior sets is denoted $\mathcal{B} \subseteq 2^{\mathcal{A}}$.*

A social network model for this logic is composed of a set of agents, and a relationship between them, which we assume is symmetric and irreflexive following the usual "Facebook logic" characteristics as we mentioned previously. That is, in an LTL-SN model, if N_{ab} holds, N_{ba} must hold as well. Being irreflexive means an agent cannot be a friend of itself, and therefore N_{aa} can never hold in such a model.

A model also contains an initial behavior set, that determines which agents exhibit the behavior at the start. The network is then allowed to evolve over time, which here is represented as a sequence of discrete "positions", over paths which will be defined later on in definition 10.

Finally, a model is also defined by an adoption threshold, which is the minimum proportion of an agent's neighbors that must exhibit the behavior so that the agent itself may adopt it.

The model definition presented here is very similar to the one in [2], however in our case we will extend the model structure to include the concept of "paths", which will allow us to traverse it using the temporal logic operators.

Definition 8 (Model)**.** *A model is a tuple $\mathcal{M} = (\mathcal{A}, N, \theta, I)$, where*

- *$\mathcal{A}$ is a finite set of agents;*
- *N is a neighborhood function $N : \mathcal{A} \mapsto 2^{\mathcal{A}}$ such that N is irreflexive, serial and symmetric;*
- *$\theta \in (0, 1]$ is a non-zero positive uniform adoption threshold;*
- *$I \in \mathcal{B}$ is a non-empty initial behavior set;*

Before we introduce the aforementioned paths, we must first define how such paths can be formed in the first place. Paths are defined over a strict relation between states (behavior sets), such that the network evolves in a manner which conforms with the rules of behavior diffusion. This relation, which we call $\leq$, is presented in definition 9.

Definition 9 (Relation $\leq$)**.** *Let $\leq$ be the binary relation over $\mathcal{B} \times \mathcal{B}$, where*

$$B \leq B' \text{ iff } B' = B \cup \left\{ a \in \mathcal{A} : \frac{|N(a) \cap B|}{|N(a)|} \geq \theta \right\}$$

Two behavior sets B and B', are related such that $B \leq B'$ when B' is the immediate result of the behavior diffusion process over B. It is worth noting that due to the seriality of the network, $N(a)$ is always non-empty, and therefore B' is always well defined. With this relation we are finally able to define paths over a model, representing the evolution of behaviors on the network. Notice this relation only permits adoption of new behaviors by agents, without unadoption. See [27], for instance, for a more extensive look on the mechanics of unadoption. In chapter 10 we also explore compartmental models of infections with agent recovery (SIRS models), in which a mechanic similar to unadoption occurs.

Definition 10 (Path on $\mathcal{M}$)**.** *A path $b = b_0, b_1, b_2, \ldots$ on a model $\mathcal{M} = (\mathcal{A}, N, \theta, I)$, where $b_i \in \mathcal{B}$ for all i, is such that $b_0 = I$ and $b_i \leq b_{i+1}$.*

Given that paths are strictly defined by the relation $\leq$, it's not particularly hard to see that such paths are unique for a given starting behavior set.

Lemma 1 (Path uniqueness)**.** *There is exactly one path b such that $b_0 = I$.*

Proof. Follows directly from the fact that for any b_i the relation $\leq$ defines strictly one b_{i+1} such that $b_i \leq b_{i+1}$. $\square$

From lemma 1, we see that all the information necessary to define satisfaction of formulas over a model $\mathcal{M}$ is a position i.

Definition 11 (Satisfaction)**.** *Given a model $\mathcal{M} = (\mathcal{A}, N, \theta, I)$ and its path $b = b_0, b_1, b_2, \ldots, b_i, \ldots$, the notion of satisfaction of formulas in $\mathcal{M}$ at a position i is defined as follows:*

$$\begin{array}{lll}
\mathcal{M}, i \models \beta_a & \textit{iff} & a \in b_i \\
\mathcal{M}, i \models N_{ax} & \textit{iff} & x \in N(a) \\
\mathcal{M}, i \models \neg\varphi & \textit{iff} & \mathcal{M}, i \not\models \varphi \\
\mathcal{M}, i \models \varphi_1 \wedge \varphi_2 & \textit{iff} & \mathcal{M}, i \models \varphi_1 \textit{ and } \mathcal{M}, i \models \varphi_2 \\
\mathcal{M}, i \models X\varphi & \textit{iff} & \mathcal{M}, i+1 \models \varphi \\
\mathcal{M}, i \models \varphi U \psi & \textit{iff} & \textit{there exists a } n \geq 0 \textit{ such that} \\
 & & \mathcal{M}, i+n \models \psi \textit{ and } \mathcal{M}, i+j \models \varphi \textit{ for all } 0 \leq j < n
\end{array}$$

All LTL operators, plus propositional operators N_{ax} and β_a defined in a standard way.

Finally, we define the concept of "validity", which will be important when we proceed to deal with soundness of the logic.

Definition 12 (Validity)**.** *$\models \varphi$ iff $\mathcal{M}, i \models \varphi$ for all models $\mathcal{M}$ and positions i.*

3.2 Examples

The diffusion of a behavior in a social network is a temporal process, and as such the LTL formalism makes it notably easier to express properties related directly to the behavior diffusion mechanism and the network structure, by leveraging the Until operator.

We will use two minimal examples of networks to illustrate this. In both cases the behavior being studied is only expressed by agent a at first, but the slightly different network structures lead to quite different outcomes.

For instance, consider the formula $\neg\beta_d U(\beta_b \vee \beta_f)$, which tells us that agent d will not have adopted the behavior until either agent b or f do. This expression holds in the network configuration shown in figure 3.1, independently of the initial behavior set. Notice the similar formula, $\neg\beta_d U(\beta_b \vee \beta_e \vee \beta_f)$, holds in figure 3.2 but agent d never gets to incorporate the behavior, since none of its friends do.

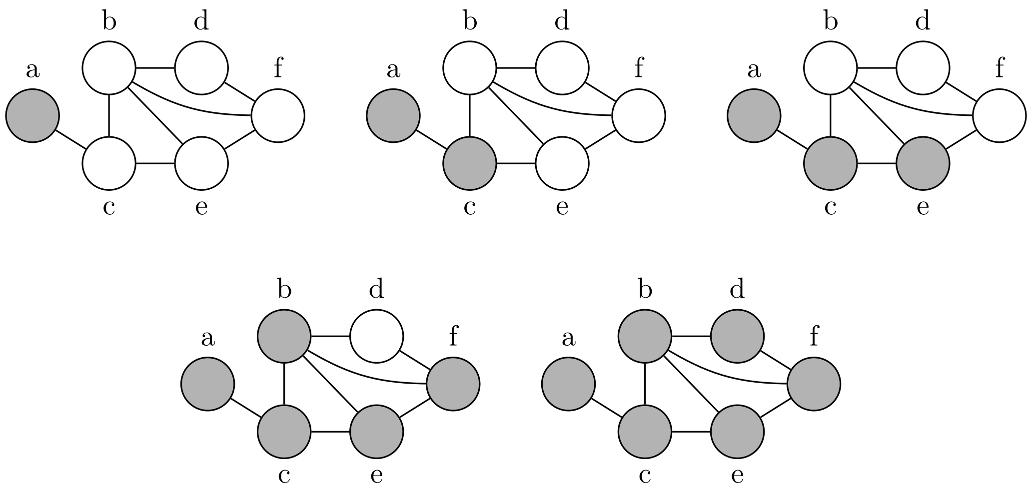


Figure 3.1: This figure presents a social network (with $\theta = 1/3$) with some agents and the evolution of a behavior. Nodes are agents, and the edges represent the relation between them. Each figure, ordered from left to right and top to bottom, represents a distinct position as the network evolves. The behavior eventually dominates the network.

It is also worth pointing out that the expression for cascades from [2], which holds if all agents eventually adopt the behavior, can be expressed here as

$$cascade = F \bigwedge_{a \in \mathcal{A}} \beta_a$$

It holds on every position of the network in figure 3.1, but does not hold anywhere in figure 3.2 since some agents never get to incorporate the behavior.

Now we look at a formula using the Next operator: $(N_{bd} \wedge \beta_b) \vee (N_{fd} \wedge \beta_f) \rightarrow X\beta_d$ holds in figure 3.1 (and vacuously in figure 3.2), and also holds independently of the initial behavior set. Notice that the left-hand side expresses the necessary conditions for agent d to incorporate the behavior itself. A generalization of formulas like this one will be present in the axiomatization in section 4.

Lastly, it is also easy to express global properties, such as $G\neg\beta_d$ meaning "agent d never exhibits the behavior", which holds in figure 3.2. This expression does not hold in the network of figure 3.1 because, as we have seen before, it cascades.

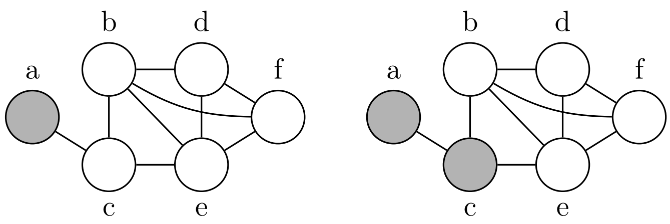


Figure 3.2: Propagation of the behavior is stopped by a "cluster" [2] of agents ($\theta = 1/3$). The cluster is a large group of interconnected agents, which provide a difficult "barrier" for behavior diffusion.

Chapter 4

Axioms for the LTL-SN system

In this section the axioms of LTL-SN are presented. This formalism is based on reduction axioms, and this strategy allows us to ultimately reduce formulas into propositional logic. However before we can do that, we need to define a few abbreviations and enunciate some useful properties.

First, we will define the "Adoption" abbreviation, which is the same as used in [2]. This formula holds when an agent will adopt the behavior in the next position of the path, because it holds when a fraction of at least θ of its neighbors have already adopted the behavior as stated in definition 9.

Definition 13 (Adoption)**.** *An abbreviation expressing the adoption threshold:*

$$\beta_{N(a)\geq\theta} \coloneqq \bigvee_{\mathcal{G}\subseteq\mathcal{N}\subseteq\mathcal{A}:\frac{|\mathcal{G}|}{|\mathcal{N}|}\geq\theta} \left(\bigwedge_{b\in\mathcal{N}} N_{ab} \wedge \bigwedge_{b\notin\mathcal{N}} \neg N_{ab} \wedge \bigwedge_{b\in\mathcal{G}} \beta_b \right)$$

We also define an expansion that is equivalent to the Until operator, and demonstrate that both are indeed semantically equivalent. This expansion will play a key part in the axiomatization, because it allows us to reduce Until occurrences into equivalent finite expressions containing only Next operators. In turn, these occurrences can be further reduced using the adoption abbreviation.

This approach makes intuitive sense: each position of the network is determined by the relation $\leq$, which is deterministic and produces unique paths. Therefore, it is reasonable to expect that the satisfaction of a formula can be verified over the initial state of the network.

The Until expansion is broken in two different definitions: definition 14 is a recurrence over which the expansion, in definition 15, is established with.

Definition 14 (Until expansion steps)**.** *Let $\varphi U\psi$ be a formula. We define its Until* *expansion steps as*

$$u^0(\varphi U\psi) \coloneqq \psi$$
$$u^i(\varphi U\psi) \coloneqq (\varphi \wedge X u^{i-1})$$

Definition 15 (Until expansion)**.** *The Until expansion $EXP_U(\varphi U\psi)$ in a model with agent set $\mathcal{A}$ is defined as*

$$EXP_U(\varphi U\psi) \coloneqq \bigvee_{0 \leq i \leq |\mathcal{A}|} u^i(\varphi U\psi)$$

Before we can show equivalence of the Until expansion and the Until operator, we demonstrate that $u^n(\varphi U\psi)$ is equivalent to $\varphi U\psi$ if and only if ψ holds before the next n positions (or vacuously never holds). After that, we also determine that the longest distance from the current position at which this can happen has to be less than $|\mathcal{A}|$ positions ahead. With both of these, we are able to prove the equivalence.

Lemma 2 (Until expansion step satisfaction)**.** *$\mathcal{M}, i \models u^n(\varphi U\psi)$ if and only if $\mathcal{M}, i+n \models \psi$ and $\mathcal{M}, i+j \models \varphi$ for all $0 \leq j < n$.*

Proof. We will abbreviate $u^n(\varphi U\psi)$ as u^n for this proof. The proof goes by induction on n.

Base case ($n = 0$):

It is trivial to see that $\mathcal{M}, i \models u^0$ if and only if $\mathcal{M}, i \models \psi$, and there is no j such that $0 \leq j < n$.

Induction hypothesis ($n = k$):

$\mathcal{M}, i \models u^k$ if and only if $\mathcal{M}, i+k \models \psi$, and $\mathcal{M}, i+j \models \varphi$ for all $0 \leq j < k$.

Induction step ($n = k+1$):

Assume $\mathcal{M}, i \models u^{k+1}$.

By definition we have that $\mathcal{M}, i \models \varphi \wedge Xu^k$, if and only if $\mathcal{M}, i \models \varphi$ and $\mathcal{M}, i \models Xu^k$. $\mathcal{M}, i \models Xu^k$ if and only if $\mathcal{M}, i+1 \models u^k$.

Without loss of generality, we can rename i to $i+1$ in the induction hypothesis, and then we have $\mathcal{M}, i+1 \models u^k$ if and only if $\mathcal{M}, i+1+k \models \psi$, and $\mathcal{M}, i+1+j \models \varphi$ for all $0 \leq j < k$.

$\mathcal{M}, i+1+j \models \varphi$ for all $0 \leq j < k$ and $\mathcal{M}, i \models \varphi$ can be rewritten as $\mathcal{M}, i+j \models \varphi$ for all $0 \leq j < k+1$.

Therefore we are able to conclude that $\mathcal{M}, i+(k+1) \models \psi$ and $\mathcal{M}, i+j \models \varphi$ for all $0 \leq j < k+1$, and that finishes our proof. □

One last point we must address is the actual equivalence between the Until operator and the expansion. While the expansion is intuitive, it relies on the fact that paths always reach a fixed-point, where no further changes happen to the next behavior sets. This is important, otherwise the expansion could potentially require infinite steps. We can show that, in fact, any path will reach a fixed-point after at most $|\mathcal{A}|$ steps.

Lemma 3 (Fixed-point)**.** *Let $b = b_0, b_1, b_2, \ldots, b_n$ be a path. For some $i < |\mathcal{A}|$ a fixed-point $b_i = b_{i+1}$ is reached.*

Proof. This follows directly from the fact that $\mathcal{A}$ is finite and $b_i \subseteq b_{i+1}$ for all i by definition. The slowest possible diffusion scenario is where $|b_0| = 1$ and only one agent adopts the behavior per position on the path until the fixed-point is reached. It follows directly that the fixed-point will have to be reached for some $i < |\mathcal{A}|$. □

With lemma 3 we can finally show equivalence between the Until operator and the Until expansion.

Lemma 4 (Until expansion soundness)**.**

$$\models \varphi U \psi \leftrightarrow EXP_U(\varphi U \psi)$$

Proof. By definition, $EXP_U(\varphi U \psi) = u^0(\varphi U \psi) \vee u^1(\varphi U \psi) \vee \ldots \vee u^{|\mathcal{A}|}(\varphi U \psi)$.

For any model $\mathcal{M}$ and position j, by lemma 2 we know that a term u^i holds if and only if ψ holds in position $j + i$. By lemma 3 we know that if $\varphi U \psi$ holds, ψ must hold at some position i such that $0 \leq j + i \leq |\mathcal{A}| + j$. Therefore $\models \varphi U \psi$ if and only if $\models EXP_U(\varphi U \psi)$. □

LTL axioms are as seen on [49], presented here on table 4.1. Notice that for completeness of this particular class of models, only axioms $A2$, $A3$, $A8$ and the rules of inference MP and NX were used. The others were kept merely for reference.

This is also an appropriate time to introduce two propositions, which will later be used in the completeness proof.

Proposition 1 (Next rule)**.** *The following inference rule is admissible:*

$$\frac{\vdash \phi \leftrightarrow \psi}{\vdash X\phi \leftrightarrow X\psi}$$

Proposition 2 (Replacement of equivalents)**.** *If* $\vdash \phi_1 \leftrightarrow \phi_2$*, then* $\vdash [\phi_1/\psi]\chi \leftrightarrow [\phi_2/\psi]\chi$.

The proof of proposition 1 follows straightforward from axiom A3 and inference rules NX and MP. The proof of proposition 2 follows by induction on the construction of χ. These proofs are rather standard in LTL literature [50].

Network axioms	
$\neg N_{aa}$	Irreflexivity
$N_{ab} \leftrightarrow N_{ba}$	Symmetry
$\bigvee_{b \in \mathcal{A}} N_{ab}$	Seriality
Reduction axioms	
$XN_{ab} \leftrightarrow N_{ab}$	Red.Ax.$X.N$
$X\beta_a \leftrightarrow \beta_a \vee \beta_{N(a) \geq \theta}$	Red.Ax.$X.\beta$
$\varphi U \psi \leftrightarrow EXP_U(\varphi U \psi)$	Red.Ax.U
LTL axioms	
All classical propositional tautologies	
$G(\varphi \rightarrow \psi) \rightarrow (G\varphi \rightarrow G\psi)$	A1
$\neg X\varphi \leftrightarrow X\neg\varphi$	A2
$X(\varphi \rightarrow \psi) \rightarrow (X\varphi \rightarrow X\psi)$	A3
$G(\varphi \rightarrow X\varphi) \rightarrow (\varphi \rightarrow G\varphi)$	A4
$(\varphi U \psi) \leftrightarrow \psi \vee (\varphi \wedge X(\varphi U \psi))$	A5
$(\varphi U \psi) \rightarrow F\psi$	A6
$\varphi U(\psi_1 \vee \psi_2) \leftrightarrow \varphi U \psi_1 \vee \varphi U \psi_2$	A7
$X(\varphi \wedge \psi) \leftrightarrow X\varphi \wedge X\psi$	A8
$\dfrac{\varphi \qquad \varphi \rightarrow \psi}{\psi}$	MP
$\dfrac{\vdash \varphi}{\vdash G\varphi}$	NG
$\dfrac{\vdash \varphi}{\vdash X\varphi}$	NX

Table 4.1: LTL-SN axioms.

Chapter 5

Soundness and completeness

Soundness and completeness are two important properties in a logic system. Soundness refers to the property that a logical system only proves statements that are true, i.e., all provable statements are valid. Completeness, on the other hand, refers to the property that a logical system can prove all valid statements, i.e., all valid statements are provable.

These properties are important because they provide a guarantee of the reliability and accuracy of a logical system. Soundness ensures that the system does not prove false statements, which would undermine its usefulness in making decisions or drawing conclusions. Completeness, on the other hand, ensures that the system is not missing any valid statements, which would limit its expressive power and potentially lead to incorrect conclusions.

In practical terms, soundness and completeness are important for ensuring the correctness and effectiveness of automated reasoning systems, such as theorem provers, model checkers, and automated decision-making systems. A sound and complete logical system provides a strong foundation for these systems, allowing them to make accurate and reliable decisions based on the logical rules and constraints encoded in the system.

In this chapter we will prove soundness and completeness properties of the LTL-SN axiomatization presented. As is usually the case, the soundness proof is quite straightforward. The completeness proof, however, is more elaborate and the approach used is a two-step translation of LTL-SN formulas into propositional ones.

Since the formulas of LTL-SN are evaluated over conventional LTL paths (or "flows of time" [51, 52]), all standard LTL axioms, as well as the inference rule, are also sound in this logic.

Lemma 5 (Soundness)**.** *All axiom schemata presented in table 4.1 are valid, per definition 12.*

Proof. For this proof, let $\mathcal{M}$ be an arbitrary LTL-SN model and $B \in \mathcal{B}$ a behavior set. Consider each case:

- All network axioms follow directly from definition 8;
- $\mathcal{M}, i \models XN_{ab} \leftrightarrow N_{ab}$ follows directly from the fact that $\leq$ never alters the network structure;
- $\mathcal{M}, i \models \varphi U \psi \leftrightarrow EXP_U(\varphi U \psi)$ follows directly from lemma 4;
- For $\mathcal{M}, i \models X\beta_a \leftrightarrow \beta_a \vee \beta_{N(a)\geq\theta}$, consider $B = b_i$ and a B' such that $B \leq B'$. $\mathcal{M}, i \models X\beta_a$ iff $\mathcal{M}, i+1 \models \beta_a$ iff $a \in B' = B \cup \left\{ b \in \mathcal{A} : \frac{|N(b) \cap B|}{|N(b)|} \geq \theta \right\}$ iff $\mathcal{M}, i \models \beta_a$ or $a \in \left\{ b \in \mathcal{A} : \frac{|N(b) \cap B|}{|N(b)|} \geq \theta \right\}$.

  It can be shown that the large disjunct in definition 13 is satisfied iff $a \in \left\{ b \in \mathcal{A} : \frac{|N(b) \cap B|}{|N(b)|} \geq \theta \right\}$ just as in [2].

  Hence $\mathcal{M}, i \models X\beta_a$ iff $\mathcal{M}, i \models \beta_a$ or $\mathcal{M}, i \models \beta_{N(A)\geq\theta}$.

$\square$

5.1 Completeness

The strategy for this completeness proof is a two-step translation of LTL-SN formulas into propositional formulas. The first step replaces occurrences of the Until operator in formulas, and the second step replaces Next operator occurrences. Both translations need proofs of equivalence, and each of these are achieved by induction.

As previously mentioned in chapter 4, this approach makes intuitive sense. The network evolution has strict rules, and its evolution is determined by the initial state.

Next, we define the first translation necessary to convert a formula into the static part of the logic. This translation is responsible for the Until occurrences, using the expansion from definition 15.

Definition 16 (Until Translation)**.**

$$\begin{aligned} t_u(N_{ab}) &= N_{ab} \\ t_u(\beta_a) &= \beta_a \\ t_u(\varphi \wedge \psi) &= t_u(\varphi) \wedge t_u(\psi) \\ t_u(\neg\varphi) &= \neg t_u(\varphi) \\ t_u(X\varphi) &= X t_u(\varphi) \\ t_u(\varphi U \psi) &= t_u(EXP_U(\varphi U \psi)) \end{aligned}$$

Next lemma assures that every formula is equivalent to another formula without any occurrence of the Until operator.

Lemma 6. *For an LTL-SN formula φ:*

$$\vdash \varphi \leftrightarrow t_u(\varphi)$$

Proof of this lemma is in appendix A.

Now we proceed to the second step of the translation process, with this one being responsible for occurences of the X operator.

