Synthese (2023) 202:79
https://doi.org/10.1007/s11229-023-04264-6

ORIGINAL RESEARCH

Detecting bots with temporal logic

Mina Young Pedersen[1] · **Marija Slavkovik**[1] · **Sonja Smets**[2]

Received: 28 February 2022 / Accepted: 6 July 2023 / Published online: 28 August 2023

Abstract
Social bots are computer programs that act like human users on social media platforms. Social bot detection is a rapidly growing field dominated by machine learning approaches. In this paper, we propose a complementary method to machine learning by exploring bot detection as a model checking problem. We introduce Temporal Network Logic (TNL) which we use to specify social networks where agents can post and follow each other. Using this logic, we formalize different types of social bot behavior with formulas that are satisfied in a model of a network with bots. We also consider an extension of the logic where we explore the expressive power of including elements from hybrid logic in our framework. We give model checking algorithms for TNL and its hybrid extension, and show that the complexity of the former is in P and the latter in PSPACE.

Keywords Social bots · Bot detection · Model checking · Temporal logic · Social network logic · Hybrid logic

1 Introduction

Software-controlled bots, often called social bots, act like human users on social media: they interact with other users, both humans and other bots; share content; and target users that are likely to believe in misinformation (Shao et al., 2018). Social bots may have beneficial purposes (Gilani et al., 2017), but they can also be used to amplify or

✉ Mina Young Pedersen
mina.pedersen@uib.no

Marija Slavkovik
marija.slavkovik@uib.no

Sonja Smets
s.j.l.smets@uva.nl

1 Department of Information Science and Media Studies, University of Bergen, 5007 Bergen, Norway

2 ILLC, University of Amsterdam, 1098 XG Amsterdam, The Netherlands

direct misinformation. In the worst case, they can be seen as a threat to democracy (Gorodnichenko et al., 2018).

Bot detecting is a rapidly growing field, mainly dominated by literature on algorithms using machine learning techniques (Cresci, 2020). These algorithms try to catch bots by identifying behavior such as aggressive following and unfollowing (Lee et al., 2011), or a "bursty nature": posting a lot in short periods of time followed by long periods of inactivity (Chu et al., 2012). Other algorithms try to detect bots by looking for specific network topologies, such as clusters of bots (Cao et al., 2012), or by considering content of the bots' posts and metadata about them (Kudugunta & Ferrara, 2018).

Machine learning approaches require a large amount of labeled data for training. We are interested in exploring a complementary approach to bot detection that circumvents this problem: Can social bot detection be done as a model checking problem?

Model checking is a method for verifying whether a formally specified model of a system meets a given specification. Among several purposes, it has been proposed as an approach for the analysis of social networks in the context of information and opinion diffusion properties (Belardinelli & Grossi, 2015; Dennis & Slavkovik, 2020; Dennis et al., 2022; Machado & Benevides, 2022) and privacy (Dennis et al., 2017; Pardo & Schneider, 2017). It has also been proposed to identify groups of compromised smartphones, called mobile botnets, in mobile apps (Bernardeschi et al., 2019). In the line of work on formal verification of social networks, we propose to use this method for detecting social bots.

We introduce Temporal Network Logic (TNL) to represent sufficient information about a social network to detect whether the network contains social bots. In this framework, bot behavior is expressed in logical formulas. This behavior is based on existing bot detectors as well as empirical findings on how social bots have acted in online social networks. By representing a social network as a model of TNL and bot behavior as a logical formula, checking whether a specific bot formula is satisfied in a model of TNL checks the presence of that particular bot behavior in the social network. We also present a simple algorithm for building a TNL model from social network data, enabling future work to implement this detection method on a real-life social network.