Definition 17 (Translation)**.** *Using the reduction axioms it is possible to translate an LTL-SN formula with no Until occurrences into propositional logic.*

$$\begin{aligned}
t(N_{ab}) &= N_{ab} \\
t(\beta_a) &= \beta_a \\
t(\varphi \wedge \psi) &= t(\varphi) \wedge t(\psi) \\
t(\neg\varphi) &= \neg t(\varphi) \\
t(XN_{ab}) &= N_{ab} \\
t(X\beta_a) &= t(\beta_a \vee \beta_{N(a)\geq\theta}) \\
t(X(\varphi \wedge \psi)) &= t(X\varphi \wedge X\psi) \\
t(X\neg\varphi) &= t(\neg X\varphi) \\
t(XX\varphi) &= t(Xt(X\varphi))
\end{aligned}$$

Translation inspired by the ones from [53, 54].

Finally we define a cost measure for formulas of the language, a very commonplace method in literature, which enables us to prove the translation equivalence by induction over cost, instead of subformulas.

Definition 18 (Cost measure).

$$\begin{aligned} c(\beta_a) &= 1 \\ c(N_{ab}) &= 1 \\ c(\neg\varphi) &= 1 + c(\varphi) \\ c(\varphi_1 \wedge \varphi_2) &= 1 + max(c(\varphi_1), c(\varphi_2)) \\ c(X\beta_a) &= 4 + 2 \cdot |\mathcal{A}|^3 \\ c(X\psi) &= 2 \cdot c(\psi) \end{aligned}$$

where $X\psi$ matches any formula besides those of form $X\beta_a$.

Lemma 7 (Cost of the adoption abbreviation). *The cost of the adoption abbreviation $c(\beta_{N(a)\geq\theta})$ in a model with agent set $\mathcal{A}$ is $2 \cdot |\mathcal{A}|^3$.*

Proof. For ease of reading we reproduce the adoption abbreviation here:

$$\beta_{N(a)\geq\theta} := \bigvee_{\mathcal{G}\subseteq\mathcal{N}\subseteq\mathcal{A}:\frac{|\mathcal{G}|}{|\mathcal{N}|}\geq\theta} \left(\bigwedge_{b\in\mathcal{N}} N_{ab} \wedge \bigwedge_{b\notin\mathcal{N}} \neg N_{ab} \wedge \bigwedge_{b\in\mathcal{G}} \beta_b \right)$$

The outer disjunction ranges over two groups, $\mathcal{G} \subseteq \mathcal{N} \subseteq \mathcal{A}$, therefore in the worst case there will be as many terms as $|\mathcal{A}|^2$ for each possible combinations of both. Now analyzing each term, the first two conjunctions contribute at most $|\mathcal{A}|$ propositions, and the last conjunction another $|\mathcal{A}|$. Therefore each term contributes, in the worst case, a cost of $2 \cdot |\mathcal{A}|$. Therefore, the total cost is $2 \cdot |\mathcal{A}|^3$. □

Lemma 8 (Translation costs). *For all φ, ψ:*

1. $c(\varphi) \quad \geq c(\psi)$ *if* $\psi \in Sub(\varphi)$
2. $c(XN_{ab}) \quad > c(N_{ab})$
3. $c(X\beta_a) \quad > c(\beta_a \vee \beta_{N(a)\geq\theta})$
4. $c(X(\varphi \wedge \psi)) > c(X\varphi) \wedge c(X\psi)$
5. $c(X\neg\varphi) \quad > \neg c(X\varphi)$
6. $c(XX\varphi) \quad > c(Xt(X\varphi))$

Proof of this lemma is in appendix B.

Lemma 9 (Translation equivalence). *For an LTL-SN formula φ with no Until occurrences:*

$$\vdash \varphi \leftrightarrow t(\varphi)$$

Proof of this lemma is in appendix C.

Theorem 1 (Completeness)**.** *For every* $\varphi \in \mathcal{LTL} - \mathcal{SN}$*:*

$$\models \varphi \textit{ implies } \vdash \varphi$$

Proof. Suppose $\models \varphi$. By lemma 6 and lemma 9 and since the proof system is sound, we have that $\models t(t_u(\varphi))$. The formula $t(t_u(\varphi))$ is propositional, therefore $\mathcal{PROP}(\mathcal{LTL} - \mathcal{SN}) \vdash t(t_u(\varphi))$ by completeness of propositional logic and by the network axioms. We also have that $\mathcal{LTL} - \mathcal{SN} \vdash t(t_u(\varphi))$ since $\mathcal{PROP}(\mathcal{LTL} - \mathcal{SN})$ is a subsystem of LTL-SN. Again by lemma 9 we have that $\mathcal{LTL} - \mathcal{SN} \vdash \varphi$. □

Chapter 6

Generalization for multiple behaviors and weighted relationships

The logic presented in chapter 3 models a single behavior spreading across the network. In this chapter, we introduce a simple generalization of LTL-SN supporting multiple behaviors simultaneously. We redefine the language and semantics, such that operator β_a can be applied to any behavior from a set of behaviors B.

Definition 19 (Model)**.** *A model is a tuple $\mathcal{M} = (\mathcal{A}, \mathsf{B}, N, \theta, I)$, where*

- *$\mathcal{A}$ is a finite set of agents (a, b, c, ...);*
- *B is a finite set of behaviors (α, β, γ, ...);*
- *N is the weight function $N : \mathcal{A} \times \mathcal{A} \mapsto \mathbb{N}_0$ such that:*

    $N(a, a) = 0$;

    For all a, $N(a, b) > 0$ for some $b \neq a$;
- *$\theta \in (0, 1]$ is a non-zero positive uniform adoption threshold;*
- *I is a function $\mathsf{B} \mapsto 2^{\mathcal{A}}$ defining the initial behaviors for all agents;*

Also consider $N_a(B) := \sum_{b \in B} N(a, b)$ where $B \in 2^{\mathcal{A}}$. Notice $N_a(\mathcal{A})$ is the sum of all weights of a's relationships.

Definition 20 (Relation $\leq$)**.** *Let $\leq$ be the binary relation over $2^{\mathcal{A}} \times 2^{\mathcal{A}}$, such that*

$$B \leq B' \textit{ iff } B' = B \cup \left\{ a \in \mathcal{A} : \frac{N_a(B)}{N_a(\mathcal{A})} \geq \theta \right\}$$

where $B \in 2^{\mathcal{A}}$.

That is, two behavior sets B and B', are related such that $B \leq B'$ when B' is the immediate result of the behavior diffusion process over B. With this relation we are able to define paths on a model, which represent the evolution of behaviors on the network. Notice this relation only permits adoption of new behaviors by agents, without unadoption. See [27] for a more extensive look on the mechanics of unadoption, for instance.

Definition 21 (Behaviors). *$\mathcal{B} \subseteq 2^{\mathcal{A}}$ is the set of behavior sets.*

Definition 22 (Path on $\mathcal{M}$). *We define a path $b = b_0, b_1, b_2, \dots$ on a model $\mathcal{M} = (\mathcal{A}, \mathsf{B}, N, \theta, I)$ to be such that $b_0 = I$ and for all $\alpha \in \mathsf{B}$, $b_i(\alpha) \leq b_{i+1}(\alpha)$.*

Lemma 10 (Path uniqueness). *There is exactly one path b such that $b_0 = I$.*

Proof. Follows directly from the fact that for any $b_i(\alpha)$ the relation $\leq$ defines strictly one $b_{i+1}(\alpha)$ such that $b_i(\alpha) \leq b_{i+1}(\alpha)$. $\square$

Given lemma 10, we can define satisfaction for a model directly over paths.

Definition 23 (Language). *The language of LTL-SN is given by the following BNF:*

$$\varphi := \top \mid N_{ab} \mid \alpha_a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid X\varphi \mid \varphi_1 U \varphi_2$$

where $a, b \in \mathcal{A}$, $\alpha \in \mathsf{B}$.

We use the standard abbreviations $\bot \equiv \neg\top$, $\varphi \vee \phi \equiv \neg(\neg\varphi \wedge \neg\phi)$, $\varphi \to \phi \equiv \neg(\varphi \wedge \neg\phi)$, $F\varphi \equiv \top U \varphi$ and $G\varphi \equiv \neg F \neg\varphi$. Intuitive readings of the operators are the same as previously introduced in definition 1.

Definition 24 (Satisfaction). *Given a model $\mathcal{M} = (\mathcal{A}, \mathsf{B}, N, \theta, I)$ and its path b. The notion of satisfaction of formulas in $\mathcal{M}$ at a position i is defined as follows:*

$$\begin{array}{lll}
\mathcal{M}, i \models \alpha_a & \textit{iff} & a \in b_i(\alpha) \\
\mathcal{M}, i \models N_{ab} & \textit{iff} & N(a,b) > 0 \\
\mathcal{M}, i \models \neg\varphi & \textit{iff} & \mathcal{M}, i \not\models \varphi \\
\mathcal{M}, i \models \varphi_1 \wedge \varphi_2 & \textit{iff} & \mathcal{M}, i \models \varphi_1 \textit{ and } \mathcal{M}, i \models \varphi_2 \\
\mathcal{M}, i \models X\varphi & \textit{iff} & \mathcal{M}, i+1 \models \varphi \\
\mathcal{M}, i \models \varphi U \psi & \textit{iff} & \textit{there exists a } n \geq 0 \textit{ such that} \\
 & & \mathcal{M}, i+n \models \psi \textit{ and } \mathcal{M}, i+j \models \varphi \textit{ for all } 0 \leq j < n
\end{array}$$

All LTL operators, plus propositional operators N_{ab} and α_a defined in a standard way.

Definition 25 (Validity). *$\models \varphi$ iff $\mathcal{M}, i \models \varphi$ for all models $\mathcal{M}$ and positions i.*

Definition 26 (Generalized majority). *An abbreviation representing "majority":*

$$\alpha_{N(a)\geq\theta} \coloneqq \bigvee_{\mathcal{G}\subseteq\mathcal{N}\subseteq\mathcal{A}:\frac{W_a(\mathcal{G})}{W_a(\mathcal{A})}\geq\theta} \left(\bigwedge_{b\in\mathcal{N}} N_{ab} \wedge \bigwedge_{b\notin\mathcal{N}} \neg N_{ab} \wedge \bigwedge_{b\in\mathcal{G}} \alpha_b \right)$$

6.1 Expressiveness

This logic is built over LTL, usually of first-order logic expressiveness for discrete-time initial models [55]. However, this formalism is restricted over a particular class of models for social networks, and as such yields a logic of reduced expressiveness, equivalent to that of propositional logic (which is, in fact, the argument used for the completeness proof). This is not unlike [2], also translatable into propositional formulas.

The defining characteristic of these models is the fact that their entire trace is ultimately defined by their initial conditions. This observation enables the expansion of the non-static operators into static ones, and with that the reduction of either dynamic or temporal parts of these logics into their static parts.

Notice, however, this is not the case in the hybrid logic of [37]. While this paper also employs a reduction to its static part, that part is not propositional in nature. The hybrid formalism, which allows naming agents and therefore agents that are reflective about their own states, still retains additional expressiveness.

6.2 Examples

In figure 6.1 agent a represents an important and reliable broadcaster (like a TV channel, for instance) spreading the behavior α. Notice a is not a friend of the other agents. Figure 6.2 shows a strong cluster of agents resisting to incorporate a behavior.

Notice the formula $(N_{fa} \wedge \alpha_a) \vee (N_{fd} \wedge \alpha_d \wedge N_{fc} \wedge \alpha_c \wedge N_{fe} \wedge \alpha_e) \to X\alpha_f$ holds in both of them at every position, and it determines how agent f is able to incorporate the behavior α itself. A generalization of formulas like this one will be used in the axiomatization present in section 4.

For an example of an Until formula, consider $\neg\alpha_e U \alpha_c$. It holds in figure 6.1, and expresses the fact that agent e will not have adhered to the behavior until his friend agent c did. Notice it does not hold in figure 6.2 because e never gets to incorporate the behavior. Considering the previous statement, it is easy to see that, for instance, $G\neg\alpha_e$ holds only in figure 6.2.

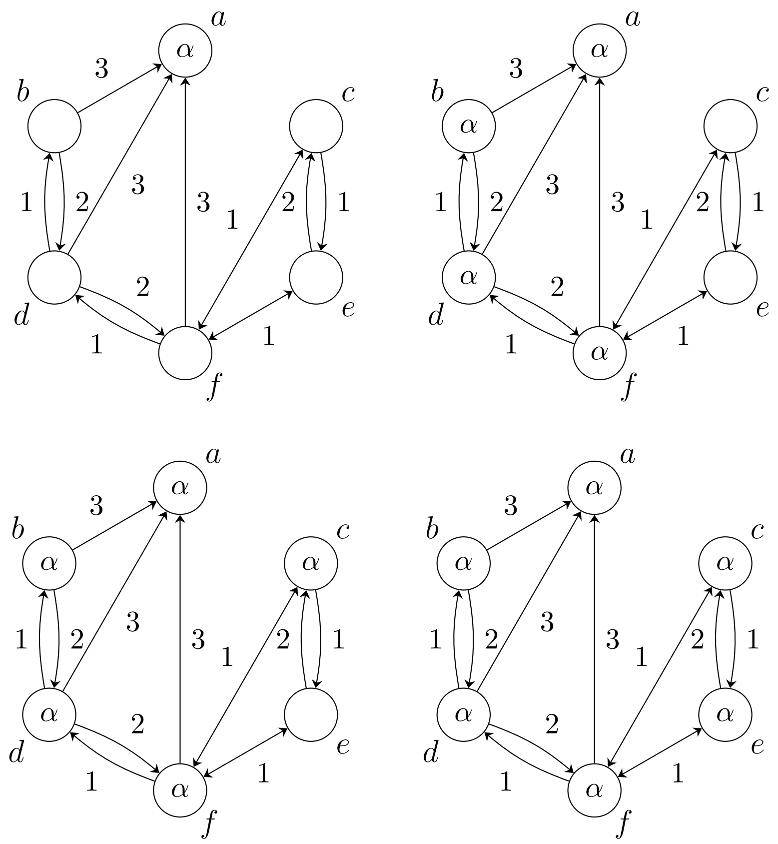


Figure 6.1: This figure presents a social network (with $\theta = 1/2$) with six agents and the evolution of a behavior. Nodes are agents, their contents indicate their behaviors, and the edges represent the relations between them. Each figure, ordered from left to right and top to bottom, represents a distinct position as the network evolves. Agent a represents a trusted broadcaster, spreading behavior α, which eventually dominates the network.

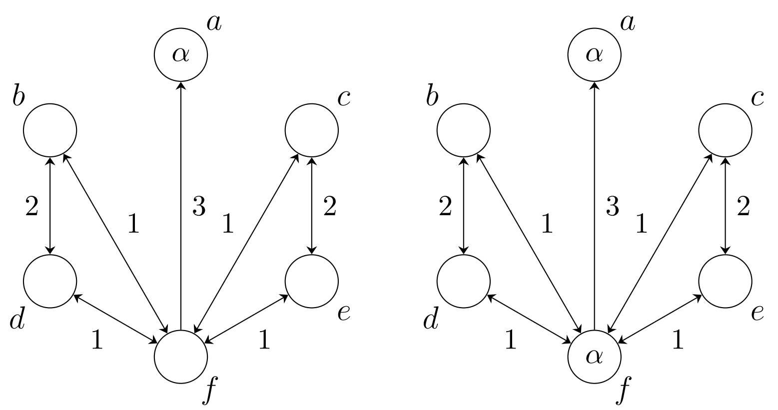


Figure 6.2: Propagation of the behavior is stopped by a cluster of agents ($\theta = 1/2$).

6.2.1 Soundness and completeness

Most of the definitions, lemmas and proofs presented in chapters 4 and 5 are easily adapted to the setting presented in this generalization. For instance, the axiom Red.Ax.$X.\beta$ presented in table 4.1 is essentially the same, though it now refers to any arbitrary behavior $\beta \in \mathsf{B}$.

For soundness, we review the case of the α_a operator as follows: For $\mathcal{M}, i \models X\alpha_a \leftrightarrow \alpha_a \vee \alpha_{N(a)\geq\theta}$, consider $B = b_i$ and a $B' = b_{i+1}$ such that $B \leq B'$. $\mathcal{M}, i \models X\alpha_a$ iff $\mathcal{M}, i+1 \models \alpha_a$ iff $a \in b_{i+1}(\alpha)$ and $B' = B \cup \left\{ b \in \mathcal{A} : \frac{W_a(\mathcal{G})}{W_a(\mathcal{A})} \geq \theta \right\}$ iff $\mathcal{M}, i \models \alpha_a$ or $a \in \left\{ b \in \mathcal{A} : \frac{W_a(\mathcal{G})}{W_a(\mathcal{A})} \geq \theta \right\}$.

It can be shown that the large disjunct in definition 26 is satisfied iff $a \in \left\{ b \in \mathcal{A} : \frac{W_a(\mathcal{G})}{W_a(\mathcal{A})} \geq \theta \right\}$ just as in [2].

Hence $\mathcal{M}, i \models X\alpha_a$ iff $\mathcal{M}, i \models \alpha_a$ or $\mathcal{M}, i \models \alpha_{N(A)\geq\theta}$

Chapter 7

Model checking

Model checking is a technique used for formal verification of the correctness of a system model with respect to a specified property or set of properties. The idea behind model checking is to automatically explore all possible states of the system model and check whether they satisfy the specified properties. This is done by constructing a mathematical model of the system, typically expressed as a transition system or automaton, and then using an algorithmic procedure to verify the properties.

Model checking for verifying the behavior of finite-state systems has been around since the early 1980s, and since then saw many applications to both theoretical and practical problems [56], such as verification of sequential circuit designs, communication protocols.

One of the major problems in model checking is quick explosion of states in a system, a problem that is usually circumvented by what is called "symbolic model checking", that is, model checking that operates on a representation that does not explicitly contain every individual state [46].

Some of the current approaches to achieve symbolic model checking are the use of BDDs and SAT-based approaches [57], and more recently also SMT-based [58]. The first two are employed in NUSMV, and all three in NUXMV. NUXMV is an extension of the NUSMV model checker for fair transition systems, including support for both finite- and infinite-state systems [58].

For the remaining of this section our focus will be in model checking with NUXMV, but it is worth mentioning that other approaches and other relevant tools exist, such as: mCRL2 for modal μ-calculus formulas, a CTL/LTL/CTL* superset; PAT [10], a model checker for LTL properties with a focus on fairness constraints; LTSmin, a toolbox with multiple languages and algorithms, including LTL.

7.1 Model checking complexity

In this section we analyse the computational complexity of the model checking problem.

Definition 27. *The model checking problem consist of, given a formula ϕ and a finite LTL-SN model $\mathcal{M} = (\mathcal{A}, N, \theta, I)$, determining the set $\mathcal{S}(\phi) = \{i \mid b_i \in b \text{ and } \mathcal{M}, i \models \phi\}$*

Next, we present the model checking algorithms for LTL-SN. Algorithm 1 is the entry point, which invokes the remaining algorithms for each of the operators as they are parsed. Let *label*(i) denote the set of subformulas that hold at path position i. We start by initializing $label(i) := \{\beta_a \mid a \in b_i\} \cup \{N_{ab} \mid b \in N(a)\}$ for all path positions i.

Algorithm 1 procedure Check(ϕ)

---

while $|\phi| \geq 1$ **do**
  if $\phi = (\neg\phi_1)$ **then**
    $Check(\phi_1)$; $CheckNOT(\phi_1)$
  else if $\phi = (\phi_1 \wedge \phi_2)$ **then**
    $Check(\phi_1)$; $Check(\phi_2)$; $CheckAND(\phi_1, \phi_2)$
  else if $\phi = X\phi_1$ **then**
    $Check(\phi_1)$; $CheckX(\phi_1)$
  else if $\phi = \phi_1 U \phi_2$ **then**
    $Check(\phi_1)$;$Check(\phi_2)$; $CheckU(\phi_1, \phi_2)$
  end if
end while

Algorithm 2 procedure CheckNOT(ϕ)

---

for all i **do**
  if $\phi \notin label(i)$ **then**
    $label(i) := label(i) \cup \{(\neg\phi)\}$
  end if
end for

Algorithm 3 procedure CheckAND(($\phi_1 \wedge \phi_2$)

---

for all i **do**
  if $\phi_1 \in label(i)$ **and** $\phi_2 \in label(i)$ **then**
    $label(i) := label(i) \cup \{(\phi_1 \wedge \phi_2)\}$
  end if
end for

Algorithm 4 procedure CheckX(ϕ_1)

$T := \{i \mid \phi_1 \in label(i)\}$
while $T \neq \emptyset$ **do**
  choose $i \in T$
  $T := T \setminus \{i\}$
  if $X\phi_1 \notin label(i-1)$ **then**
    $label(i-1) := label(i-1) \cup \{X\phi_1\}$
  end if
end while

Algorithm 5 procedure CheckEU(α_1,α_2)

$T := \{i \mid \phi_2 \in label(i)\}$
for all $i \in T$ **do**
  $label(i) := label(i) \cup \{E(\alpha_1 \mathcal{U} \alpha_2)\}$
end for
while $T \neq \emptyset$ **do**
  choose maximum $i \in T$
  $T := T \backslash \{i\}$
  while $\phi_1 \in label(i-1)$ **do**
    if $E(\phi_1 \mathcal{U} \phi_2) \notin label(i-1)$ **then**
      $label(i-1) := label(i-1) \cup \{E(\phi_1 \mathcal{U} \phi_2)\}$
      $i := i-1$
      $T := T \backslash \{i\}$
    end if
  end while
end while

Theorem 2. *Given a LTL-SN model $\mathcal{M} = (\mathcal{A}, N, \theta, I)$ and formula ϕ. The computational complexity of the model checking problem is $O(\mid \phi \mid \times \mid \mathcal{A} \mid)$, i.e., linear in the size of the formula times the size of the set of agents.*

Proof. By lemma 1, there exists a unique path such that $b_0 = I$. By lemma 3, this path $b_0, ..., b_n$ reaches a fixed-point such that $n < |\mathcal{A}|$.

The algorithm Check(ϕ) is called once for each sub-formula of ϕ which is $O(|\phi|)$ and each time it activates the algorithms CheckNOT, CheckAND, CheckX and CheckU. The algorithm CheckX, in the worse case, has to visit all b_i, which is $O(|\mathcal{A}|)$ The CheckU, starts with the greatest i where ϕ_2 holds, and then it goes labelling states smaller than i, where ϕ_1 holds, with $\phi_1 U \phi_2$. In worse case, it has to search the whole model which is also $O(|\mathcal{A}|)$. The algorithms CheckNOT, CheckAND take constant time. Thus, the complexity of Check(ϕ) is $O(|\mathcal{A}| \times |\phi|)$. In order to o build the set $S(\phi)$ we only need to search $label(i)$, for all $0 \leq i < |\mathcal{A}|$, and check if $\phi \in label(i)$. □

7.2 Model checking LTL-SN

In this section we present an example of a social network built under the definitions of LTL-SN, which we specify in the language of the nuXmv model checker. Notice however that nuXmv is a general LTL/CTL model checker, as such it is not optimized for the model restrictions of LTL-SN and therefore can be somewhat slow for larger models.

It is also important to notice that the Bounded Model Checking (BMC) algorithm that was used looks for counter-examples only up to configurable length k. As such, it cannot prove a property is valid, only find counter-examples of limited length [59].

We have used the "Star Wars Social Network" dataset to build a social network [1]. The dataset parser source-code is available in appendix D and also on GitHub[2].

The parser reads the dataset input and converts it into SMV specification language, following algorithm 6. This specifications defines the automata that generates the states of the social network model. It is then possible to check LTL formulas against it.

In summary, the parser algorithm performs the following steps:

- all the "links" in the model definition are read and used to initialize a matrix with the relationship weights between each of the agents. Non connected agents have a relationship of value 0.
- the initial state is read from the "behavior" property of the "nodes", and defined as a vector using the *init* function.
- The next state is defined by the *next* function, which conditionally attributes the behavior state for each agent, depending on if the proportion of its neighbors that display the behavior is above the adoption threshold. This is calculated by iterating over the previously defined state vector, and summing the relationship weight from the current agent to each of the agents in the vector which currently display the behavior.

The parser is also capable of outputting GraphViz files for visualization of the generated social network. Figures 7.1, 7.2 and 7.3 show one such network model, where nodes marked in red are ones that were defined to have the behavior initially,

[1] `https://www.kaggle.com/ruchi798/star-wars`
[2] `https://github.com/vittau/starwars-dataset-parser`

and yellow nodes denote agents who have just incorporated the behavior of the network.

Algorithm 6 Star Wars dataset parser

t

  for $\{source, target, value\} \in links$ **do**
    $relationships[source][target] \leftarrow value$
    if symmetric **then**
      $relationships[target][source] \leftarrow value$
    end if
  end for
  for $\{index, behavior\} \in nodes$ **do**
    $init(agents[index]) \leftarrow behavior$
    $neighbors\_with\_behavior \leftarrow \sum_{a \in \mathcal{A} | agents[a] = behavior} relationships[index][a]$
    $neighbors \leftarrow \sum_{a \in \mathcal{A}} relationships[index][a]$
    if $\frac{neighbors\_with\_behavior}{neighbors} \geq threshold$ **then**
      $next(agents[index]) \leftarrow behavior$
    end if
  end for
  Output *relationships*, *init* and *next* in SMV syntax

The commands used in NUXMV to input this model are shown in listing 7.1.

Listing 7.1: NUXMV commands used to load and setup the model

```
read_model -i input.smv
flatten_hierarchy
encode_variables
build_boolean_model
bmc_setup
```

An execution trace of the model checker is shown in listing 7.2. The initial state is represented in block "State: 1.1", and in the following states several other agents adhere to the behavior. Agents "DARTH VADER", "MOTTI" and "TARKIN" never adhere because they form a strong "cluster" [2], and agent "GOLD FIVE" is isolated.

Listing 7.2: Checking the formula $GF\beta_{luke}$ against the model in figure 7.1. *luke* is $agents[2]$. Some variables used for calculations were omitted from the trace.

```
check_ltlspec_bmc -p "G ( F agents[2] = behavior)"

-- specification G ( F agents[2] = behavior) is false
-- as demonstrated by the following execution sequence
Trace Description: BMC Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
```

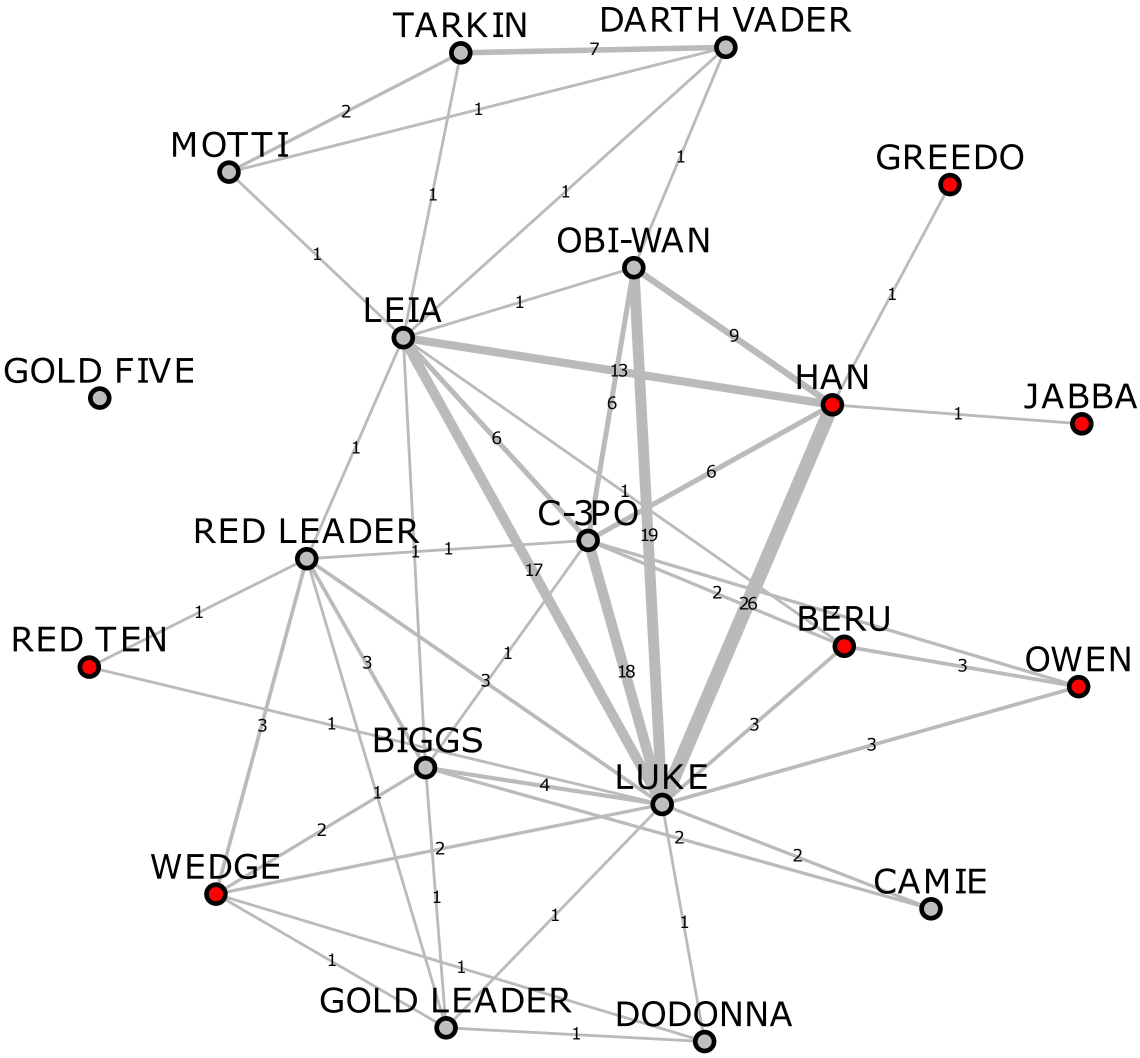


Figure 7.1: Star Wars: Episode IV's social network initial state. Red agents exhibit the behavior being modelled; Numbers and thickness of the edges represent the weight of the relationship (the number of scenes both characters appeared together)

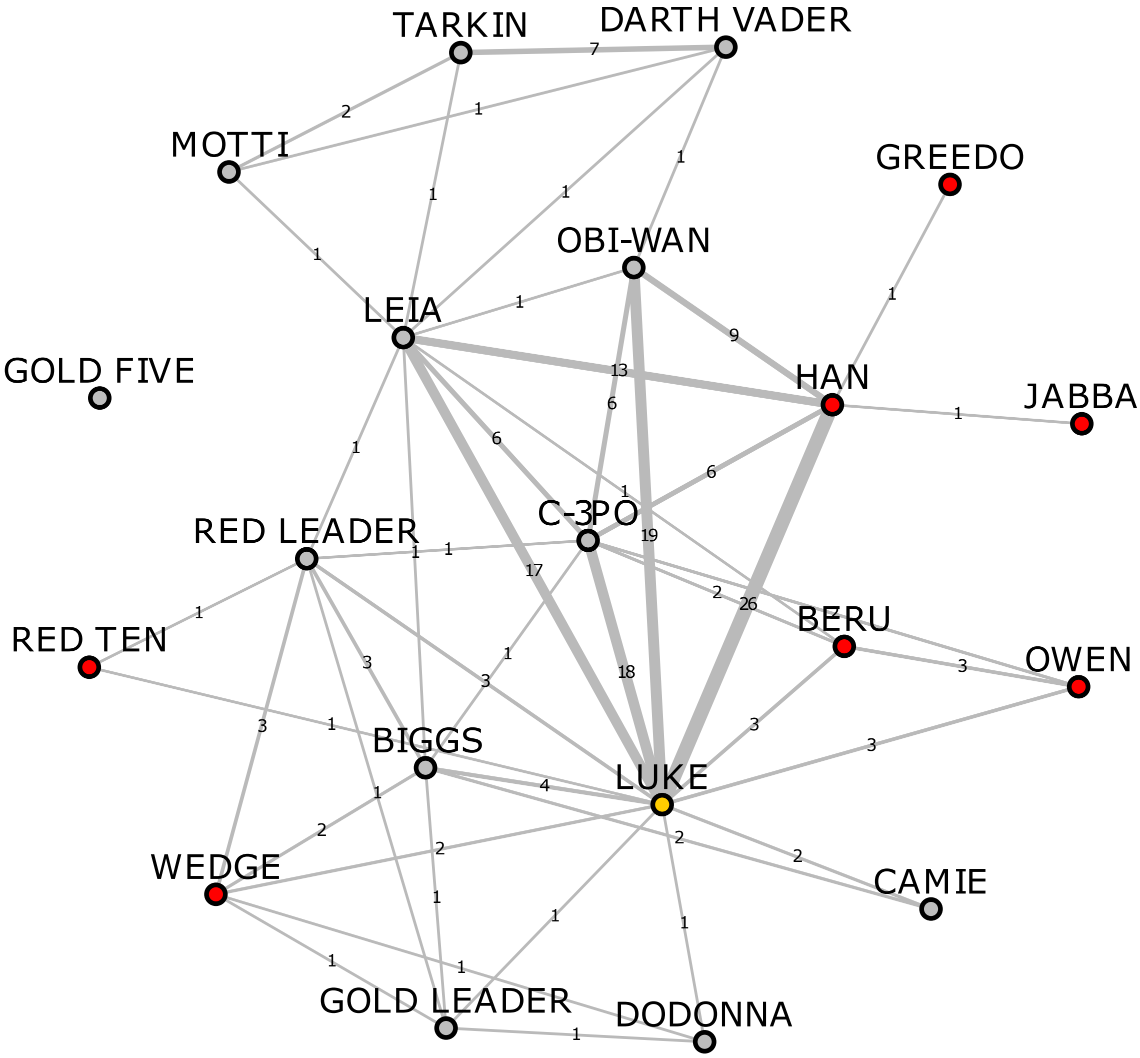


Figure 7.2: On the next state, only Luke incorporates the behavior, indicated by the yellow node

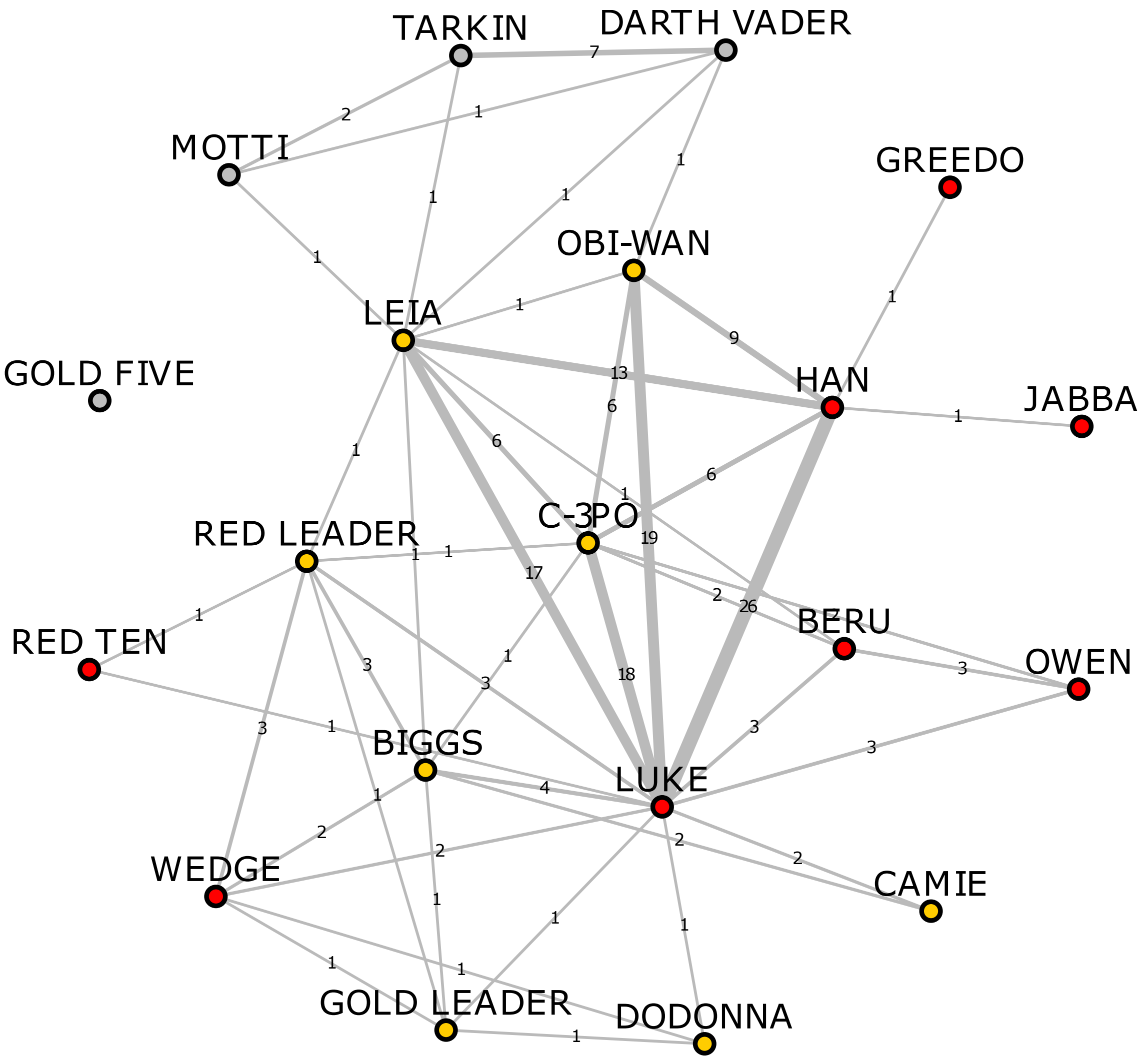


Figure 7.3: After Luke incorporates the behavior, and since he is an influential agent with multiple connections, many other connected agents follow suit

```
  agents[0] = empty
  agents[1] = empty
  agents[2] = empty
  agents[3] = empty
  agents[4] = empty
  agents[5] = empty
  agents[6] = behavior // BERU
  agents[7] = behavior // OWEN
  agents[8] = empty
  agents[9] = empty
  agents[10] = empty
  agents[11] = behavior // HAN
  agents[12] = behavior // GREEDO
  agents[13] = behavior // JABBA
  agents[14] = empty
  agents[15] = empty
  agents[16] = behavior // WEDGE
  agents[17] = empty
  agents[18] = behavior // RED TEN
  agents[19] = empty
  [...]
-> State: 1.2 <-
  agents[1] = behavior // LUKE
  [...]
-- Loop starts here
-> State: 1.3 <-
  agents[0] = behavior // C-3PO
  agents[3] = behavior // CAMIE
  agents[4] = behavior // BIGGS
  agents[5] = behavior // LEIA
  agents[8] = behavior // OBI-WAN
  agents[14] = behavior // DODONNA
  agents[15] = behavior // GOLD LEADER
  agents[17] = behavior // RED LEADER
  [...]
-> State: 1.4 <-
```

Chapter 8

SIR model variation

The previously introduced LTL-SN logic provides a good basis for modelling diffusion phenomena. In this chapter we introduce another type of diffusion model to be studied in this LTL framework, a compartmental model for epidemics denoted "SIR" which we previously mentioned in chapter 2.4.

In [40] a Probabilistic Linear Temporal Logic (PLTL) approach is briefly proposed for model checking SIR models. Here, we expand our LTL formalism for such models, which include again a sound and complete axiomatization.

To work with these we define a different subset of LTL models that will conform to the SIR model behavior, previously introduced in section 2.4. To do so, we must first redefine behavior sets, model definition and the binary relation $\leq$ that defines the evolution of the network at each step.

The language and semantics are the same as in LTL-SN, presented here in definitions 28 and 29 for self-containedness.

Definition 28 (Language)**.** *The language of LTL-SIR is given in Backus-Naur Form (BNF) as follows:*

$$\varphi := \top \mid N_{ab} \mid \alpha_{(a,i)} \mid \alpha_{(a,r)} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid X\varphi \mid \varphi_1 U \varphi_2$$

where $a, b \in \mathcal{A}$, a finite set of agents.

The remaining operators can be defined via abbreviations: $\bot \equiv \neg\top$, $\varphi \vee \phi \equiv \neg(\neg\varphi \wedge \neg\phi)$, $\varphi \rightarrow \phi \equiv \neg(\varphi \wedge \neg\phi)$, $F\varphi \equiv \top U \varphi$ and $G\varphi \equiv \neg F \neg\varphi$. Intuitive readings of the operators are the same as previously introduced in definition 1.

Definition 29 (Satisfaction)**.** *Given a model $\mathcal{M} = (\mathcal{A}, N, \theta, I)$ and its path $b = b_0, b_1, b_2, \ldots, b_i, \ldots$, the notion of satisfaction of formulas in $\mathcal{M}$ at a position i is* *defined as follows:*

$$
\begin{array}{lll}
\mathcal{M}, i \models \alpha_{(a,i)} & \textit{iff} & (a,i) \in b_i \\
\mathcal{M}, i \models \alpha_{(a,r)} & \textit{iff} & (a,r) \in b_i \\
\mathcal{M}, i \models N_{ax} & \textit{iff} & x \in N(a) \\
\mathcal{M}, i \models \neg\varphi & \textit{iff} & \mathcal{M}, i \not\models \varphi \\
\mathcal{M}, i \models \varphi_1 \wedge \varphi_2 & \textit{iff} & \mathcal{M}, i \models \varphi_1 \textit{ and } \mathcal{M}, i \models \varphi_2 \\
\mathcal{M}, i \models X\varphi & \textit{iff} & \mathcal{M}, i+1 \models \varphi \\
\mathcal{M}, i \models \varphi U \psi & \textit{iff} & \textit{there exists a } n \geq 0 \textit{ such that} \\
 & & \mathcal{M}, i+n \models \psi \textit{ and } \mathcal{M}, i+j \models \varphi \textit{ for all } 0 \leq j < n
\end{array}
$$

We redefine behavior sets as sets of tuples indicating the current state of agents. For simplicity, we explicitly mark agents as either "Infected" (i) or "Recovered" (r), and all other agents are assumed "Susceptible".

Definition 30 (SIR Behavior sets)**.** *A behavior set B is a set of tuples indicating the current state of agents. Therefore, $B \subseteq \mathcal{B} = \{(a,x) \mid a \in \mathcal{A}, x \in \{i,r\}\}$, such that $\{(a,i),(a,r)\} \not\subseteq B$ for all a.*

We proceed to adjust the model definition: the adoption threshold θ is removed, since agents are infected by the presence of a single infected neighbour, and I is a subset of $\mathcal{B}$ instead of being an element of it.

Definition 31 (SIR Model)**.** *A model is a tuple $\mathcal{M} = (\mathcal{A}, N, I)$, where*

- *$\mathcal{A}$ and N are defined as in definition 8;*
- *$I \subseteq \mathcal{B}$ is a non-empty initial behavior set.*

Before we proceed to the definition of the $\leq$ relation for SIR models, we define abbreviations for the sets of infected, recovered and exposed agents. Exposed agents are those that have infected neighbours, and are not yet infected or recovered.

Definition 32 (Infected, recovered and exposed agent sets)**.** *Let $B \in \mathcal{B}$ be a behavior set. We define the sets of Infected, Recovered and Exposed agents in B respectively as:*

$$
\begin{aligned}
I_B &= \{a : (a,i) \in B\} \\
R_B &= \{a : (a,r) \in B\} \\
E_B &= \{a : \{N(a) \cap I_B\} \neq \emptyset \textit{ and } a \notin I_B \textit{ and } a \notin R_B\}
\end{aligned}
$$

With these we proceed to define the $\leq$ relation for SIR models.

Definition 33 (SIR relation $\leq$). *Let $\leq$ be the binary relation over $\mathcal{B} \times \mathcal{B}$, where*

$$B \leq B' \text{ iff } B' = \{(a, i) : a \in E_B\} \cup \{(a, r) : a \in I_B \cup R_B\}$$

Intuitively, the first term of the union captures agents that were exposed and become infected, and the second term captures agents that were infected and become recovered.

It is straightforward to see that SIR models without reinfections, and with a fixed set of agents, always reach a fixed point [37].

Lemma 11 (Fixed-point for the SIR model). *Let $b = b_0, b_1, b_2, \ldots, b_n$ be a path. For some $i < |\mathcal{A}|$ a fixed-point $b_i = b_{i+1}$ is reached.*

Proof. This follows directly from the fact that $\mathcal{A}$ is finite and the compartment of each agent can only strictly follow the path $S \to I \to R$. In the slowest diffusion case, a single agent is infected ((a, I)), and for each subsequent position another agent becomes infected and the previous one recovers ((a, R)). Since a recovered agent cannot become infected again, it follows directly that a fixed-point must be reached in some position $i < |\mathcal{A}|$. □

Therefore it is possible to adapt the argument used for the Until expansion satisfaction (lemma 2) for this type of process.

A single infected neighbor is sufficient to infect an agent, and this behavior is captured in the revised adoption threshold abbreviation.