There are three specific advantages of our bot detection method:

1. **We do not need to train classifiers on big data.** Social bots are constantly evolving (Cresci, 2020). Requiring that each type of a bot needs a classifier to be trained for it, might be an issue because of the amount of training data required. In our approach, we can formalize certain types of new bot behavior as a new formula. Once the specification in TNL of the social network is available, we can check both for properties of agents or for properties of groups of agents in the network.
2. Model checking, unlike some machine learning methods, is **directly inspectable by design**. This detection method is relevant to the current need for responsible algorithms in artificial intelligence research.
3. **Detection of logical inconsistencies in agents' posts is possible.** The logic syntax lets us formalize the content of a post in terms of propositions and logical connectives. In theory, this allows us to, for instance, check for posts that themselves are

logical contradictions. The practical applicability of this option, of course, depends heavily on the availability of natural language processing tools.

Beyond inconsistencies, a symbolic approach allows us to perform a qualitative analysis on chosen segments and time intervals of the social network. Intuitively, we can translate what a human social network user would consider bot behavior into a property that we can formally verify.

Our approach makes it possible to give an exact logical specification of pre-defined mechanisms for bot behavior. This builds a direct connection to studies in social science on information flow in social networks. A limitation of our approach is that it does not yet offer the possibility of real-time bot detection. This work is perhaps best described as forensics: we uncover evidence of bot activity in the network in the past. This, however, is still sufficient to understand the influence that bots have on propagation of information in the network and to use this information to improve social network services to limit the power of bots.

The framework we use is a temporal logic that is also interesting in its own respect. We discuss similarities between TNL and existing logics, such as linear-time temporal logic and tense logic, and show that TNL is sound and weakly complete by combining axioms from existing axiomatic systems. In the latter sections of the paper, we take advantage of the structure of the models of TNL to extend our logical semantics and evaluate formulas not only from the viewpoint of a global timeline, but also for each local agent in the social network. This enables us to further analyze powerful network positions a suspected malicious agent might have. We give model checking algorithms for both TNL and the extended TNL.

The paper is structured as follows. We begin in Sect. 2 by investigating existing bot detectors and empirical research on the behavior of social bots. In Sect. 3, we assess related work. Then, in Sect. 4, we introduce TNL with an example and show some network and agent properties that can be expressed in this language. We also present a sound and weakly complete axiomatization of TNL. In Sect. 5, we give formulas for bot behavior. Section 6 is devoted to a discussion of the complexity and efficiency of model checking for bot detection in TNL. In Sect. 7, we extend TNL to TNL_a to be able to evaluate formulas at agents as well as time points. Here, we also introduce a model checking algorithm for TNL_a. Section 8 proposes a way to build temporal network models from social network data. We end with a conclusion and prospects for future work in Sect. 9.

This paper is an extended version of a paper by Pedersen et al. (2021), first published in the proceedings of the Eight International Workshop on Logic, Rationality and Interaction (LORI 2021). The extension includes the following additions: a review of related work; a sound and weakly complete axiomatization of TNL; a model checking algorithm for TNL; and, the introduction of the logic TNL_a as well as a model checking algorithm for TNL_a.

2 Social bots

To develop a method for bot detection, it is crucial to first analyze how a bot behaves. We do this by considering existing bot detectors and empirical research on social bot behavior.

Misinformation. Misinformation is, according to common dictionaries, known as incorrect or misleading information. Disinformation is often regarded as a subset of misinformation that is deliberately spread to influence public opinion or distort the truth. There is a general agreement in the literature on bots on social media platforms that their malicious behaviors are inherently related to the spread of misinformation such as hoaxes, rumors, conspiracy theories, or fabricated reports (Shao et al., 2018).

Empirical studies conducted by Vosoughi et al. (2018) on Twitter found that false information, and in particular political news, was diffused significantly faster and farther than true information. Two reasons for this tendency are presumed to be the novelty and shock effect of false information (O'Connor & Weatherall, 2019). People are more likely to share novel or surprising information, and since false information is more likely to be novel and surprising than true information, fake news tend to propagate faster through a social network. Although misinformation is linked to bot behavior, Vosoughi et al. (2018) also found that human behavior contributes more to the spread of false news than bots. As was mentioned by one of the reviewers, identifying bots as well as false information in the network can help us understand which agents spread this type of information.