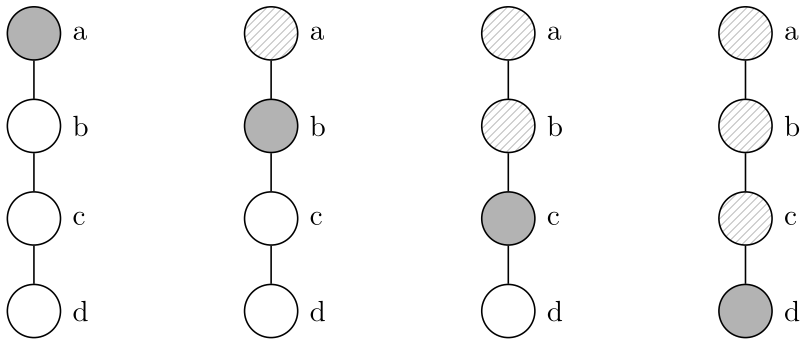
Definition 34 (Revised Adoption). *An abbreviation expressing the adoption threshold for SIR models:*

$$\beta_{N(a)} \coloneqq \bigvee_{c \in \mathcal{N} \subseteq \mathcal{A}} \left(\bigwedge_{b \in \mathcal{N}} N_{ab} \wedge \bigwedge_{b \notin \mathcal{N}} \neg N_{ab} \wedge (c, i) \right)$$

Definition 35 (Revised cost measure).

$$\begin{aligned}
c(\beta_a) &= 1 \\
c(N_{ab}) &= 1 \\
c(\neg\varphi) &= 1 + c(\varphi) \\
c(\varphi_1 \wedge \varphi_2) &= 1 + max(c(\varphi_1), c(\varphi_2)) \\
c(X\beta_a) &= 4 + 2 \cdot |\mathcal{A}|^2 \\
c(X\psi) &= 2 \cdot c(\psi)
\end{aligned}$$

where $X\psi$ matches any formula besides those of form $X\beta_a$.



(a) Position 1 (b) Position 2 (c) Position 3 (d) Position 4

Figure 8.1: In this simple SIR setting, every agent reachable from an infected agent via the neighborhood relationship N will eventually become infected.

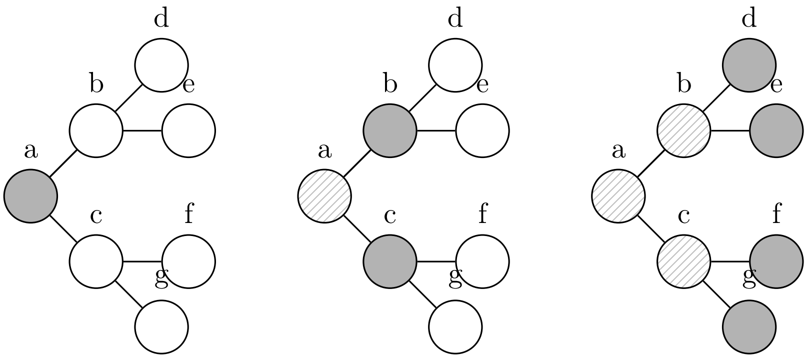
Lemma 12 (Revised cost of the adoption abbreviation)**.** *The cost of the adoption abbreviation* $c(\beta_{N(a)\geq\theta})$ *in a model with agent set* $\mathcal{A}$ *is* $2 \cdot |\mathcal{A}|^2$.

Proof. The outer disjunction ranges over one group, $\mathcal{N} \subseteq \mathcal{A}$, therefore in the worst case there will be as many terms as $|\mathcal{A}|$ for each possible combinations of both. Now analyzing each term, the first conjunction contributes at most $|\mathcal{A}|$ propositions, and the second conjunction another $|\mathcal{A}|$. Therefore each term contributes, in the worst case, a cost of $2 \cdot |\mathcal{A}|$. Therefore, the total cost is $2 \cdot |\mathcal{A}|^2$. □

The soundness and completeness proofs are dependant on the following: correctness of lemma 2; the fact that paths generated by this model are equivalent to conventional LTL paths; the revised adoption abbreviation and cost measure definition, with which the proof of lemma 8 can be adapted; and the ability to express the adoption threshold as a formula in the logic (definition 34). Therefore this logic is also sound and complete used in conjunction with SIR models as presented here.

Lastly, the model checking technique presented in section 7 can be applied almost directly, merely adapting the label initialization to $label(i) := \{\beta_a \mid (a, I) \in b_i\} \cup \{\beta_a \mid (a, R) \in b_i\} \cup \{N_{ab} \mid b \in N(a)\}$ for all path positions i. As such, it remains possible to verify SIR models under the LTL formalism with the same tooling and time complexity previously presented.

As figures 8.1 and 8.2 showcase, these models are very simple and predictable. They don't allow for any variation of the basic reproduction number, that is, the expected number of agents infected by a single infected agent. While we don't tune this number directly, the modelling present in chapter 9 allows us to explore more diverse infection-rate scenarios by leveraging the branching time formalism of CTL.



(a) Position 1 (b) Position 2 (c) Position 3

Figure 8.2: In this simple SIR setting, every agent reachable from an infected agent via the neighborhood relationship N will eventually become infected.

Chapter 9

Branching SIR models using CTL

Compartmental models for epidemics have figured in the literature since 1916 [34], first introduced as sets of differential equations for what is today referred to as the SIS (Susceptible-Infectious-Susceptible) model. These compartmental models are usually name after their compartment classes, indicating the evolution of an individual over time in such models. Later [35] proposed the SIR (Susceptible-Infectious-Recovered) model, where infected agents are infectious for one round, and then become recovered and cannot be infected nor spread the disease any further. The presence of a single connection in the network is sufficient to spread a disease from one agent to another. In this section we will focus on SIR models in particular.

There has been some work on SIR models in the field of logics. In [37], these models are studied in a sound and complete hybrid logic, and a tableau system is also provided. While [38] and a section of [39] explore fuzzy logic approaches to SIR epidemic models.

In [60] for the first time the logic of branching-time CTL was presented, with the aim of dealing with the set of every possible execution tree generated by a given program. This logic was specially designed to take care of the consequences of non-determinism just like the one generated by programs that interact asynchronously.

In [15] the usual CTL specification was presented, providing a decision procedure, and for this reason the logic presented here is closely related to the previously established logic.

In this section we explore SIR models, which are closely related to social networks behavior spreading models, using CTL. CTL allows us to introduce branching time events as part of the model evolution over time. In our particular case, whenever an agent is exposed to another infected agent, time branches: in one line the agent is infected, and in the other the agent evades infection. This allows us to see multiple "what-if" scenarios along the evolution of the network, and we can also see and analyze social constructs which act as "barriers" for the spread of diseases, and conversely, social components which facilitate superspreading events where the likelihood of contagion is extremely high.

We redefine behavior sets as sets of tuples indicating the current state of agents. For simplicity, we explicitly mark agents as either “Infected” (i) or “Recovered” (r), and all other agents are assumed “Susceptible”.

Definition 36 (SIR Behavior sets). *Let $\mathcal{B} = 2^{\mathcal{A}\times\{i,r\}}$ be the set of all possible behaviors of agents. A behavior set $B \in \mathcal{B}$ is a set of tuples, each of which indicates the state of an agent in the network. Therefore, $B = \{(a, x) \mid a \in \mathcal{A}$ and $x \in \{i, r\}\}$, such that $\{(a, i), (a, r)\} \not\subset B$ for all a.*

We proceed to adjust the model definition: the adoption threshold θ is removed, since agents are infected by the presence of a single infected neighbour, and I is a subset of $\mathcal{B}$ instead of being an element of it.

Definition 37 (SIR Model). *A model is a tuple $\mathcal{M} = (\mathcal{A}, N, I)$, where*

- *$\mathcal{A}$ and N are defined as in definition 8;*
- *$I \subseteq \mathcal{B}$ is a non-empty initial behavior set.*

Before we proceed to the definition of the $\leq$ relation for SIR models, we define abbreviations for the sets of infected, recovered and exposed agents. Exposed agents are those that have infected neighbours, and are not yet infected or recovered.

Definition 38 (Infected, recovered and exposed agent sets). *Let $B \in \mathcal{B}$ be a behavior set. We define the sets of Infected, Recovered and Exposed agents in B respectively as:*

$$\begin{aligned}
I_B &= \{a : (a, i) \in B\}\\
R_B &= \{a : (a, r) \in B\}\\
E_B &= \{a : \{N(a) \cap I_B\} \neq \emptyset \text{ and } a \notin I_B \text{ and } a \notin R_B\}
\end{aligned}$$

With these we proceed to define the $\leq$ relation for SIR models.

Definition 39 (SIR relation $\leq$). *Let $\leq$ be the binary relation over $\mathcal{B} \times \mathcal{B}$, $B'_1 = \{(a, i) : a \in E_B\}$, $B'_2 = \{(a, r) : a \in I_B\}$ and $B'_3 = \{(a, r) : a \in R_B\}$:*

$$B \leq B' \text{ iff } B'_2 \subseteq B' \text{ and } (B' \setminus B'_2) \subseteq B'_1 \text{ and } B'_3 \subseteq B'$$

Intuitively, the B'_1 captures agents that were exposed and become infected, B'_2 captures agents that were infected and become recovered, and B'_3 captures agents that were already recovered and stay recovered. Notice this relation is not linear, and allows branching: every agent infected in B must recover in B', but not every exposed agent in B must become infected in B'.

It is straightforward to see that SIR models without reinfections, and with a fixed set of agents, always reach a fixed point [37].

Lemma 13 (Fixed-point for the SIR model)**.** *Let $b = b_0, b_1, b_2, \dots, b_n$ be a path. For some $i < |\mathcal{A}|$ a fixed-point $b_i = b_{i+1}$ is reached.*

Proof. This follows directly from the fact that $\mathcal{A}$ is finite and the compartment of each agent can only strictly follow the path $S \to I \to R$. In the slowest diffusion case, a single agent is infected ((a, I)), and for each subsequent position another agent becomes infected and the previous one recovers ((a, R)). Since a recovered agent cannot become infected again, it follows directly that a fixed-point must be reached in some position $i < |\mathcal{A}|$. □

Therefore it is possible to adapt the argument used for the Until expansion satisfaction (lemma 2) for this type of process.

A single infected neighbor is sufficient to infect an agent, and this behavior is captured in the revised adoption threshold abbreviation.

Definition 40 (Infection)**.** *An abbreviation expressing the infection for SIR models:*

$$\alpha_{N(a)} := \bigvee_{c \in \mathcal{N} \subseteq \mathcal{A}} \left(\bigwedge_{b \in \mathcal{N}} N_{ab} \wedge \bigwedge_{b \notin \mathcal{N}} \neg N_{ab} \wedge \alpha_{(c,i)} \right)$$

Definition 41 (CTL Language)**.** *The language of SIR-CTL is given by the following BNF:*

$$\varphi := \top \mid N_{ab} \mid \alpha_{(a,i)} \mid \alpha_{(a,r)} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid EX\varphi \mid EG\varphi \mid E[\varphi_1 U \varphi_2]$$

where $a, b \in \mathcal{A}$, $\alpha \in \mathsf{B}$.

We use the standard abbreviations [16]: $\bot \equiv \neg\top$, $\varphi \wedge \phi \equiv \neg(\neg\varphi \vee \neg\phi)$, $\varphi \to \phi \equiv \neg(\varphi \wedge \neg\phi)$, $AX\varphi \equiv \neg EX\neg\varphi$, $EF\varphi \equiv E[\top U\varphi]$,$AG \equiv \neg EF\neg\varphi$, $AF \equiv \neg EG\neg\varphi$, $A(\alpha\mathcal{U}\beta) \equiv (\neg E((\neg\beta)\ \mathcal{U}\ (\neg\alpha \wedge \neg\beta))) \wedge (\neg[EG](\neg\beta))$. Intuitive readings of the operators are the same as previously introduced in definition 1.

Definition 42 (CTL Satisfaction)**.** *Given a model $\mathcal{M} = (\mathcal{A}, \mathsf{B}, N, I)$ and a set of paths $b(s)$ such that $b_0 = s$. The notion of satisfaction of formulas in $\mathcal{M}$ at a* *position s is defined as follows:*

$$
\begin{array}{lll}
\mathcal{M}, s \models \alpha_{(a,i)} & \textit{iff} & (a,i) \in b_0 \\
\mathcal{M}, s \models \alpha_{(a,r)} & \textit{iff} & (a,r) \in b_0 \\
\mathcal{M}, s \models N_{ab} & \textit{iff} & N(a,b) > 0 \\
\mathcal{M}, s \models \neg\varphi & \textit{iff} & \mathcal{M}, s \not\models \varphi \\
\mathcal{M}, s \models \varphi_1 \vee \varphi_2 & \textit{iff} & \mathcal{M}, s \models \varphi_1 \textit{ or } \mathcal{M}, s \models \varphi_2 \\
\mathcal{M}, s \models EX\varphi & \textit{iff} & \textit{there exists a path } b \in b(i) \textit{ such that } \mathcal{M}, b_1 \models \varphi \\
\mathcal{M}, s \models EG\varphi & \textit{iff} & \textit{for all paths } b \in b(i) \textit{ it is the case that } \mathcal{M}, b_0 \models \varphi \\
\mathcal{M}, s \models E[\varphi U \psi] & \textit{iff} & \textit{there exists a path } b \in b(i) \textit{ and an } n \geq 0 \textit{ such that} \\
 & & \mathcal{M}, b_n \models \psi \textit{ and } \mathcal{M}, b_j \models \varphi \textit{ for all } 0 \leq j < n
\end{array}
$$

All CTL operators are defined in a standard way.

Definition 43 (CTL Validity). $\models \varphi$ *iff* $\mathcal{M}, s \models \varphi$ *for all models* $\mathcal{M}$ *and positions* s.

Definition 44 (CTL Until expansion). *Let* $E[\varphi U \psi]$ *be a formula. We define its Until expansion steps as*

$$
\begin{aligned}
u^0(E[\varphi U \psi]) &\coloneqq \psi \\
u^i(E[\varphi U \psi]) &\coloneqq (\varphi \wedge EXu^{i-1})
\end{aligned}
$$

Then, its Until expansion $EXP_U(\varphi U \psi)$ *in a model with agent set* $\mathcal{A}$ *is defined as*

$$EXP_U(E[\varphi U \psi]) \coloneqq \bigvee_{0 \leq i \leq |\mathcal{A}|} u^i(E[\varphi U \psi])$$

Definition 45 (CTL EG expansion). *Let* $EG\varphi$ *be a formula. We define its EG expansion steps as*

$$
\begin{aligned}
g^0(EG\varphi) &\coloneqq \varphi \\
g^i(EG\varphi) &\coloneqq (\varphi \wedge EXg^{i-1})
\end{aligned}
$$

Then, its EG expansion $EXP_{EG}(\varphi)$ *in a model with agent set* $\mathcal{A}$ *is defined as*

$$EXP_{EG}(EG\varphi) \coloneqq \bigvee_{0 \leq i \leq |\mathcal{A}|} g^i(EG\varphi)$$

Proposition 3 (Replacement of equivalents). *If* $\vdash \phi_1 \leftrightarrow \phi_2$*, then* $\vdash [\phi_1/\psi]\chi \leftrightarrow [\phi_2/\psi]\chi$.

The proof of proposition 3 is standard and uses the fact that the semantics is compositional, meaning that the truth value of a formula is determined by the truth

Network axioms	
$\neg N_{aa}$	Irreflexivity
$N_{ab} \leftrightarrow N_{ba}$	Symmetry
$\bigvee_{b \in \mathcal{A}} N_{ab}$	Seriality
Reduction axioms	
$EXN_{ab} \leftrightarrow N_{ab}$	Red.Ax.$EX.N$
$EX\alpha_{(a,i)} \leftrightarrow \alpha_{N(a)}$	Red.Ax.$EX.\alpha.i$
$EX\alpha_{(a,r)} \leftrightarrow \alpha_{(a,i)} \vee \alpha_{(a,r)}$	Red.Ax.$EX.\alpha.r$
$EG\varphi \leftrightarrow EXP_{EG}(EG\varphi)$	Red.Ax.EG
$E[\varphi U \psi] \leftrightarrow EXP_U(E[\varphi U \psi])$	Red.Ax.EU
CTL axioms	
All classical propositional tautologies	
$AG\varphi \leftrightarrow \neg EF \neg\varphi$	
$EG\varphi \leftrightarrow \neg AF \neg\varphi$	
$EX(\varphi \vee \psi) \leftrightarrow EX\varphi \vee EX\psi$	
$AX\varphi \leftrightarrow \neg EX \neg\varphi$	

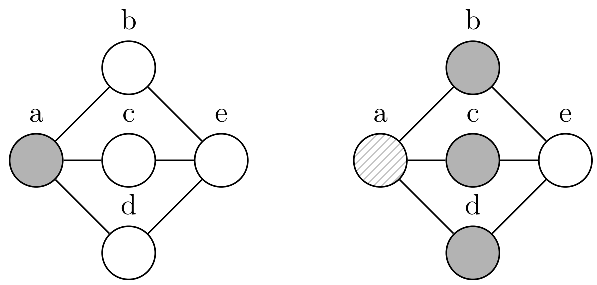
Table 9.1: SIR-CTL axioms.

values of its subformulas. We can then use mathematical induction to show that for any equivalent subformulas ϕ and ψ in φ, substituting ϕ for ψ (or vice versa) in φ does not change the set of traces that satisfy φ.

9.1 Examples

These are some examples of interesting situations and phenomena that can be captured with this framework. Notice that each of these figures only show one of many possible paths: at each position, there are as many branching paths as there are combinations of non-infected/non-recovered agents being exposed to infected agents. Filled nodes indicate infected agents, and nodes with dashed lines indicate a recovered agent.

Figure 9.1 portraits a situation where a single agent is exposed to multiple infected agents, and yet this does not increase the likelihood of this agent becoming infected. In step 9.1a agent a is infected, while agents b, c and d are exposed. There will be 8 distinct paths after this position, for the combinations of each of these agents becoming infected or not. Let's consider the path such that all three of them become infected, which leads us to step 9.1b. In this step, agent e is exposed to 3 infected agents simultaneously. A characteristic of this simple contact model, is that the chances of agent e becoming infected are exactly the same as if it were exposed to a single infected agent. This differs from a threshold model, where infection is directly related to the amount of exposure and only happens if this exposure exceeds



(a) Position 1 (b) Position 2

Figure 9.1: Simultaneous exposure to multiple infected agents does not increase the likelihood of an agent to be infected in this modelling. After agents b, c and d are (possibly) infected, they will expose agent e to infection simultaneously. Agent e will either be infected or resist, just as if it were exposed to a single agent. An inherent limitation of a simple contact model, as opposed to a threshold model.

a given threshold.

In contrast, figure 9.2 illustrates a more dangerous sequence of events for agent e. In this case, it is exposed to infected agents at multiple different positions, and in each position it can become infected (if it hasn't already become infected and/or recovered before). This figure shows only one of the possible paths, in which agent e is exposed to agent c but does not get infected. It is then exposed to agent d on the next position.

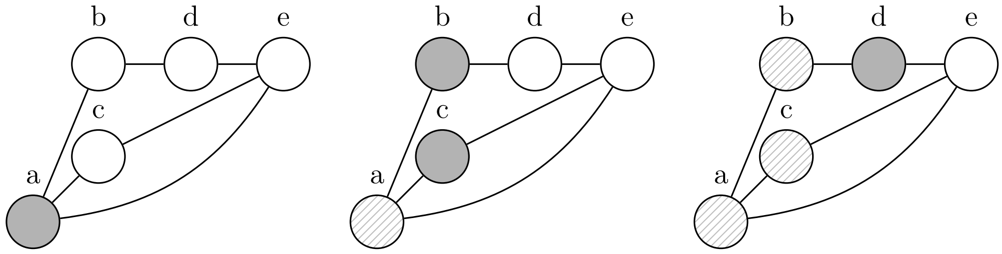
Next, we look at characteristics of infection events themselves. For instance, figure 9.3 depicts a possible superspreading event, that is, when a single agent (agent a in this case) infects multiple other agents simultaneously. There are $2^5 = 32$ branching paths from this position, and in all but one there is at least one infected agent as a result.

Whereas figure 9.4 depicts a much harder barrier for the infection to transpose, since only a single agent gets exposed to infection at every position.

Some interesting questions arise: how many steps must pass until a given agent can become infected? How likely is it for an agent to become infected?

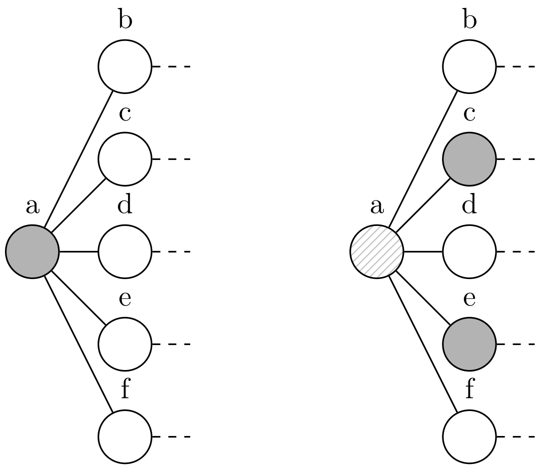
9.2 Expansion equivalences

Before we can show equivalence of the entire EG and Until expansions, we demonstrate that its first n steps are equivalent to the operator if and only if the right-hand side holds before the next n positions (or vacuously never at all). After that, we also determine that the longest distance from the current position at which this can happen has to be less than $|\mathcal{A}|$ positions ahead. With both of these, we are able to prove equivalence of the complete EG and Until expansions.



(a) Position 1 (b) Position 2 (c) Position 3

Figure 9.2: A situation where an agent, agent e in this case, is exposed to infected agents multiple times. Therefore it has a higher chance of being infected than in figure 9.1. Dashed lines indicate a recovered agent. This is only one of the possible paths, in which agent e is exposed to agent c but does not get infected. It is then exposed to agent d on the next position.



(a) Position 1 (b) Position 2

Figure 9.3: An easy barrier for the infection to transpose. Agent a will likely become a super-spreader, as in only 1 of 32 paths no other agent is infected. In this path, agents c and e became infected, while the others resisted.

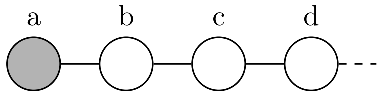


Figure 9.4: A hard barrier for the infection to transpose. For every agent in series the likelihood of infection spreading halves.

Lemma 14 (CTL Until expansion step satisfaction). *$\mathcal{M}, s \models u^n(E[\varphi U \psi])$ if and only if there exists a path $b \in b(s)$ such that $\mathcal{M}, b_n \models \psi$ and $\mathcal{M}, b_j \models \varphi$ for all $0 \leq j < n$.*

Proof. We will abbreviate $u^n(E[\varphi U \psi])$ as u^n for this proof. The proof goes by induction on n.

Base case ($n = 0$):

It is trivial to see that $\mathcal{M}, s \models u^0$ if and only if $\mathcal{M}, s \models \psi$, and there is no j such that $0 \leq j < n$.

Induction hypothesis ($n = k$):

$\mathcal{M}, s \models u^k$ if and only if there exists a path $b \in b(s)$ and an $k \geq 0$ such that $\mathcal{M}, b_k \models \psi$ and $\mathcal{M}, b_j \models \varphi$ for all $0 \leq j < k$.

Induction step ($n = k + 1$):

Assume $\mathcal{M}, s \models u^{k+1}$.

By definition we have that $\mathcal{M}, s \models \varphi \wedge EXu^k$, if and only if $\mathcal{M}, s \models \varphi$ and $\mathcal{M}, s \models EXu^k$. $\mathcal{M}, s \models EXu^k$ if and only if there exists a path $b \in b(s)$ and $\mathcal{M}, b_1 \models u^k$.

Using the I.H., $\mathcal{M}, b_1 \models u^k$ if and only if there exists a path $b' \in b(b_1)$ and an $k \geq 0$ such that $\mathcal{M}, b'_k \models \psi$ and $\mathcal{M}, b'_j \models \varphi$ for all $0 \leq j < k$.

By definition of paths, $b'_k = b_{k+1}$ and $b'_j = b_{j+1}$.

Replacing the first, we get $\mathcal{M}, b_{k+1} \models \psi$. Replacing the second, we get $\mathcal{M}, b_{j+1} \models \varphi$ for all $0 \leq j < k$, which can be written as $\mathcal{M}, b_j \models \varphi$ for all $0 \leq j < k + 1$.

Therefore we are able to conclude that $\mathcal{M}, b_{(k+1)} \models \psi$ and $\mathcal{M}, b_j \models \varphi$ for all $0 \leq j < k + 1$, and that finishes our proof. □

Lemma 15 (CTL Until expansion soundness).

$$\models E[\varphi U \psi] \leftrightarrow EXP_U(E[\varphi U \psi])$$

Proof. By definition, $EXP_U(E[\varphi U \psi]) = u^0(E[\varphi U \psi]) \vee u^1(E[\varphi U \psi]) \vee \ldots \vee u^{|\mathcal{A}|}(E[\varphi U \psi])$. For this expansion to hold, a right-hand side term u^n must hold, and by lemma 18 it is such that $n \leq |\mathcal{A}| + 1$.

Suppose for some model and some position s we have $\mathcal{M}, s \models u^n(E[\varphi U \psi])$. By lemma 14, this is if and only if there exists a path $b \in b(s)$ such that $\mathcal{M}, b_n \models \psi$ and $\mathcal{M}, b_j \models \varphi$ for all $0 \leq j < n$. But this is if and only if $\mathcal{M}, s \models E[\varphi U \psi]$ by definition 42.

Therefore $\models E[\varphi U \psi]$ if and only if $\models EXP_U(E[\varphi U \psi])$.

□

Lemma 16 (CTL EG expansion step satisfaction). *$\mathcal{M}, s \models g^n([EG]\varphi)$ if and only if there exists a path $b \in b(s)$ and for all j, $n \geq j \geq 0$, $\mathcal{M}, b_j \models \varphi$.*

Proof. We will abbreviate $g^n(EG)\varphi$ as g^n for this proof. The proof goes by induction on n.

Base case ($n = 0$):

It is trivial to see that $\mathcal{M}, s \models g^0$ if and only if $\mathcal{M}, s \models \varphi$.

Induction hypothesis ($n = k$):

$\mathcal{M}, s \models g^k$ if and only if there exists a path $b \in b(s)$ and for all $j,\ k \geq j \geq 0,\ \ \mathcal{M}, b_j \models \varphi$.

Induction step ($n = k + 1$):

Assume $\mathcal{M}, s \models g^{k+1}$.

By definition we have that $\mathcal{M}, s \models \varphi \wedge EXg^k$, if and only if $\mathcal{M}, s \models \varphi$ and $\mathcal{M}, s \models EXg^k$. $\mathcal{M}, s \models EXg^k$ if and only if there exists a path $b \in b(s)$ and $\mathcal{M}, b_1 \models g^k$.

By the induction hypothesis, we have $\mathcal{M}, b_1 \models g^k$ if and only if $\mathcal{M}, b_{1+j} \models \varphi$ for all $0 \leq j < k$.

$\mathcal{M}, b_{1+j} \models \varphi$ for all $0 \leq j \leq k$ and $\mathcal{M}, s \models \varphi$ can be rewritten as $\mathcal{M}, b_j \models \varphi$ for all $0 \leq j < k + 1$.

Therefore we are able to conclude that $\mathcal{M}, b_j \models \varphi$ for all $0 \leq j < k+1$, and that finishes our proof. □

Lemma 17 (CTL EG expansion soundness)**.**

$$\models EG\varphi \leftrightarrow EXP_{EG}(EG\varphi)$$

Proof. By definition, $EXP_{EG}(EG\varphi) = g^0 \vee g^1 \vee \ldots \vee g^{|\mathcal{A}|}$.

Suppose $\mathcal{M}, s \models [EG]\varphi$. But this is iff there exists a path $b \in b(s)$ such that that $\mathcal{M}, b_j \models \varphi$, for all $j \geq 0$. By lemma 18, we know that path b reaches a fix point for some $k \leq |\mathcal{A}|$, $b_k = b_{k+1}$. By lemma 16, we have that $\mathcal{M}, s \models g^k$. Therefore $\mathcal{M}, s \models [EG]\varphi$ if and only if $\models EXP_{EG}(EG\varphi)$. □

Lemma 18 (CTL Fixed-point)**.** *Let $b = b_0, b_1, b_2, \ldots, b_n$ be a path. For some $i < |\mathcal{A}|$ a fixed-point $b_i = b_{i+1}$ is reached.*

Proof. This follows directly from the fact that $\mathcal{A}$ is finite and $b_i \subseteq b_{i+1}$ for all i by definition. The slowest possible diffusion scenario is where $|b_0| = 1$ and only one agent becomes infected per position on the path until the fixed-point is reached. It follows directly that the fixed-point will have to be reached for some $i \leq |\mathcal{A}| + 1$. □

9.3 Soundness and completeness

Since SIR-CTL is a specialization of standard CTL for a class of social networks models, all standard CTL axioms are sound in SIR-CTL.

Lemma 19 (Soundness)**.** *We will show all the axiom schemata presented are valid, per definition 43. For this, let $\mathcal{M}$ be an arbitrary SIR-CTL model and $B \in \mathcal{B}$ a behavior set.*

Proof. Consider each case:

- All network axioms follow directly from definition 37;
- $\mathcal{M}, s \models EXN_{ab} \leftrightarrow N_{ab}$ follows directly from the fact that $\leq$ never alters the network structure;
- $\mathcal{M}, s \models E[\varphi U \psi] \leftrightarrow EXP_U(E[\varphi U \psi])$ follows directly from lemma 15;
- $\mathcal{M}, s \models EG\varphi \leftrightarrow EXP_{EG}(EG\varphi)$ follows directly from lemma 17;
- $\mathcal{M}, s \models EX\alpha_{(a,i)} \leftrightarrow \alpha_{N(a)}$: By definition, $\mathcal{M}, s \models EX\alpha_{(a,i)}$ if and only if there exists a path $b \in b(s)$ and $\mathcal{M}, b_1 \models \alpha_{(a,i)}$. By definition 39, $\mathcal{M}, b_1 \models \alpha_{(a,i)}$ is the case if and only if $\mathcal{M}, s \models \alpha_{(c,i)}$ for some $c \in N(a)$. Since $c \in N(a)$, the first two terms in definition 40 ensure $c \in \mathcal{N}$, and since $\mathcal{M}, s \models \alpha_{(c,i)}$, it can be concluded that $\mathcal{M}, s \models \alpha_{N(a)}$.
- $\mathcal{M}, s \models EX\alpha_{(a,r)} \leftrightarrow \alpha_{(a,i)} \vee \alpha_{(a,r)}$ follows directly from the term $REC \subseteq B'$ in definition 39.

□

For the completeness proof, our procedure is a translation of formulas into equivalent formulas with no EG and Until operator occurrences, and then a subsequent replacement of the occurrences of EX in this formula. The resulting formula contains only standard propositional operators, which yields the proof directly.

Definition 46 (EG-Until Translation)**.**

$$\begin{aligned}
t(N_{ab}) &= N_{ab} \\
t(\alpha_a) &= \alpha_a \\
t(\varphi \wedge \psi) &= t(\varphi) \wedge t(\psi) \\
t(\neg\varphi) &= \neg t(\varphi) \\
t(EX\varphi) &= EXt(\varphi) \\
t(EG\varphi) &= t(EXP_{EG}(EG\varphi)) \\
t(E[\varphi U \psi]) &= t(EXP_U(E[\varphi U \psi]))
\end{aligned}$$

The next lemma assures that every formula is equivalent to another formula without any occurrence of the EG and EU operators.

Lemma 20 (EG-Until translation equivalence)**.**

$$\vdash \varphi \leftrightarrow t(\varphi)$$

Proof. Proof by induction on the length $|\varphi| = n$ of the formula φ.
Base case ($n = 1$): $\varphi = N_{ab}$ or $\varphi = \alpha_a$

This is straightforward from definition 16.

Induction hypothesis: $\vdash \varphi \leftrightarrow t(\varphi)$ for $|\varphi| \leq n$.
Induction step:

We have five cases:

1. $\varphi = \neg\psi$: we have that $t(\neg\psi) = \neg t(\psi)$.

   By the I. H., $\vdash \psi \leftrightarrow t(\psi)$ and
   $\vdash \neg\psi \leftrightarrow \neg t(\psi)$. Thus $\vdash \varphi \leftrightarrow t(\neg\psi)$.

2. $\varphi = \varphi_1 \vee \varphi_2$: we have that $t(\varphi_1 \vee \varphi_2) = t(\varphi_1) \vee t(\varphi_2)$.

   By the I. H., $\vdash \varphi_1 \leftrightarrow t(\varphi_1)$ and $\vdash \varphi_2 \leftrightarrow t(\varphi_2)$.
   Thus $\vdash \varphi_1 \vee \varphi_2 \leftrightarrow t(\varphi_1) \vee t(\varphi_2)$ and
   $\vdash \varphi_1 \vee \varphi_2 \leftrightarrow t(\varphi_1 \vee \varphi_2)$.

3. $\varphi = EX\psi$: we have that $t(EX\psi) = EXt(\psi)$.

   By the I. H., $\vdash \psi \leftrightarrow t(\psi)$.
   Using using Replacement of equivalents rule (proposition 3),
   $\vdash EX\psi \leftrightarrow t(EX\psi)$. Thus $\vdash \varphi \leftrightarrow t(EX\psi)$.

4. $\varphi = E[\varphi_1 U \varphi_2]$: we have that $t(E[\varphi_1 U \varphi_2]) = t(EXP_U(E[\varphi_1 U \varphi_2]))$.

   Claim 1: $\vdash t(EXP_U(E[\varphi_1 U \varphi_2])) \leftrightarrow EXP_U(E[t(\varphi_1) U t(\varphi_2)])$

   Proof: $EXP_U(E[t(\varphi_1) U t(\varphi_2)]) = \bigvee_{0 \leq i \leq |\mathcal{A}|+1} u^i(E[t(\varphi_1) U t_(\varphi_2)])$
   $= u^0 \vee u^1 \vee \ldots = t(\varphi_2) \vee (t(\varphi_1) \wedge EXt(\varphi_2)) \vee \ldots$

   Using 1., 2. and 3., we can bring the translation operator outside yielding
   $\vdash t(EXP_U(E[\varphi_1 U \varphi_2])) \leftrightarrow EXP_U(E[t(\varphi_1) U t(\varphi_2)])$.

   Returning to the proof of 4., we have $t(E[\varphi_1 U \varphi_2]) = t(EXP_U(E[\varphi_1 U \varphi_2]))$
   $= EXP_U(E[t(\varphi_1) U t(\varphi_2)])$ by Claim 1. Using axiom Red.Ax.EU, we have
   $\vdash EXP_U(E[t(\varphi_1) U t(\varphi_2)]) \leftrightarrow (E[t(\varphi_1) U t(\varphi_2)])$

   By the I. H., $\vdash \varphi_1 \leftrightarrow t(\varphi_1)$ and $\vdash \varphi_2 \leftrightarrow t(\varphi_2)$. And so
   $\vdash EXP_U(E[t(\varphi_1) U t(\varphi_2)]) \leftrightarrow (E[\varphi_1 U \varphi_2])$.

   Using Claim 1, we obtain $\vdash t(EXP_U(E[\varphi_1 U \varphi_2])) \leftrightarrow (E[\varphi_1 U \varphi_2])$. Thus,
   $\vdash t(E[\varphi_1 U \varphi_2])\varphi \leftrightarrow (E[\varphi_1 U \varphi_2])$.

5. $\varphi = [EG]\varphi_1$: we have that $t([EG]\varphi_1) = t(EXP_{EG}([EG]\varphi))$.

   Claim 2: $\vdash t(EXP_{EG}([EG]\varphi)) \leftrightarrow EXP_{EG}([EG]t(\varphi))$

   Proof: $EXP_{EG}([EG]t(\varphi)] = \bigvee_{0 \leq i \leq |\mathcal{A}|+1} u^i([EG]t(\varphi))$

   $= u^0 \vee u^1 \vee \ldots = t(\varphi) \vee (t(\varphi) \wedge Xt(\varphi)) \vee \ldots$

   Using 1., 2. and 3., we can bring the translation operator outside yielding $\vdash t(EXP_{EG}([EG]\varphi)) \leftrightarrow EXP_{EG}([EG]t(\varphi))$.

   Returning to the proof of 5., we have $t([EG]\varphi_1) = t(EXP_{EG}(EG\varphi_1))$ $= EXP_{EG}([EG]t(\varphi_1))$ by Claim 2. Using axiom Red.Ax.EG, we have $\vdash$ $EXP_{EG}([EG]t(\varphi_1)) \leftrightarrow [EG]t(\varphi_1)$

   By the I. H., $\vdash \varphi_1 \leftrightarrow t(\varphi_1)$. And so
   $\vdash EXP_{EG}([EG]t(\varphi_1)) \leftrightarrow [EG]\varphi_1$.

   Using Claim 2,we obtain $\vdash t(EXP_{EG}([EG]\varphi_1)) \leftrightarrow [EG]\varphi_1$. Thus,
   $\vdash t(EXP_{EG}([EG]\varphi_1)) \leftrightarrow [EG]\varphi_1$.

□

Next we present the replacement of EX occurrences for a formula with no EG and Until occurrences. The intuition behind this replacement stems from the fact that relation $\leq$ directly binds the state of the following position, in which the content of EX is evaluated. Therefore we are able to represent a given formula $EX\varphi$ directly from the state of the current position.

Definition 47 (Replacement of $EX\varphi$)**.** *For a formula of the form $\varphi = EX\phi$ where ϕ does not contain any CTL operator (EX, EG and EU), then a replacement $\phi\,|_R$ is such that all instances of $\alpha_{(a,i)}$ are replaced with $\alpha_{N(a)}$ and instances of $\alpha_{(a,r)}$ are replaced with $(\alpha_{(a,i)} \vee \alpha_{(a,r)})$.*

Example: $EX\neg(\alpha_{(a,i)} \vee \alpha_{(a,r)} \vee \alpha_{(b,i)})$ turns into $\neg(\alpha_{N(a)} \vee \alpha_{(a,i)} \vee \alpha_{(a,r)} \vee \alpha_{N(b)})$

Lemma 21 (EX replacement equivalence)**.** *For a formula of the form $\varphi = EX\phi$ where ϕ does not contain instances of EX and a replacement $\phi\,|_R$, it is the case that $\vdash \varphi \leftrightarrow \phi\,|_R$.*

Proof. Induction on the number of operators in ϕ.

Base case ($n = 0$): Trivial by the reduction axioms.

Induction hypothesis ($n = k$):
$\vdash EX\phi \leftrightarrow \phi\,|_R$ for a ϕ with k operators.

Induction step ($n = k + 1$): There are two possible cases:

- $EX\varphi_1 \vee \varphi_2$: By the axioms, EX distributes over $\vee$, and we have that $EX\varphi_1 \vee \varphi_2 \leftrightarrow EX\varphi_1 \vee EX\varphi_2$. By the I.H. we can replace on the previous formula such that $EX\varphi_1 \vee EX\varphi_2 \leftrightarrow \varphi_1\,|_R \vee \varphi_2\,|_R$, which is equivalent to $(\varphi_1 \vee \varphi_2)\,|_R$.

- $EX\neg\varphi$: This can be in two forms:
  - $EX\neg\neg\psi$: This is the same as $EX\psi$, which is equivalent to $\psi\ |_R$ by the I.H..
  - $EX\neg(\psi_1 \vee \psi_2)$: We consider each direction separately:
    * $\vdash\ EX\neg(\psi_1 \vee \psi_2) \rightarrow \neg(\psi_1 \vee \psi_2)\ |_R$: Since it is equivalent to $EX(\neg\psi_1 \wedge \neg\psi_2)$ and EX distributes over $\wedge$ in the $\rightarrow$ direction, $EX(\neg\psi_1 \wedge \neg\psi_2) \rightarrow EX\neg\psi_1 \wedge EX\neg\psi_2 \leftrightarrow (\neg\psi_1)\ |_R \wedge(\neg\psi_2)\ |_R$. $(\neg\psi_1)\ |_R \wedge(\neg\psi_2)\ |_R$ is equivalent to $\neg(\psi_1 \vee \psi_2)\ |_R$.
    * $\vdash\ EX\neg(\psi_1 \vee \psi_2) \leftarrow \neg(\psi_1 \vee \psi_2)\ |_R$: $\neg(\psi_1 \vee \psi_2)\ |_R$ is equivalent to $\neg(\psi_1\ |_R \vee \psi_2\ |_R)$. By the I. H., this is equivalent to $\neg(EX\psi_1 \vee EX\psi_2)$, which is equivalent to $\neg EX(\psi_1 \vee \psi_2)$ due to EX's distributive property. This is equivalent to $AX\neg(\psi_1 \vee \psi_2)$. This implies $EX\neg(\psi_1 \vee \psi_2)$.

$\square$

We prove every formula φ with no occurrences of EG and Until is equivalent to a formula without any occurrences EX. Before that we define the EX-degree of a formula. Intuitively, it is the maximum nesting of EX operators occurring in the formula.

Definition 48 (EX-degree)**.** *Let φ be a formula without any occurrence of EG and Until operators. We define the EX-degree of φ, $D(\varphi)$, as follows*

- $D(p) = 0$*, for all propositional symbol p;*
- $D(\neg\varphi) = D(\varphi)$*;*
- $D(\varphi_1 \vee \varphi_2) = max(D(\varphi_1), D(\varphi_2))$*;*
- $D(EX\varphi) = D(\varphi) + 1$*.*

Lemma 22 (Translation equivalence)**.**

$$\vdash \varphi \leftrightarrow \hat{\varphi}$$

where φ is a formula without occurrences of EG and Until, and $\hat{\varphi}$ is a formula without EX.

Proof. By induction on $D(\varphi)$, the EX-degree of φ.

Base case: $D(\varphi) = 0$. $\hat{\varphi} = \varphi$ by definition.

$D(\varphi) = 1$. Every occurrence $\psi = EX\phi$ is equivalent to its translation by lemma 17 (EX Repl), that is, $t(\psi) = \psi\ |_R$. By CTL replacement of equivalents, we can replace each of these occurrences in φ by their equivalent translation, which yields a φ' without any EX occurrences. $\hat{\varphi} = \varphi'$.

Induction hypothesis: For all φ such that $D(\varphi) < n : \vdash \varphi \leftrightarrow \hat{\varphi}$ such that $\hat{\varphi}$ is a formula without EX.

Induction step:

Consider φ with $D(\varphi) = n$. Every occurrence $\phi = EX\psi$ in φ such that $D(\phi) = 1$ is equivalent to $\phi\mid_R$ by lemma 17. By CTL replacement of equivalents, we can replace each of these occurrences in φ by their equivalent translation, which yields a φ' of degree $n-1$. By the I.H. $\vdash \varphi' \leftrightarrow \hat{\varphi}'$ where $\hat{\varphi}'$ has no EX occurrences. Therefore $\vdash \varphi \leftrightarrow \hat{\varphi}$ where $\hat{\varphi} = \hat{\varphi}'$ and $\hat{\varphi}$ has no EX occurrences. □

Theorem 3 (Completeness)**.** *For every $\varphi \in SIR - CTL$:*

$$\models \varphi \textit{ implies } \vdash \varphi$$

Proof. Suppose $\models \varphi$. By lemma 20 we have that $\vdash \varphi \leftrightarrow t(\varphi)$, and $t(\varphi)$ is a formula with no EG and Until occurrences. By lemma 22 and since the proof system is sound, we have that $\models \varphi \leftrightarrow t(\varphi) \leftrightarrow \widehat{t(\varphi)}$. The formula $\widehat{t(\varphi)}$ is propositional, therefore $\mathcal{PROP} \vdash \widehat{t(\varphi)}$ by completeness of propositional logic. We also have that $SIR - CTL \vdash \widehat{t(\varphi)}$ since $\mathcal{PROP}$ is a subsystem of SIR-CTL. Again by lemma 22 we have that $SIR - CTL \vdash \varphi$. □

Chapter 10

SIRS Models in CTL

In this chapter we explore another approach for compartmental models using CTL. This time we will go through an SIRS model where, in contrast do SIR models, agents do become susceptible to infection again after they recover. We will demonstrate the syntax and semantics of the logic and a few examples, and then proceed to define an axiomatization that is sound and complete. We show completeness via construction of a canonical model, in contrast to the previous approach of translating the formulas onto the propositional part of the logic.