Bursty nature. Analyses of bot behavior on Twitter have revealed specific activity patterns, both posting- and following-activity. In a well-known study, Chu et al. (2012) analyzed more than 40 million tweets from over 500,000 users and found that bots tend to act in a "bursty" nature, posting a lot in a short period of time and have longer periods of inactivity.

One reason for this bursty posting behavior is speculated in Shao et al. (2018) to be that early and aggressive engagement with information, shortly after it has first been published by a low-credibility source, exposes many users to the content and increases the chances that the article might go "viral": spread to large groups of users in a short time. In a similar study, Lee and Kim (2014) confirmed that bot account creation is bursty too: that accounts are created in bulk within a short period of time, just before the accounts start spamming. A likely reason for bursty account creation before spamming is that it allows the bots to create vast amounts of content before being removed.

Hashtag, subgroup and user targeting. Bots often strive to gain visibility in their social networks. One specific method is to target hashtags by posting irrelevant information to divert a discussion (Chang & Iyer, 2012). An example of bots' misuse of Twitter hashtags was seen under the 2017 natural disasters Hurricane Harvey, Irma and Maria, as well as an earthquake that occurred in Mexico. By gathering data from over 1.2 million tweets by more than 770,000 accounts, Khaund et al. (2018) found bots using disaster-related hashtags to promote unrelated political content on topics such as North Korean leader Kim Jong-un and the Black Lives Matter movement. Social bots also posted hoaxes such as "shark swimming on freeway" under the same hashtags.

Another tactic to gain visibility in social networks is to target subgroups. Leading up to the United States presidential election in 2016, certain Twitter accounts posed as being queer-friendly, dog-loving pages, and only after reaching a number of followers started to publish political content (O'Connor & Weatherall, 2019). Although these accounts were not confirmed to be social bots, this is a potential tactic that could be used for manipulation online. Misusing followers' trust by agreeing on particular topics potentially put these pages in a position to convince their audience on political matters.

Aggressive following and unfollowing. One feature social bots have shown to have on social media platforms is aggressive following and unfollowing (Lee et al., 2011). A reason for this behavior is, as in the case of targeting, to gain visibility; most online social media platforms give an alert to their users when they have a new follower, or contact request. In this way, a bot can enhance the probability that users will engage with their posts.

3 Related work

The work in this paper lies in the intersection of several scientific fields. We can find related literature in research on social bot detectors, in work on logical frameworks to model social network dynamics, and in the formal verification community, in particular work on model checking properties in social networks. In this section, we give a review of related work across these three fields.

3.1 Bot detectors

Some of the social bot detectors and their mechanisms to detect bots have been reviewed in the previous Sect. 2. Most bot detecting frameworks technically differ a lot from ours, and few papers combine work on social bot detectors and formal logic. Bernardeschi et al. (2019) propose model checking to identify mobile botnets in the operating system Android. A mobile botnet is a group of compromised smartphones controlled by an owner using software called Command and Control (C&C). The method in the paper uses a branching temporal logic to characterize botnet behavior which is checked against a set of finite state automata representing the run-time behavior of an app on a smartphone. Although a mobile botnet in its nature is quite different from a social bot, we include the paper as related work because the method proposed is similar to ours, and because both social bots and botnets can be used for malicious purposes.

3.2 Logics for social network dynamics

Our work uses a modal logic framework to reason about the dynamics of a social network. In recent years, the field of social network logics, and particularly dynamic social network logics, has seen many influential papers. By a social network, we mean a graph structure in which the nodes represent agents, and the relation between them

represents some kind of friendship relation or communication channel. We take social network logics to be logical frameworks that model this network graph explicitly.[1] Some work in the field is reviewed in a paper by Liu and Li (2022), which focuses on work on social network logics in China.

As one of the first on the topic, Pacuit and Parikh (2005) introduce a logic of communication graphs where edges between agents depict possibilities of direct communication. This logic uses a neighborhood semantics[2] which language includes a knowledge modality and a dynamic diamond. A formula $\Diamond K_i\phi$ is to be read as "after some communications, agent i knows ϕ according to i's current information".