10.1 SIRS model definition

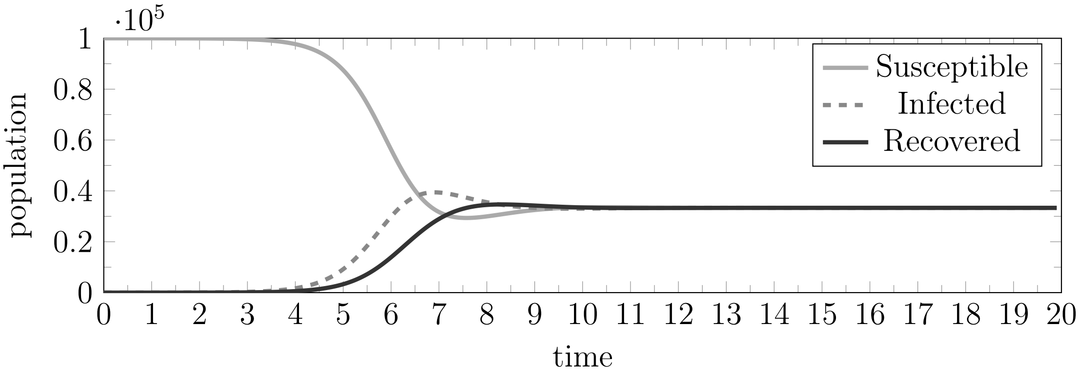
Unlike the simpler SIR model, the SIRS model allows for individuals to transition from the recovered compartment back to the susceptible compartment, representing the loss of immunity over time. This feature is particularly relevant for diseases with short-lived immunity or where multiple strains of the disease exist [61].

Definition 49 (SIRS condition sets)**.** *A condition set B is a set of tuples indicating the current state of agents. Therefore, $B \subseteq \mathcal{B} = \{(a, x) \mid a \in \mathcal{A}, x \in \{i, r\}\}$.*

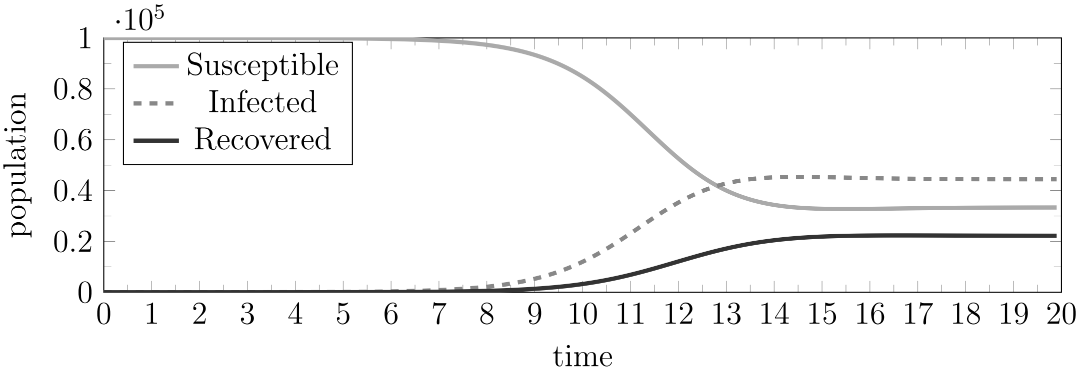
Definition 50 (SIRS Model)**.** *A model is a tuple $\mathcal{M} = (\mathcal{A}, N, I)$, where*

- *$\mathcal{A}$ is a finite set of agents;*
- *N is a neighborhood function $N : \mathcal{A} \mapsto 2^{\mathcal{A}}$ such that N is irreflexive, serial and symmetric;*
- *$I \subseteq \mathcal{B}$ is an initial condition set, such that for any agent a, (a, i) and (a, r) cannot be in I simultaneously.*

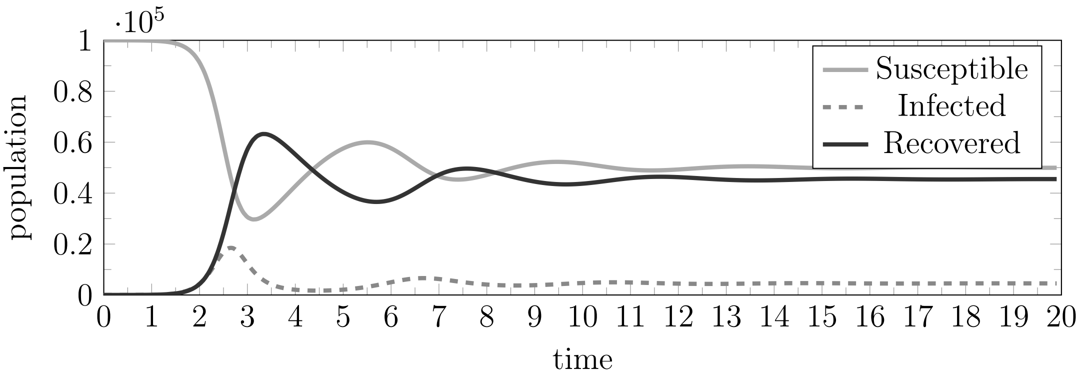
Before we proceed to the definition of the $\leq$ relation for SIRS models, we define abbreviations for the sets of infected, recovered and exposed agents. Exposed agents are those that have infected neighbours, and are not yet infected or recovered.



(a) This case demonstrates a moderately infectious disease with a reasonably quick recovery time. Infection peaks quickly and stabilizes. Population $N = 100{,}000$ and a single infected individual at the start. Infection rate is $\beta = 3$, and recovery rate $\gamma = 1$.



(b) This case demonstrates a mild infectious disease with a slow recovery time. Agents take a long time to be infected, and recovery has a long tail. Population $N = 100{,}000$ and a single infected individual at the start. Infection rate is $\beta = 1.5$, and recovery rate $\gamma = 0.5$.



(c) This case demonstrates a disease that spreads very quickly, with a moderate recovery time. In this case agents have an immunity period, and the susceptibility rate is $f = 0.5$. Waves can be seen, and the peak of infections becomes flattened over time. Population $N = 100{,}000$ and a single infected individual at the start. Infection rate is $\beta = 10$, and recovery rate $\gamma = 5$.

Figure 10.1: Multiple settings showcasing the evolution of SIRS models.

Definition 51 (Infected, recovered and exposed agent sets)**.** *Let $B \in \mathcal{B}$ be a condition set. We define the sets of Infected, Recovered and Exposed agents in B respectively as:*

$$\begin{aligned} I_B &= \{a : (a,i) \in B\} \\ R_B &= \{a : (a,r) \in B\} \\ E_B &= \{a : \{N(a) \cap I_B\} \neq \emptyset \text{ and } a \notin I_B \text{ and } a \notin R_B\} \end{aligned}$$

With these we proceed to define the $\leq$ relation for SIRS models.

Definition 52 (SIRS relation $\leq$)**.** *Let $\leq$ be the binary relation over $\mathcal{B} \times \mathcal{B}$, where*

$$B \leq B' \text{ iff } \{(a,i) \in B'\} \subseteq INF,\ REC \subseteq B' \text{ and } SUS \subseteq B'$$

where

- $INF = \{(a,i) : a \in E_B\}$ *(Infected)*
- $REC = \{(a,r) : a \in I_B\}$ *(Recovered)*
- $SUS = R_B$ *(Susceptible)*

10.2 SIRS Logic

In this section we define $\mathcal{L}_{SIRS}$, the logic for SIRS models, and a sound and complete axiomatization for it.

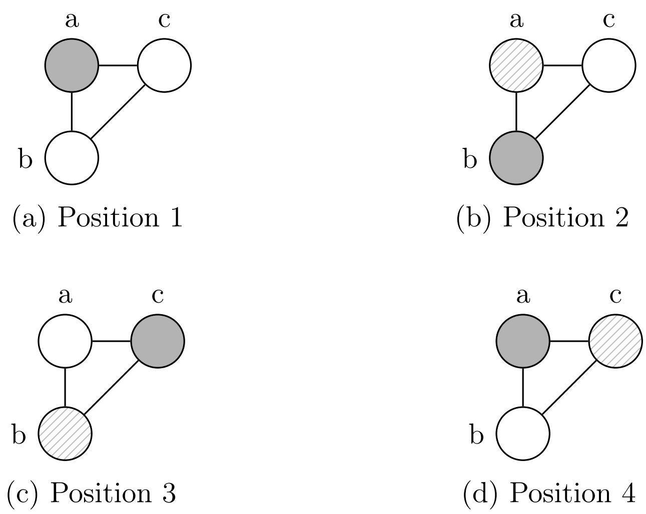
10.2.1 Language

Definition 53 (SIRS-CTL Language)**.** *The language of SIRS-CTL is given by the following BNF:*

$$\varphi := \top \mid N_{ab} \mid \alpha_{(a,i)} \mid \alpha_{(a,r)} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid EX\varphi \mid EG\varphi \mid E[\varphi_1 U \varphi_2]$$

where $a, b \in \mathcal{A}$, $\alpha \in \mathsf{B}$.

We use the standard abbreviations [16]: $\bot \equiv \neg\top$, $\varphi \wedge \phi \equiv \neg(\neg\varphi \vee \neg\phi)$, $\varphi \to \phi \equiv \neg(\varphi \wedge \neg\phi)$, $AX\varphi \equiv \neg EX\neg\varphi$, $EF\varphi \equiv E[\top U\varphi]$,$AG \equiv \neg EF\neg\varphi$, $AF \equiv \neg EG\neg\varphi$, $A(\alpha\mathcal{U}\beta) \equiv (\neg E((\neg\beta)\ \mathcal{U}\ (\neg\alpha \wedge \neg\beta))) \wedge (\neg[EG](\neg\beta))$. Intuitive readings of the operators are the same as previously introduced in definition 1.



(a) Position 1 (b) Position 2

(c) Position 3 (d) Position 4

Figure 10.2: A simple SIRS model with a reinfection loop. Notice that in this particular path, agent a infected agent b but not c in the first position. Had agent a infected both b and c the loop would not have happened, since a would be recovered and both b and c would be simultaneously infected, leaving no other agent to be infected.

10.3 Sound and complete system

The Until and EG expansion definitions are very similar to definitions 44 and 45 respectively, the difference being the range of the disjunctions which is now $|3^{\mathcal{A}}|$ to capture all possible combinations of states. The reason for this will become clearer later with lemma 23 and the soundness proofs for each of the expansions.

Definition 54 (SIRS-CTL Until expansion)**.** *Let $E[\varphi U \psi]$ be a formula. We define its Until expansion steps as*

$$
\begin{aligned}
u^0(E[\varphi U \psi]) &:= \psi \\
u^i(E[\varphi U \psi]) &:= (\varphi \wedge EXu^{i-1})
\end{aligned}
$$

Then, its Until expansion $EXP_U(\varphi U \psi)$ in a model with agent set $\mathcal{A}$ is defined as

$$
EXP_U(E[\varphi U \psi]) := \bigvee_{0 \leq i \leq |3^{\mathcal{A}}|} u^i(E[\varphi U \psi])
$$

Definition 55 (SIRS-CTL EG expansion)**.** *Let $EG\varphi$ be a formula. We define its EG expansion steps as*

$$
\begin{aligned}
g^0(EG\varphi) &:= \varphi \\
g^i(EG\varphi) &:= (\varphi \wedge EXg^{i-1})
\end{aligned}
$$

Network axioms	
$\neg N_{aa}$	Irreflexivity
$N_{ab} \leftrightarrow N_{ba}$	Symmetry
$\bigvee_{b \in \mathcal{A}} N_{ab}$	Seriality
Reduction axioms	
$EXN_{ab} \leftrightarrow N_{ab}$	Red.Ax.$EX.N$
$EX\neg(\alpha_{(a,i)} \vee \alpha_{(a,r)}) \leftrightarrow \alpha_{(a,r)}$	Red.Ax.$EX.\alpha.s$
$EX\alpha_{(a,i)} \leftrightarrow \alpha_{N(a)}$	Red.Ax.$EX.\alpha.i$
$EX\alpha_{(a,r)} \leftrightarrow \alpha_{(a,i)}$	Red.Ax.$EX.\alpha.r$
$EG\varphi \leftrightarrow EXP_{EG}(EG\varphi)$	Red.Ax.EG
$E[\varphi U\psi] \leftrightarrow EXP_U(E[\varphi U\psi])$	Red.Ax.EU
CTL axioms	
All classical propositional tautologies	
$AG\varphi \leftrightarrow \neg EF\neg\varphi$	
$EG\varphi \leftrightarrow \neg AF\neg\varphi$	
$EX(\varphi \vee \psi) \leftrightarrow EX\varphi \vee EX\psi$	
$AX\varphi \leftrightarrow \neg EX\neg\varphi$	

Table 10.1: SIRS-CTL axioms. The only axiom added from SIR-CTL is Red.Ax.$EX.\alpha.s$, which conveys the transition from being recovered to susceptible again. The abbreviation $\alpha_{N(a)}$ is presented in definition 40.

Then, its EG expansion $EXP_{EG}(\varphi)$ in a model with agent set $\mathcal{A}$ is defined as

$$EXP_{EG}(EG\varphi) := \bigvee_{0 \leq i \leq |3^{\mathcal{A}}|} g^i(EG\varphi)$$

Lemma 23 (Path cycles). *Any path s will cycle in at most $|3^{\mathcal{A}}|$ steps.*

Proof. The set of agents $\mathcal{A}$ is finite by definition. The neighborhood function is invariant for any given model, therefore it is irrelevant as a distinction between states. Each agent can be in only one condition c_i in any given step i. c_i can be either (a, i), (a, r) or neither. Therefore, the total number of combinations of agent conditions is $|3^{\mathcal{A}}|$, which implies no path can contain a sequence of unique states larger than $|3^{\mathcal{A}}|$. □

The axiom schemata for $\mathcal{L}_{SIRS}$ are presented in table 10.1.

10.3.1 Expansion equivalences

Refer to lemmas 14 and 16 for the step satisfaction of the Until and EG equivalences, respectively. Here we focus solely on the full expansion equivalence expressions, which now must account for the different range of the expansion terms.

Lemma 24 (Until expansion equivalence). $\models E[\varphi U\psi] \leftrightarrow EXP_U(E[\varphi U\psi])$.

Proof. A direct consequence of lemma 23 is that no path can contain a sequence of unique states larger than $|3^{\mathcal{A}}|$. Therefore, if $\varphi U \psi$ holds in a position i_0, then either φ holds in the entire unique sequence or ψ holds at some position i_n where $0 \leq n \leq |3^{\mathcal{A}}|$. With this, it is sufficient to expand the Until expression using axiom $A5$ up to $|3^{\mathcal{A}}|$ terms. □

Definition 56 (Until replacement)**.** *For a formula φ a replacement $\varphi\mid_U$ is such that all instances of $\psi U \phi$ are replaced with $EXP_U(\psi U \phi)$.*

Lemma 25 (Until replacement equivalence)**.** *For a formula φ and a replacement $\varphi\mid_U$, it is the case that $\vdash \varphi \leftrightarrow \varphi\mid_U$.*

Proof. Induction on the number n of operators in the formula φ.

Base case ($n = 1$): The only case affected by the replacement is $\varphi = \psi U \phi$, which is valid by direct consequence of lemma 24.

Induction hypothesis ($n = k$):

$\vdash \varphi \leftrightarrow \varphi\mid_U$ for a φ with k operators.

Induction step ($n = k + 1$): The only relevant case is $\varphi = \varphi_1 U \varphi_2$, as the rest is trivial by the I.H..

We have that $\varphi_1 U \varphi_2 = EXP_U(\varphi_1 U \varphi_2)$ by lemma 24.

By the I.H. we know that $\vdash \varphi_1 U \varphi_2 \leftrightarrow \varphi_1\mid_U U \varphi_2\mid_U$.

Therefore $\vdash EXP_U(\varphi_1 U \varphi_2) \leftrightarrow EXP_U(\varphi_1\mid_U U \varphi_2\mid_U)$, which is equivalent to $\varphi\mid_U$ by definition. □

10.3.2 Soundness

The soundness proof for SIRS-CTL consists on the validity of the items of the SIR-CTL proof presented in lemma 19, with the inclusion of the axiom Red.Ax.$EX.\alpha.s$, which captures the transition of agents becoming susceptible again after infection. All items are presented again for self-containedness.

Lemma 26 (Soundness)**.** *We will show all the axiom schemata presented are valid, in the usual sense of validity. For this, let $\mathcal{M}$ be an arbitrary $\mathcal{L}_{SIRS}$ model and $B \in \mathcal{B}$ a behavior set.*

Proof. Consider each case:

- All network axioms follow directly from definition 50;
- $\mathcal{M}, s \models EXN_{ab} \leftrightarrow N_{ab}$ follows directly from the fact that $\leq$ never alters the network structure;
- $\mathcal{M}, s \models E[\varphi U \psi] \leftrightarrow EXP_U(E[\varphi U \psi])$ follows directly from lemma 15;

- $\mathcal{M}, s \models EG\varphi \leftrightarrow EXP_{EG}(EG\varphi)$ follows directly from lemma 17;

- $\mathcal{M}, s \models EX\alpha_{(a,i)} \leftrightarrow \alpha_{N(a)}$: By definition, $\mathcal{M}, s \models EX\alpha_{(a,i)}$ if and only if there exists a path $b \in b(s)$ and $\mathcal{M}, b_1 \models \alpha_{(a,i)}$. By definition 52, $\mathcal{M}, b_1 \models \alpha_{(a,i)}$ is the case if and only if $\mathcal{M}, s \models \alpha_{(c,i)}$ for some $c \in N(a)$. Since $c \in N(a)$, the first two terms in definition 40 ensure $c \in \mathcal{N}$, and since $\mathcal{M}, s \models \alpha_{(c,i)}$, it can be concluded that $\mathcal{M}, s \models \alpha_{N(a)}$.

- $\mathcal{M}, s \models EX\alpha_{(a,r)} \leftrightarrow \alpha_{(a,i)} \vee \alpha_{(a,r)}$ follows directly from the term $REC \subseteq B'$ in definition 52.

- $\mathcal{M}, s \models EX\neg(\alpha_{(a,i)} \vee \alpha_{(a,r)}) \leftrightarrow \alpha_{(a,r)}$ follows directly from the term $SUS \subseteq B'$ in definition 52.

□

10.3.3 Completeness

CTL in general is not a compact logic. A semantics is said to be compact if and only if every finite subset of a set of sentences has a model, and so does the entire set. This is not the case for CTL, as demonstrated by the following formula:

$$\{\neg AGp\} \cup \{AX^i p \mid i \in \mathbb{N}\}$$

It can be shown that this set of CTL sentences has a model, but it has no finite model. Intuitively, this is because for any finite subset of sentences from this set, there is always a path in the system that satisfies that subset, but when taken for all $\mathbb{N}$ it cannot be the case that $\neg AGp$. Therefore, no finite model can satisfy all the sentences in this set, and thus the compactness theorem is violated.

A non-compact logic cannot be canonical [62], which means it cannot have a single canonical model built such that any valid formula holds in it, a common approach to proving completeness in many logics.

It is possible, however, to build a finitary canonical model. This consists in building a specific canonical model for a given consistent set of formulas.

We demonstrate the axiomatization for $\mathcal{L}_{SIRS}$ is complete by showing that a canonical model in CTL constructed from any given φ and the set of Network and Diffusion axioms is in fact an $\mathcal{L}_{SIRS}$ model.

Before we proceed to the proof, the definition of a maximal consistent set is necessary.

Definition 57 (Maximal consistent set)**.** *A set Φ of formulas in $\mathcal{L}_{SIRS}$ is said to be a maximal consistent set if and only if:*

- *Φ is a consistent set, i.e., there is no formula ϕ such that both ϕ and $\neg\phi$ are in Φ.*

- *For any formula ψ in $\mathcal{L}_{SIRS}$, either ψ or $\neg\psi$ can be added to Φ while maintaining its consistency, i.e., $\Phi \cup \{\psi\}$ or $\Phi \cup \{\neg\psi\}$ is consistent.*

The completeness proof goes as follows:

- We define a canonical model for an arbitrary formula φ in $\mathcal{L}_{SIRS}$;
- We demonstrate the Lindenbaum lemma, which states that, for any consistent set of logical sentences, there exists a consistent, complete, and maximal extension of that set;
- We demonstrate the Truth Lemma, which states that given a maximal consistent set for a formula and a canonical model constructed for that set, every formula contained in that maximal set must be true in the constructed canonical model;
- And finally, the completeness proof is by contraposition: we assume that a formula φ is not a tautology of $\mathcal{L}_{SIRS}$; therefore, its negation must be a consistent set;
- Then, by the Lindenbaum lemma, we can extend the negation of φ to a consistent maximal set;
- With this set, we can construct a canonical model, and according to the Truth Lemma, we know that this model satisfies the negation of φ;
- Finally, from there, we know that φ is not valid in $\mathcal{L}_{SIRS}$ since its negation was satisfied in a model, and this is what we need to conclude the proof by contraposition.

Through this entire section we assume a finite set of agents $\mathcal{A}$. See section 7.2 of [53] and lectures 1 and 2 of [63].

Definition 58 (L_{SIRS} canonical model for Γ)**.** *The canonical model construction is based on standard CTL (and modal logics in general) [62].*

Given an arbitrary formula φ, let Γ be a maximally consistent set containing φ. The canonical model $M^C = \langle N^C, I^C \rangle$ for Γ is defined as follows:

- $N^C(N_{ab}) = \{N_{ab} \mid N_{ab} \in \Gamma\}$
- $I^C(\Gamma) = \{(a,i) \mid (a,i) \in \Gamma\} \cup \{(a,r) \mid (a,r) \in \Gamma\}$

Notice there are many possible maximally consistent sets containing φ, and therefore many possible canonical models for a given Γ. This is, however, not an issue for the completeness proof: the model M^C is a SIRS-CTL model because the neighborhood function is built ranging over a maximal consistent set which is, crucially, consistent with the irreflixivity, symmetry and seriality axioms.

For the completeness proof we will need to demonstrate Lindenbaum's lemma. The lemma states that for any consistent set of propositional logical sentences, there exists a consistent, complete and maximal extension of that set.

A consistent set of sentences is one in which no sentence in the set contradicts another sentence in the set. A complete extension of a set of sentences is one that includes all the logical consequences of the original set, and a maximal extension is one that cannot be further extended without becoming inconsistent.

Lemma 27 (Lindenbaum)**.** *Every consistent set can be extended to a maximally consistent set.*

Proof. Take an enumeration of all L_{SIRS} formulas, which is possible because the set of agents is finite, and therefore the atomic propositions are also finite. Let φ_k be the k-th formula in this enumeration. Now, consider Φ an arbitrary consistent set, and the following sequence of sets of formulas:

$$\Gamma_0 = \Phi \tag{10.1}$$

$$\Gamma_n = \begin{cases} \Gamma_k \cup \{\varphi_{k+1}\} & \text{if } \Gamma_k \cup \{\varphi_{k+1}\} \text{ is consistent} \\ \Gamma_k & \text{otherwise} \end{cases} \tag{10.2}$$

Take $\Gamma = \Gamma_0 \cup \Gamma_1 \cup \Gamma_2 \cup \ldots \cup \Gamma_N$, where N is the last N-th formula in the enumeration. Γ is a consistent set because it is the union of a chain of consistent sets. Finally, Γ has to be maximal, otherwise there would be a formula in L_{SIRS} not present in the enumeration. □

Lemma 28 (Auxiliary)**.** *If* Γ *and* Δ *are maximal consistent sets, then*

1. $\varphi \in \Gamma$ *iff* $\neg\varphi \notin \Gamma$*;*

2. $\varphi \wedge \phi \in \Gamma$ *iff* $\varphi \in \Gamma$ *and* $\phi \in \Gamma$*;*

3. $\Gamma \leq \Delta$ *iff* $\{\varphi \mid X\varphi \in \Gamma\} \subseteq \Delta$*.*

Proof. 1. $\Rightarrow$ Suppose $\varphi \in \Gamma$. By consistency $\neg\varphi \notin \Gamma$.

$\Leftarrow$ Suppose $\neg\varphi \notin \Gamma$. By maximality $\Gamma \cup \{\neg\varphi\} \vdash \bot$. But then $\Gamma \vdash \varphi$. Therefore $\varphi \in \Gamma$, because Γ is maximally consistent and therefore deductively closed (i.e., it contains every formula than can be logically deduced from Γ, including φ).

2. $\varphi \wedge \phi \in \Gamma$ is equivalent to $\varphi \in \Gamma$ and $\phi \in \Gamma$ because Γ is deductively closed.

3. $\Rightarrow$ Direct from the definition of $\leq$ and the fact that Γ and Δ are maximally consistent.

   $\Leftarrow$ Suppose $\{\varphi \mid X\varphi \in \Gamma\} \subseteq \Delta$. For every atomic proposition $(a, x) \in \Gamma$, there is an $X\varphi$ according to the diffusion axioms. By supposition $\varphi \in \Delta$, therefore it must be that $\Gamma \leq \Delta$.

$\square$

Lemma 29 (Truth)**.** *Let Γ be a maximally consistent set containing φ. Let $M^C = \langle N^C, I^C \rangle$ be the canonical model for Γ. For all $\varphi \in \Gamma$:*

$$\varphi \in \Gamma \textit{ iff } M^C, 0 \models \varphi$$

Proof. By induction on $\varphi\mid_U$ (definition 56).

Base case Suppose φ is a propositional variable p of form N_{ab} or (a, x). Then by definition of N and I, $p \in \Gamma$ iff $\Gamma \in N^C$ or $\Gamma \in I^C$, respectively. By semantics, this is equivalent to $M^C, 0 \models p$.

Induction hypothesis For a given φ and ψ it is the case that $\varphi \in \Gamma$ iff $M^C, 0 \models \varphi$, and $\psi \in \Gamma$ iff $M^C, 0 \models \psi$.

Induction step We distinguish the following cases:

- $\neg\varphi$: $\neg\varphi \in \Gamma$ is equivalent to $\varphi \notin \Gamma$ by item 1 of lemma 28. By the induction hypothesis this is equivalent to $M^C, 0 \not\models \varphi$, which by the semantics is equivalent to $M^C, 0 \models \neg\varphi$.

- $\varphi \wedge \psi$: $\varphi \wedge \psi \in \Gamma$ is equivalent to $\varphi \in \Gamma$ and $\psi \in \Gamma$ by item 2 of lemma 28. By the induction hypothesis this is equivalent to $M^C, 0 \models \varphi$ and $M^C, 0 \models \psi$, which by semantics is equivalent to $M^C \models \varphi \wedge \psi$.

- $X\varphi$: $\Rightarrow$: Suppose $X\varphi \in \Gamma$. Take an arbitrary maximal consistent set Δ. Suppose $\Gamma \leq \Delta$. By item 3 of lemma 28 and since both sets are maximally consistent, it must be the case that $\varphi \in \Delta$. By the I.H. this is iff $\Delta, 0 \models \varphi$. By the semantics definition, and since $\Gamma \leq \Delta$, this is equivalent to $w, 1 \models \varphi$ where $w_0 = \Gamma$. Since Γ is maximally consistent, it must be the case that $w, 0 \models X\varphi$.

  $\Leftarrow$: Suppose $w, 0 \models X\varphi$. Therefore $w, 1 \models \varphi$, where $w_0 = \Gamma$ and $w_1 = \Delta$, such that $\Gamma \leq \Delta$. By the I.H. it is the case that $\varphi \in \Delta$. By item 3 of lemma 28, it is the case that $X\varphi \in \Gamma$.

$\square$

Theorem 4 (Completeness)**.** *For every* $\varphi \in \mathcal{L}_{SIRS}$

$$\models_{L_{SIRS}} \varphi \Rightarrow \vdash_{L_{SIRS}} \varphi \tag{10.3}$$

Proof. This is a standard proof by contraposition, as shown in [53]. Suppose $\nvdash_{L_{SIRS}} \varphi$. Then $\{\neg\varphi\}$ is a consistent set. By lemma 27 we have that $\{\neg\varphi\}$ is a subset of a maximal consistent set Γ. Using definition 58 we construct a canonical model $\mathcal{M}_\Gamma$ from Γ, such that $\mathcal{M}_\Gamma, 0 \models_{L_{SIRS}} \neg\varphi$ by lemma 29. Therefore $\not\models_{L_{SIRS}} \varphi$. $\square$

Chapter 11

Conclusion and future works

The paper presented a class of models for the study of social networks, based on threshold models as seen across the literature. With this, we have developed a logic with the same operators as LTL, but for a restricted class of models focused specifically on the study of social networks, proved its soundness and its completeness via a translation argument, and briefly discussed model checking complexity. With this framework one is able to capitalize on existing model checking solutions for LTL, which we believe is a good incentive to pursue this line of work.

Given the previous conclusion, a natural future work is to explore these model checkers and analyze the evolution of some network instances and compare them against data from real social networks.

In section 8 we mention a few preconditions for the validity of the soundness and completeness proofs, all of which remain valid in the SIR model and allowed us to expand the proofs for these models. We believe this to be case for any arbitrary model definition, and that is an interesting generalization for this framework.

Another potential future work is to extend the framework to include more complex behaviors beyond simple threshold adoption. For example, one could consider models where agents are influenced by multiple thresholds or have different preferences for adopting a behavior based on their social network connections.

Regarding SIR and SIRS models, a point of further research would be to compare the evolution of our logic models presented in the paper against the standard non-discrete differential approach. While the logic-based models offer a discrete and intuitive framework for modeling the spread of infectious diseases, it is unclear how they compare to more traditional continuous models in terms of accuracy and computational efficiency. By comparing the two approaches, one could identify the strengths and limitations of each and determine which approach is better suited for different scenarios.

Finally, it would be interesting to investigate the relationship between the threshold-based models presented in the paper and other well-known models of social influence, such as the DeGroot model [64]. This could provide insights into the similarities and differences between these models, as well as identify the conditions under which they lead to similar or divergent outcomes. Such comparisons could shed light on the strengths and limitations of different modeling approaches for social networks.

11.1 Acknowledgements

We would like to thank the research agencies CNPq, CAPES and FAPERJ for their support.

References

[1] DOBRIKOV, I., LEUSCHEL, M., PLAGGE, D. "LTL Model Checking under Fairness in ProB". In: *International Conference on Software Engineering and Formal Methods*, v. 9763, pp. 204–211, 07 2016. ISBN: 978-3-319-41590-1. doi: 10.1007/978-3-319-41591-8_14.

[2] BALTAG, A., CHRISTOFF, Z., KRÆMMER RENDSVIG, R., et al. "Dynamic Epistemic Logics of Diffusion and Prediction in Social Networks", *Studia Logica*, v. 107, pp. 489–531, 07 2018. doi: 10.1007/s11225-018-9804-x.

[3] KINNUNEN, J. "Gabriel Tarde as a Founding Father of Innovation Diffusion Research", *Acta Sociologica*, v. 39, n. 4, pp. 431–442, 1996. doi: 10.1177/000169939603900404. Available at: <`https://doi.org/10.1177/000169939603900404`>.

[4] RYAN, B., GROSS, N. "The diffusion of hybrid seed corn in two Iowa communities", *Rural Sociology*, v. 8, n. 1, pp. 15–24, 1943.

[5] WASSERMAN, S., FAUST, K. "Social Network Analysis in the Social and Behavioral Sciences". In: *Social Network Analysis: Methods and Applications*, p. 3–27, Cambridge, UK, Cambridge University Press, 1994. doi: 10.1017/CBO9780511815478.002.

[6] MACHADO, V., BENEVIDES, M. "Temporal logic for social networks", *Journal of Logic and Computation*, v. 32, n. 6, pp. 1088–1108, 03 2022. ISSN: 0955-792X. doi: 10.1093/logcom/exac023. Available at: <`https://doi.org/10.1093/logcom/exac023`>.

[7] CLEAVELAND, R., SOKOLSKY, O. "CHAPTER 6 - Equivalence and Preorder Checking for Finite-State Systems". In: Bergstra, J., Ponse, A., Smolka, S. (Eds.), *Handbook of Process Algebra*, Elsevier Science, pp. 391–424, Amsterdam, 2001. ISBN: 978-0-444-82830-9. doi: https://doi.org/10.1016/B978-044482830-9/50024-2. Available at: <`https://www.sciencedirect.com/science/article/pii/B9780444828309500242`>.

[8] PNUELI, A. “The temporal logic of programs”. In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pp. 46–57, 1977. doi: 10.1109/SFCS.1977.32.

[9] VARDI, M. “Branching vs. Linear Time: Final Showdown”. In: *Tools and Algorithms for the Construction and Analysis of Systems*, v. 2031, pp. 1–22, 03 2001. ISBN: 978-3-540-41865-8. doi: 10.1007/3-540-45319-9_1.

[10] SUN, J., LIU, Y., DONG, J. S., et al. “PAT: Towards Flexible Verification under Fairness”. In: *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, v. 5643, *Lecture Notes in Computer Science*, pp. 709–714. Springer, 2009.

[11] BAIER, C., KATOEN, J.-P. *Principles of Model Checking (Representation and Mind Series).* Cambridge, MA, The MIT Press, 2008. ISBN: 026202649X.

[12] CLARKE, E., GRUMBERG, O., HAMAGUCHI, K. “Another Look at LTL Model Checking”. In: *FORMAL METHODS IN SYSTEM DESIGN*, pp. 415–427. Springer-Verlag, 1994.

[13] ROZIER, K. Y. “Linear Temporal Logic Symbolic Model Checking”, *Computer Science Review*, v. 5, n. 2, pp. 163–203, 2011. ISSN: 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2010.06.002. Available at: <`https://www.sciencedirect.com/science/article/pii/S1574013710000407`>.

[14] WAHL, T. “Notions of Fairness and Liveness”. 2013.

[15] CLARKE, E. M., EMERSON. “Design and synthesis of synchronization skeletons using branching time temporal logic”. In: *Workshop on Logics of Programs, Lecture Notes in Computer Science 131*, pp. 52–71. Springer-Verlag, 1981.

[16] CLARKE, E. M., GRUMBERG, O., PELED, D. A. *Model checking.* Cambridge, MA, MIT Press, 2001. ISBN: 978-0-262-03270-4. Available at: <`http://books.google.de/books?id=Nmc4wEaLXFEC`>.

[17] EMERSON, E. A., HALPERN, J. Y. “"Sometimes" and "Not Never" Revisited: On Branching versus Linear Time (Preliminary Report)”. In: *Proceedings of the 10th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL ’83, p. 127–140, New York, NY, USA, 1983. Association for Computing Machinery. ISBN: 0897910907. doi: 10.1145/567067.567081. Available at: <`https://doi.org/10.1145/567067.567081`>.

[18] VARDI, M. "Linear vs. Branching Time: A Complexity-Theoretic Perspective", *Proceedings - Symposium on Logic in Computer Science*, 06 1998. doi: 10.1109/LICS.1998.705674.

[19] VISSER, W., BARRINGER, H. "Practical CTL* model checking: Should SPIN be extended?" *International Journal on Software Tools for Technology Transfer (STTT)*, v. 2, pp. 350–365, 01 2000. doi: 10.1007/s100090050042.

[20] MORENO, J., JENNINGS, H. *Who Shall Survive?* New York. R. S. Woodworth, Nervous and mental disease Publishing Company, 1934. Available at: <https://books.google.com.br/books?id=72MzzgEACAAJ>.

[21] SCOTT, J. "Social Network Analysis". Jan 2017. Available at: <https://methods.sagepub.com/book/social-network-analysis-4e>.

[22] DEGROOT, M. H. "Reaching a Consensus", *Journal of the American Statistical Association*, v. 69, n. 345, pp. 118–121, 1974. ISSN: 01621459. Available at: <http://www.jstor.org/stable/2285509>.

[23] LEHRER, K. "When Rational Disagreement is Impossible", *Noûs*, v. 10, n. 3, pp. 327–332, 1976. ISSN: 00294624, 14680068. Available at: <http://www.jstor.org/stable/2214612>.

[24] APT, K. R., MARKAKIS, E. "Diffusion in Social Networks with Competing Products". In: Persiano, G. (Ed.), *Algorithmic Game Theory*, pp. 212–223, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN: 978-3-642-24829-0.

[25] AZIMIPOUR, S., NAUMOV, P. "Lighthouse Principle for Diffusion in Social Networks", *Journal of Applied Logics - IFCoLog Journal of Logics and their Applications*, v. 5, n. 1, pp. 97–121, 2018.

[26] CHRISTOFF, Z., NAUMOV, P. "Diffusion in social networks with recalcitrant agents", *Journal of Logic and Computation*, v. 29, n. 1, pp. 53–70, 2019. doi: 10.1093/logcom/exy037.

[27] RENDSVIG, R. K. "Diffusion, Influence and Best-Response Dynamics in Networks: An Action Model Approach", *CoRR*, v. abs/1708.01477, 2017. Available at: <http://arxiv.org/abs/1708.01477>.

[28] LIU, F., SELIGMAN, J., GIRARD, P. "Logical Dynamics of Belief Change in the Community", *synthese*, v. 191, pp. 2403–2431, 07 2014. doi: 10.1007/s11229-014-0432-3.

[29] GRANDI, U., LORINI, E., NOVARO, A., et al. "Strategic Disclosure of Opinions on a Social Network". In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, p. 1196–1204, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

[30] BELARDINELLI, F., GROSSI, D. "On the Formal Verification of Diffusion Phenomena in Open Dynamic Agent Networks". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, p. 237–245, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems. ISBN: 9781450334136.

[31] ZHEN, L., SELIGMAN, J. "A Logical Model of the Dynamics of Peer Pressure", *Electronic Notes in Theoretical Computer Science*, v. 278, pp. 275 – 288, 2011. ISSN: 1571-0661. doi: https://doi.org/10.1016/j.entcs.2011.10.021. Available at: <http://www.sciencedirect.com/science/article/pii/S1571066111001496>. Proceedings of the 7th Workshop on Methods for Modalities (M4M'2011) and the 4th Workshop on Logical Aspects of Multi-Agent Systems (LAMAS'2011).

[32] SELIGMAN, J., LIU, F., GIRARD, P. "Facebook and the Epistemic Logic of Friendship", *CoRR*, v. abs/1310.6440, 2013. Available at: <http://arxiv.org/abs/1310.6440>.

[33] CHRISTOFF, Z., HANSEN, J. U., PROIETTI, C. "Reflecting on Social Influence in Networks", *J. of Logic, Lang. and Inf.*, v. 25, n. 3–4, pp. 299–333, dez. 2016. ISSN: 0925-8531. doi: 10.1007/s10849-016-9242-y. Available at: <https://doi.org/10.1007/s10849-016-9242-y>.

[34] ROSS, R. "An application of the theory of probabilities to the study of a priori pathometry.—Part I", *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, v. 92, n. 638, pp. 204–230, 1916. doi: 10.1098/rspa.1916.0007. Available at: <https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1916.0007>.

[35] KERMACK, W. O., MCKENDRICK, A. G., WALKER, G. T. "A contribution to the mathematical theory of epidemics", *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, v. 115, n. 772, pp. 700–721, 1927. doi: 10.1098/rspa.1927.0118. Available at: <https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1927.0118>.

[36] KEELING, M. J., EAMES, K. T. D. “Networks and epidemic models”, *Journal of The Royal Society Interface*, v. 2, pp. 295 – 307, 2005.

[37] CHRISTOFF, Z., HANSEN, J. U. “A logic for Diffusion in Social Networks”, *Journal of Applied Logic*, v. 13, n. 1, pp. 48 – 77, 2015. ISSN: 1570-8683. doi: https://doi.org/10.1016/j.jal.2014.11.011. Available at: <`http://www.sciencedirect.com/science/article/pii/S1570868314000858`>.

[38] VERMA, R., TIWARI, S. P., UPADHYAY, R. K. “Dynamical Behaviors of Fuzzy SIR Epidemic Model”. In: Kacprzyk, J., Szmidt, E., Zadrożny, S., et al. (Eds.), *Advances in Fuzzy Logic and Technology 2017*, pp. 482–492. Springer International Publishing, 2018. ISBN: 978-3-319-66827-7.

[39] MASSAD, E., ORTEGA, N., BARROS, L., et al. *Fuzzy Logic in Action: Applications in Epidemiology and Beyond*, v. 232. Heidelberg, Germany, Springer Berlin Heidelberg, 01 2008. ISBN: 978-3-540-69092-4. doi: 10.1007/978-3-540-69094-8.

[40] CONNOLLY, S., GILBERT, D., HEINER, M. “From Epidemic to Pandemic Modelling”, *Frontiers in Systems Biology*, v. 2, jul. 2022. doi: 10.3389/fsysb.2022.861562. Available at: <`https://doi.org/10.3389/fsysb.2022.861562`>.

[41] CLARKE, E. M., EMERSON, E. A. “Design and synthesis of synchronization skeletons using branching time temporal logic”. In: Kozen, D. (Ed.), *Logics of Programs*, pp. 52–71, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg. ISBN: 978-3-540-39047-3.

[42] QUEILLE, J. P., SIFAKIS, J. “Specification and verification of concurrent systems in CESAR”. In: Dezani-Ciancaglini, M., Montanari, U. (Eds.), *International Symposium on Programming*, pp. 337–351, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg. ISBN: 978-3-540-39184-5.

[43] VERMA, S., LEE, P., HARRIS, I. G. “Error Detection Using Model Checking vs. Simulation”. In: *2006 IEEE International High Level Design Validation and Test Workshop*, pp. 55–58, 2006. doi: 10.1109/HLDVT.2006.319964.

[44] FAINEKOS, G. E., GIRARD, A., PAPPAS, G. J. “Temporal Logic Verification Using Simulation”. In: Asarin, E., Bouyer, P. (Eds.), *Formal Modeling and Analysis of Timed Systems*, pp. 171–186, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN: 978-3-540-45031-3.

[45] PETIOT, G., KOSMATOV, N., BOTELLA, B., et al. "How testing helps to diagnose proof failures", *Formal Aspects of Computing*, v. 30, n. 6, pp. 629–657, Nov 2018. ISSN: 1433-299X. doi: 10.1007/s00165-018-0456-4. Available at: <https://doi.org/10.1007/s00165-018-0456-4>.

[46] BURCH, J., CLARKE, E., MCMILLAN, K., et al. "Symbolic model checking: 1020 States and beyond", *Information and Computation*, v. 98, n. 2, pp. 142–170, 1992. ISSN: 0890-5401. doi: https://doi.org/10.1016/0890-5401(92)90017-A. Available at: <https://www.sciencedirect.com/science/article/pii/089054019290017A>.

[47] BIERE, A., CIMATTI, A., CLARKE, E., et al. "Symbolic Model Checking without BDDs". In: Cleaveland, W. R. (Ed.), *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 193–207, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN: 978-3-540-49059-3.

[48] CHEN, J., WU, J. "GMC: A performance model checker for concurrent systems", *ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and Engineering, Proceedings*, v. 3, 08 2010. doi: 10.1109/ICACTE.2010.5579829.

[49] DAM, M. "Lecture notes in Temporal Logic". 1 2021. Available at: <http://www.csc.kth.se/~mfd/Courses/Temporal_logic/course_description.php>.

[50] BURGESS, J. P., OTHERS. "Axioms for tense logic. I."Since"and"until"." *Notre Dame Journal of Formal Logic*, v. 23, n. 4, pp. 367–374, 1982.

[51] FINGER, M., GABBAY, D. M. "Adding a Temporal Dimension to a Logic System", *Journal of Logic, Language, and Information*, v. 1, n. 3, pp. 203–233, 1992. ISSN: 09258531, 15729583. Available at: <http://www.jstor.org/stable/40180020>.

[52] VENEMA, Y. "Temporal Logic". In: *The Blackwell Guide to Philosophical Logic*, cap. 10, pp. 203–223, Oxford, UK, John Wiley and Sons, Ltd, 2017. ISBN: 9781405164801. doi: https://doi.org/10.1002/9781405164801.ch10. Available at: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781405164801.ch10>.

[53] DITMARSCH, H. V., VAN DER HOEK, W., KOOI, B. *Dynamic Epistemic Logic*. 1st ed. New York, NY, Springer Publishing Company, Incorporated, 2007. ISBN: 1402058381.

[54] KOOI, B., BENTHEM, J. "Reduction axioms for epistemic actions", *AiML-2004: Advances in Modal Logic*, pp. 197–211, 01 2004.

[55] GABBAY, D., PNUELI, A., SHELAH, S., et al. "On the Temporal Analysis of Fairness". In: *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '80, p. 163–173, New York, NY, USA, 1980. Association for Computing Machinery. ISBN: 0897910117. doi: 10.1145/567446.567462. Available at: <`https://doi.org/10.1145/567446.567462`>.

[56] CLARKE, E., WANG, Q. "2^5 Years of Model Checking". In: *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, pp. 26–40, 06 2014. ISBN: 978-3-662-46822-7. doi: 10.1007/978-3-662-46823-4_2.

[57] CIMATTI, A., GIUNCHIGLIA, E., PISTORE, M., et al. "Integrating BDD-Based and SAT-Based Symbolic Model Checking". In: *Proceedings of the 4th International Workshop on Frontiers of Combining Systems*, FroCoS '02, p. 49–56, Berlin, Heidelberg, 2002. Springer-Verlag. ISBN: 3540433813.

[58] CAVADA, R., CIMATTI, A., DORIGATTI, M., et al. "The nuXmv Symbolic Model Checker". In: Biere, A., Bloem, R. (Eds.), *CAV*, v. 8559, *Lecture Notes in Computer Science*, pp. 334–342. Springer, 2014. ISBN: 978-3-319-08866-2.