The frameworks (Seligman et al., 2013; Liu et al., 2014) built upon the work first presented by Seligman et al. (2011) have been deeply influential in the field of social network logics. The underlying logical models have a set of agents, a set of epistemic states and two binary relations: one family of friendship relations on the set of agents and one family of epistemic relations on the set of epistemic states. The logical language includes nominals and hybrid operators, which makes it possible to have statements such as "Bella knows that she is not a spy, but doesn't know if a friend of hers is a spy" $@_b(K\neg s \wedge \neg K\langle F\rangle s)$ (Seligman et al., 2013). The frameworks are used to model phenomena such as peer pressure, belief change and epistemic updates. Work by other authors that has continued the technical results of these frameworks include (Christoff et al., 2016; Sano, 2017; Fernández González, 2021; Balbiani & Fernández González, 2020; Zhen, 2020).

Our logic tracks the dynamics of the social network through a temporal logic. Other work in social network logic that uses temporal operators include (Pedersen & Slavkovik, 2017; Machado & Benevides, 2022; Van der Hoek et al., 2020). Of the papers in this section, we have found the frameworks in these papers to be the most similar to ours as they combine social networks and temporal logics. We therefore include a longer description of their logical models. Pedersen and Slavkovik (2017) introduce a logic with next-time as its only temporal operator to model social influence in a network of agents. The logical model is constructed such that each state is a social network $G = (N, \square, pro, E)$ where N is a set of agents, $\square$ is a set of relevant issues, pro is a function deciding for each issue in $\square$ which agents support them, and E is a set of edges between agents. A successor of G is defined to be a new social network $(N, \square, pro, E/A)$ where only the set of edges has possibly been reduced. The motivation behind reducing edges in E is that agents cut links to their neighbors to avoid conflict, which might appear when the agents' neighbors disagree on the same issue.

Machado and Benevides (2022) present a logic called LTL-SN, based on linear-time temporal logic with the operators next-time and until. With until a future operator can also be defined. LTL-SN tracks the changes of a social network similar to the models of diffusion first defined by Baltag et al. (2019), in which agents adopt a behavior in the next time point if a number of their neighbors have the behavior, relative to a given threshold. The changes in the network are therefore decided by the initial model $\mathcal{M} = (\mathcal{A}, N, \theta, I)$ where $\mathcal{A}$ is a set of agents, N a set of relations between agents, θ

[1] We thank Rasmus K. Rendsvig for proposing this definition.

[2] See Pacuit (2017) for further information on neighborhood semantics for modal logic.

a given threshold and $I \subseteq \mathcal{A}$ a set of agents that initially exhibit the behavior. LTL-SN can be reduced to propositional logic, due to the fact that the changes through the temporal operators are decided based on $\mathcal{M}$. The paper also includes a model checking algorithm, and it is shown how their models can be used to represent a known model for epidemics called the Susceptible-Infectious-Recovered (SIR) model.

The work by Van der Hoek et al. (2020) is based on a framework originally presented by van der Hoek et al. (2019); Van der Hoek et al. (2022), where balance games are introduced. Balance games are game theoretical renditions of social situations where relations between groups of three agents are judged to be balanced or not. Given a set of agents A, a relation on A is defined such that each relation is either a friend relation "+" or an enemy relation "−", but not both. For each triad of three agents, the triad is balanced if on the form $+++$ or $--+$ and unbalanced otherwise. When all triads between all agents are balanced, the whole network is said to be balanced. In balance games, a random agent is chosen in each turn to decide whether to change one of its relations for a cost. The agents receive a higher utility when they are in balanced triads. A network is defined to be stable when all agents have more reason for their relationships to stay the same than to change. Van der Hoek et al. (2020) analyze balance games through a temporal logic called Logic of Allies and Enemies (LAE). The language of LAE includes operators $AX\phi$, $A[\phi U \phi]$, and $E[\phi U \phi]$ from Computation Tree Logic (CTL). It is shown that some formulas are valid in the network if and only if, for instance, the network eventually becomes stable. Other works by Xiong (2017), Xiong and Ågotnes (2020), Pedersen et al. (2019) and Pedersen (2019) also use logic to model balance, although neither with a temporal logic nor a game theoretic framework.