[59] CAVADA, R., CIMATTI, A., JOCHIM, C. A., et al. *NuSMV 2.6 User Manual*. NuSMV Staff, 2015.

[60] BEN-ARI, M., MANNA, Z., PNUELI, A. "The temporal logic of branching time". In: *POPL '81: Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 164–176. ACM Press, 1981. ISBN: 0-89791-029-X. doi: http://doi.acm.org/10.1145/567532.567551.

[61] VAKIL, V., TRAPPE, W. "Projecting the Pandemic Trajectory through Modeling the Transmission Dynamics of COVID-19", *International Journal of Environmental Research and Public Health*, v. 19, n. 8, 2022. ISSN: 1660-4601. doi: 10.3390/ijerph19084541. Available at: <`https://www.mdpi.com/1660-4601/19/8/4541`>.

[62] BLACKBURN, P., DE RIJKE, M., VENEMA, Y. "Modal Logic", *Studia Logica*, v. 76, n. 1, pp. 142–148, 2001.

[63] DAM, M. "Lecture notes in Temporal Logic". 2019. Available at: <https://www.csc.kth.se/~mfd/Courses/Temporal_logic/course_description.php>.

[64] HANSEN, J. U. "Reasoning about opinion dynamics in social networks", *Journal of Logic and Computation*, v. 29, n. 7, pp. 1121–1137, 12 2015. ISSN: 0955-792X. doi: 10.1093/logcom/exv083. Available at: <https://doi.org/10.1093/logcom/exv083>.

Appendix A

Proof of lemma 6

Proof. Proof by induction on the length $|\varphi| = n$ of the formula φ.
Base case $(n = 1)$: $\varphi = N_{ab}$ or $\varphi = \beta_a$

This is straightforward from definition 16.

Induction hypothesis: $\vdash \varphi \leftrightarrow t_u(\varphi)$ for $|\varphi| \leq n$.
Induction step:

We have four cases:

1. $\varphi = \neg\psi$:

   By the I. H., $\vdash \psi \leftrightarrow t_u(\psi)$.
   Thus $\vdash \neg\psi \leftrightarrow \neg t_u(\psi)$ by replacement (proposition 2), and
   $\vdash \neg\psi \leftrightarrow t_u(\neg\psi)$ by definition 16.

2. $\varphi = \psi_1 \wedge \psi_2$:

   By the I. H., $\vdash \psi_1 \leftrightarrow t_u(\psi_1)$ and $\vdash \psi_2 \leftrightarrow t_u(\psi_2)$.
   Thus $\vdash \psi_1 \wedge \psi_2 \leftrightarrow t_u(\psi_1) \wedge t_u(\psi_2)$ by replacement, and
   $\vdash \psi_1 \wedge \psi_2 \leftrightarrow t_u(\psi_1 \wedge \psi_2)$ by definition 16.

3. $\varphi = X\psi$:

   By the I. H., $\vdash \psi \leftrightarrow t_u(\psi)$.
   Thus $\vdash X\psi \leftrightarrow Xt_u(\psi)$ by the Next rule (proposition 1).
   Thus $\vdash X\psi \leftrightarrow t_u(X\psi)$ by definition 16.

4. $\varphi = \psi_1 U \psi_2$:

   First we will show that $\vdash t_u(EXP_U(\psi_1 U \psi_2)) \leftrightarrow EXP_U(t_u(\psi_1) U t_u(\psi_2))$. Consider $EXP_U(t_u(\psi_1) U t_u(\psi_2)) = \bigvee_{0 \leq i \leq |\mathcal{A}|} u^i(t_u(\psi_1) U t_(\psi_2)) =$
   $= u^0 \vee u^1 \vee \ldots = t_u(\psi_2) \vee (t(\psi_1) \wedge Xt_u(\psi_2)) \vee \ldots.$

   Using 1., 2. and 3., we can bring the translation operator outside yielding $\vdash t_u(EXP_U(\psi_1 U \psi_2)) \leftrightarrow EXP_U(t_u(\psi_1) U t_u(\psi_2))$.

Considering the above and definition 16, we have
$t_u(\psi_1 U \psi_2) = t_u(EXP_U(\psi_1 U \psi_2)) = EXP_U(t_u(\psi_1) U t_u(\psi_2))$.
Using axiom $Red.Ax.U$, we have
$\vdash EXP_U(t_u(\psi_1) U t_u(\psi_2)) \leftrightarrow (t_u(\psi_1) U t_u(\psi_2))$.

By the I. H., $\vdash \psi_1 \leftrightarrow t_u(\psi_1)$ and $\vdash \psi_2 \leftrightarrow t_u(\psi_2)$. And so
$\vdash EXP_U(t_u(\psi_1) U t_u(\psi_2)) \leftrightarrow (\psi_1 U \psi_2)$.

Thus $\vdash t_u(EXP_U(\psi_1 U \psi_2)) \leftrightarrow (\psi_1 U \psi_2)$, and $\vdash t_u(\psi_1 U \psi_2)\varphi \leftrightarrow (\psi_1 U \psi_2)$ by definition 16.

$\square$

Appendix B

Proof of lemma 8

Proof. We analyze case by case:

Case $c(\varphi) \geq c(\phi)$ if $\phi \in Sub(\varphi)$:

By induction on φ.

Base case: if φ is β_a or N_{ab}, its complexity is 1 and it is its only subformula.

Induction hypothesis: $c(\varphi) \geq c(\phi)$ if $\phi \in Sub(\varphi)$.

Induction step:

- Negation ($\neg\varphi$): ϕ is a subformula of $\neg\varphi$, therefore ϕ is either $\neg\varphi$ itself or a subformula of φ. In the first case it follows directly that $c(\neg\varphi) \geq c(\phi)$. In the second case, we have that $c(\neg\varphi) = 1 + c(\varphi)$, and since ϕ is a subformula of φ, it follow directly by the I. H. that $c(\neg\varphi) \geq c(\phi)$.

- Conjunction ($\varphi \wedge \varphi'$): ϕ is a subformula of $\varphi \wedge \varphi'$, therefore ϕ is either $\varphi \wedge \varphi'$ itself or a subformula of φ or φ'. In the first case it follows directly that $c(\varphi \wedge \varphi') \geq c(\phi)$. In the second case, we have that $c(\varphi \wedge \varphi') = 1 + max(c(\varphi), c(\varphi'))$, and since ϕ is a subformula of either φ or φ', it follow directly by the I. H. that $c(\varphi \wedge \varphi') \geq c(\phi)$.

- Next ($X\varphi$): ϕ is a subformula of $X\varphi$, therefore ϕ is either $X\varphi$ itself or a subformula of φ. In the first case it follows directly that $c(X\varphi) \geq c(\phi)$. The second case can be broken down in two: formulas of the form $X\beta_a$, and otherwise. The first case is trivial, and for the second case we have that $c(X\varphi) = 2 \cdot c(\varphi)$, and since ϕ is a subformula of φ, it follow directly by the I. H. that $c(X\varphi) \geq c(\phi)$.

Case $c(XN_{ab}) > c(N_{ab})$:

$$c(XN_{ab}) = 2 \cdot c(N_{ab}) = 2$$
$$\text{and}$$
$$c(N_{ab}) = 1$$

Case $c(X\beta_a) > c(\beta_a \vee \beta_{N(a)\geq\theta})$:

$$\begin{aligned}
c(X\beta_a) &= 4 + 2 \cdot |\mathcal{A}|^3 \\
&\text{and} \\
c(\beta_a \vee \beta_{N(a)\geq\theta}) &= c(\neg(\neg\beta_a \wedge \neg\beta_{N(a)\geq\theta})) \\
&= 1 + c(\neg\beta_a \wedge \neg\beta_{N(a)\geq\theta}) \\
&= 2 + max(c(\neg\beta_a), c(\neg\beta_{N(a)\geq\theta})) \\
&= 2 + max(1 + c(\beta_a), 1 + c(\beta_{N(a)\geq\theta})) \\
&= 2 + max(2, 1 + c(\beta_{N(a)\geq\theta})) \\
&= 2 + max(2, 1 + 2 \cdot |\mathcal{A}|^3) \\
&= 3 + 2 \cdot |\mathcal{A}|^2
\end{aligned}$$

$3 + 2 \cdot |\mathcal{A}|^3$ is less than $4 + 2 \cdot |\mathcal{A}|^3$.

Case $c(X(\varphi \wedge \psi)) > c(X\varphi \wedge X\psi)$:

$$\begin{aligned}
c(X(\varphi \wedge \psi)) &= 2 \cdot c(\varphi \wedge \psi) = 2 \cdot (1 + max(c(\varphi), c(\psi))) \\
&= 2 + 2 \cdot max(c(\varphi), c(\psi)) \\
&\text{and} \\
c(X\varphi \wedge X\psi) &= 1 + max(c(X\varphi), c(X\psi)) = 1 + max(2 \cdot c(\varphi), 2 \cdot c(\psi)) \\
&= 1 + 2 \cdot max(c(\varphi), c(\psi))
\end{aligned}$$

$1 + 2 \cdot max(c(\varphi), c(\psi))$ is less than $2 + 2 \cdot max(c(\varphi), c(\psi))$.

Case $c(X\neg\varphi) > c(\neg X\varphi)$:

$$\begin{aligned} c(X\neg\varphi) &= 2 \cdot c(\neg\varphi) = 2 \cdot (1 + c(\varphi)) \\ &= 2 + 2 \cdot c(\varphi) \\ &\text{and} \\ c(\neg X\varphi) &= 1 + c(X\varphi) = 1 + 2 \cdot c(\varphi) \end{aligned}$$

$1 + 2 \cdot c(\varphi)$ is less than $2 \cdot (1 + c(\varphi))$.

Case $c(XX\varphi) > c(Xt(X\varphi))$:

$$\begin{gathered} c(XX\varphi) = 2 \cdot c(X\varphi) \\ \text{and} \\ c(Xt(X\varphi)) = 2 \cdot c(t(X\varphi)) \end{gathered}$$

Now we show $c(X\varphi) > c(t(X\varphi))$ by induction on the length of φ.

Base case: if φ has length 1, then it is either β_a or N_{ab}:

$$\begin{gathered} c(X\varphi) = 2 \cdot 1 = 2 \\ \text{and} \\ c(t(X\varphi)) = c(N_{ab}) = 1 \\ \text{or} \\ c(t(X\varphi)) = c(\beta_a) = 1 \end{gathered}$$

Induction hypothesis: $c(X\varphi) > c(t(X\varphi))$ for $|\varphi| \leq n$.

Induction step:

Case $\varphi = \neg\phi$:

$$\begin{aligned} c(X\varphi) &= c(X\neg\phi) = c(\neg X\phi) = 1 + c(X\phi) \\ &\text{and} \\ c(t(X\varphi)) &= c(t(X\neg\phi)) = c(t(\neg X\phi)) = c(\neg t(X\phi)) \\ &= 1 + c(t(X\phi)) \end{aligned}$$

By I. H., we can conclude $1 + c(X\phi) > 1 + c(t(X\phi))$, therefore $c(X\varphi) > c(t(X\varphi))$.

Case $\varphi = \phi_1 \wedge \phi_2$:

$$\begin{aligned} c(X\varphi) &= c(X(\phi_1 \wedge \phi_2)) = 2 \cdot c(\phi_1 \wedge \phi_2) \\ &= 2 \cdot (1 + max(c(\phi_1), c(\phi_2))) = 2 + 2 \cdot max(c(X\phi_1), c(X\phi_2)) \\ &\text{and} \\ c(t(X\varphi)) &= c(t(X(\phi_1 \wedge \phi_2))) = c(t(X\phi_1 \wedge X\phi_2)) \\ &= c(t(X\phi_1) \wedge t(X\phi_2)) = 1 + max(c(t(X\phi_1)), c(t(X\phi_2))) \end{aligned}$$

By I. H., we can conclude
$2 + 2 \cdot max(c(X\phi_1), c(X\phi_2)) > 1 + max(c(t(X\phi_1)), c(t(X\phi_2)))$, therefore $c(X\varphi) > c(t(X\varphi))$.

Case $\varphi = X\phi$:

$$\begin{gathered} c(X\varphi) = c(XX\phi) = 2 \cdot c(X\phi) \\ \text{and} \\ c(t(X\varphi)) = c(t(XX\phi)) = c(t(Xt(X\phi))) = \Box \end{gathered}$$

By I. H., we know $c(Xt(X\phi)) > \Box$. $c(Xt(X\phi)) = 2 \cdot c(t(X\phi))$. By I. H. we know $2 \cdot c(X\phi) > 2 \cdot c(t(X\phi)) > \Box$, therefore $c(X\varphi) > c(t(X\varphi))$. $\Box$

Appendix C

Proof of lemma 9

Proof. By induction on $c(\varphi)$.

Base case: φ is either β_a or N_{ab}, translation keeps them unchanged and therefore $\vdash \varphi \leftrightarrow t(\varphi)$.

Induction hypothesis: For all φ such that $c(\varphi) < n : \vdash \varphi \leftrightarrow t(\varphi)$.

Induction step: Case for negation and conjunction: straightforward from lemma 8 item 1.

Case XN_{ab}: This case follows from the *Red.Ax.X.N* axiom, item 2 of lemma 8 and the I. H..

Case $X\beta_a$: This case follows from the *Red.Ax.X.*β axiom, item 3 of lemma 8 and the I. H..

Case $X(\varphi \wedge \psi)$: This case follows from the $A8$ axiom, item 4 of lemma 8 and the I. H..

Case $X\neg\varphi$: This case follows from the $A2$ axiom, item 5 of lemma 8 and the I. H..

Case $XX\varphi$: By I. H. we know $\vdash X\varphi \leftrightarrow t(X\varphi)$. Therefore we can replace into $Xt(X\varphi)$ by proposition 2, and it follows from item 6 of lemma 8. $\square$

Appendix D

Dataset parser source code

The external dependencies of this source-code are the command-line manager library "commander" (version 7.2.0) and the functional programming utilities library "lodash" (version 4.17.21).

Listing D.1: Source-code entrypoint file, containing the input parameters parsing logic and output type controller code.

```
const { Command, Option } = require("commander");
const fs = require("fs");

const program = new Command();
program.requiredOption("-f, --file <file>", "Star Wars network file");
program.addOption(new Option("-o, --output <type>", "output type").choices(["smv", "graphviz"]));
program.option("-tn, --threshold_num <number>", "The network adoption threshold numerator");
program.option("-td, --threshold_den <number>", "The network adoption threshold denominator");
program.option("-l, --ltlspec <string>", "LTL specification for the nuXmv model checker");
program.option("-s, --symmetric", "Relationships will be considered symmetric");
program.parse(process.argv);

const { file, threshold_num, threshold_den, output = "smv", ltlspec, symmetric = false } = program.opts();

try {
```

```
  if (fs.existsSync(file)) {
    const network = JSON.parse(fs.readFileSync(file,
       "utf8"));
    switch (output) {
      case "smv":
        if (!threshold_num || !threshold_den) {
          throw new Error("Threshold is required for the
             SMV output");
        }
        const smvOutput = require("./processors/smv");
        const smvOutputStr = smvOutput(network,
           threshold_num, threshold_den, !!symmetric,
           ltlspec);
        console.log(smvOutputStr);
        break;
      case "graphviz":
        const graphvizOutput =
           require("./processors/graphviz");
        const graphvizOutputStr =
           graphvizOutput(network, !!symmetric);
        console.log(graphvizOutputStr);
        break;
      default:
        console.error("Invalid output type");
    }
  } else {
    console.error(`File "${file}" not found`);
  }
} catch (err) {
  console.error(err);
}
```

Listing D.2: Controller code for the SMV output, responsible for the general format of the generated SMV model file. The content of the file is a modal with an initial state and operations describing how the model evolves over time.

```
const _ = require("lodash");
const { initAgents, initSum, nextBehaviors, nextSum } =
   require("./smv/assign");
const { setThreshold, setRelationships } =
```

```
  require("./smv/define");
const { defineBehaviors, defineSum } =
  require("./smv/var");

const setLTLSpecs = (ltlspec) => `LTLSPEC ${ltlspec};`;

const smvOutput = (network, threshold_num,
  threshold_den, isSymmetric, ltlspec) => {
  const { nodes, links } = network;

  const numAgents = nodes.length;

  // Initialize the matrix
  const linksMatrix = _.chunk(
    _.range(numAgents * numAgents).map((i) => 0),
    numAgents
  );

  // Fill the matrix with the weights
  links.forEach(({ source, target, value }) => {
    if (linksMatrix.length > source) {
      if (linksMatrix[source].length > target) {
        linksMatrix[source][target] = value;
        if (isSymmetric) {
          linksMatrix[target][source] = value;
        }
      }
    }
  });

  return `MODULE main
DEFINE
  ${setThreshold(threshold_num, threshold_den)}
  ${setRelationships(linksMatrix)}
VAR
  ${defineBehaviors(numAgents)}
  ${defineSum(numAgents, linksMatrix)}
ASSIGN
  ${initAgents(nodes)}
```

```
  ${initSum(numAgents)}
  ${nextBehaviors(numAgents)}
  ${nextSum(numAgents)}
${ltlspec ? setLTLSpecs(ltlspec) : ""}`;
};

module.exports = smvOutput;
```

Listing D.3: Controller code for the GraphViz output, responsible for the general format of the generated GraphViz file. The content of the file is a representation of the initial state of the model.

```
const _ = require("lodash");
const { behaviorNodes } = require("./graphviz/behavior");
const { relationships } =
   require("./graphviz/relationships");

const graphvizOutput = (network, isSymmetric) => {
  const { nodes, links } = network;

  const numAgents = nodes.length;

  // Initialize the matrix
  const linksMatrix = _.chunk(
    _.range(numAgents * numAgents).map((i) => 0),
    numAgents
  );

  // Fill the matrix with the weights
  links.forEach(({ source, target, value }) => {
    if (linksMatrix.length > source) {
      if (linksMatrix[source].length > target) {
        linksMatrix[source][target] = value;
      }
    }
  });

  return `strict ${isSymmetric ? "graph" : "digraph"} {
    ${behaviorNodes(nodes)}
    ${relationships(nodes, linksMatrix, isSymmetric)}}`;
```

```
};

module.exports = graphvizOutput;
```

Listing D.4: Assigns the initial state and the "next" state following LTL-SN semantics.

```
const _ = require("lodash");

const initAgents = (nodes) =>
  nodes
    .map(
      (node, index) =>
        `init(agents[${index}]) := ${node.behavior ?
           "behavior" : "empty"}; -- Agent
           "${_.snakeCase(node.name)}"`
    )
    .join("\n  ");

const innerSumLine = (curAgentIndex, agentIndexes,
   hasNext) => {
  const line = agentIndexes.map(
    (agentIndexInternal) =>
      `(${hasNext ? "next(" :
         ""}agents[${agentIndexInternal}]${
        hasNext ? ")" : ""
      } = behavior ?
         relationships[${curAgentIndex}][${agentIndexInternal}]
         : 0)`
  );

  return line.join(" + ");
};

const sumLine = (numAgents, type) => {
  const agentIndexes = _.range(0, numAgents);

  const sums = agentIndexes.map((agentIndex) => {
    const line = innerSumLine(agentIndex, agentIndexes,
       type === "next");
```

```
    return `${type}(agents_behavior[${agentIndex}]) :=
       ${line};`;
  });

  return sums.join("\n  ");
};

const initSum = (numAgents) => sumLine(numAgents,
   "init");
const nextSum = (numAgents) => sumLine(numAgents,
   "next");

const innerNextBehaviorsLine = (curAgentIndex,
   agentIndexes) => {
  const line = agentIndexes.map((agentIndexInternal) =>
     `relationships[${curAgentIndex}][${agentIndexInternal}]`);
  return line.join(" + ");
};

const nextBehaviors = (numAgents) => {
  const agentIndexes = _.range(0, numAgents);

  const sums = agentIndexes.map((agentIndex) => {
    const line = innerNextBehaviorsLine(agentIndex,
       agentIndexes);
    return `next(agents[${agentIndex}]) := threshold_den
       * agents_behavior[${agentIndex}] > (${line}) *
       threshold_num ? behavior :
       agents[${agentIndex}];`;
  });

  return sums.join("\n  ");
};

module.exports = { initAgents, initSum, nextBehaviors,
   nextSum };
```

Listing D.5: Defines the adoption threshold and the relationships between agents of the model.

```
const _ = require("lodash");

const setThreshold = (threshold_num, threshold_den) =>
  `threshold_num := ${threshold_num};\n  threshold_den
     := ${threshold_den};`;

const setRelationships = (linksMatrix) => {
  // Build the string
  let result = "relationships := [\n    ";
  result += linksMatrix.map((row) =>
     `[${row.toString()}]`).join(",\n    ") + "];";

  return result;
};

module.exports = { setThreshold, setRelationships };
```

Listing D.6: Defines the block of variables declarations.

```
const _ = require("lodash");
const defineBehaviors = (numAgents) => `agents: array
   0..${numAgents - 1} of {empty, behavior};`;

const defineSum = (numAgents, linksMatrix) => {
  const sumWeights = linksMatrix.map(_.sum);
  const maxNeighborsInfluence = _.max(sumWeights);
  return `agents_behavior: array 0..${numAgents - 1} of
     0..${maxNeighborsInfluence};`;
};

module.exports = { defineBehaviors, defineSum };
```

Listing D.7: Builds and returns the nodes' definitions block

```
const map = require("lodash/fp/map");
const filter = require("lodash/fp/filter");
const flow = require("lodash/fp/flow");
const snakeCase = require("lodash/fp/snakeCase");

const behaviorNodes = (nodes) => {
  const nodesStr = flow(
```

```
    filter((node) => node.behavior),
    map((node) => `${snakeCase(node.name)};`)
  )(nodes);

  return `node [style=filled,color=red];
    ${nodesStr.join("")}`;
};

module.exports = { behaviorNodes };
```

Listing D.8: Builds and returns the nodes' relationships block.

```
const snakeCase = require("lodash/fp/snakeCase");

const relationships = (nodes, links, isSymmetric) => {
  const lines = links.map((source, srcIdx) =>
    source
      .map(
        (value, tgtIdx) =>
          `    ${snakeCase(nodes[srcIdx].name)}
             ${isSymmetric ? "--" : "->"} ${snakeCase(
            nodes[tgtIdx].name
          )} [label=${value}]`
      )
      .filter((line) => !line.includes("[label=0]"))
      .join("\n")
  );

  return `node [style=filled,color=grey];
${lines.filter((line) => line).join("\n")}`;
};

module.exports = { relationships };
```