The body of literature on logic to model dynamics in social networks also include work by Ruan and Thielscher (2011), Lorini and Sartor (2016), Christoff and Grossi (2017), Grandi et al. (2017), Smets and Velázquez-Quesada (2017a, 2017b, 2020a, 2020b), Xiong et al. (2017), Pardo et al. (2018), Viana et al. (2014), Fernández González (2022), Pedersen et al. (2020, 2021), Baltag et al. (2019), Belardinelli (2019), Perrotin et al. (2019), Yang et al. (2019), Christoff and Hansen (2015), Cristani et al. (2020), Santos (2020), Occhipinti Liberman and Rendsvig (2022) and Galimullin et al. (2022). Although these frameworks have less in common with our work, conceptually they deal with social networks in which relations are formed and lost, and attitudes are adopted. This is akin to our notions of following and posting. Several of the mentioned papers, by Baltag et al. (2019), Pedersen et al. (2019), Smets and Velázquez-Quesada (2017a, 2017b, 2020a, 2020b), Machado and Benevides (2022) and Ruan and Thielscher (2011), use threshold models to model limits to when agents adopt an opinion or behavior, or to when relationships are changed. Threshold models use a threshold representing a given number that measures quantities such as the amount of social connections that need to have an opinion before an agent adopts it, or how much two agents should have in common to be connected.

3.3 Model checking social networks

We also include here some related work which combines model checking and social networks. As mentioned earlier, Machado and Benevides (2022) give a model checking algorithm for the logic LTL-SN which models diffusion in a social network. The work by Belardinelli and Grossi (2015), Dennis and Slavkovik (2020) and Dennis et al. (2022) also explores model checking diffusion properties in social networks. Belardinelli and Grossi (2015) introduce logical models for multi-agent systems called open dynamic agent networks, in which agents can join and leave the network. The logic used is a type of first-order CTL, and it is shown that its model checking problem is decidable. Dennis and Slavkovik (2020) use Markov chains to model information diffusion in social networks, where the internal information state of an agent is explicitly modeled. It is shown that the PRISM probabilistic model checker cannot be used to check even simple models. The work by Dennis and Slavkovik (2020) is extended by Dennis et al. (2022). By supplementing the use of PRISM with Monte Carlo simulation, larger networks can be checked.

Pardo and Schneider (2017) present a formal system to model knowledge of agents in social networks. The social network is not presented as a Kripke model, which has been standard in many works in the social network logic literature. Rather, the models by Pardo and Schneider (2017), called SNMs, are social graphs with a knowledge base for each user in the system. The motivation is to formalize privacy policies for agents in the network, such as "only my friends can know my location". Model checking is proposed to check whether a user knows a given statement. It is shown that model checking formulas over SNMs is decidable. The work by Dennis et al. (2017) is also motivated by formal verification of privacy policies in social networks. More concretely, the proposal is to use model checking to analyze information leakage, which is when information is accessed by other agents who are not authorized to share it. The framework by Dennis et al. (2017) is probabilistic and uses details from BDI, Belief-Desire-Intention, models. As for Dennis and Slavkovik (2020) and Dennis et al. (2022), the probabilistic model checker PRISM is used by Dennis et al. (2017) to check properties in the network.

4 Temporal Network Logic (TNL)

We are now ready to introduce Temporal Network Logic (TNL). TNL is a temporal logic in which each time point hosts a social network. In addition to the standard temporal logic operators past $P\phi$, future $F\phi$ and next-time $X\phi$, the language includes predicates $\langle follow\rangle(a, b)$ and $\langle posted\rangle(a, \omega)$ that let us reason about agents' activities in the social network, as well as the network structure, at a specific time on the timeline.

4.1 Language

We begin by presenting the syntax of TNL.

Definition 1 (*Syntax*) Let At be a finite set of atomic posts and Ag be a finite set of agents. We define the well-formed formulas of the language $\mathcal{L}_{TNL}$ to be generated by the following grammar with the separation property:

$$\phi ::= p \mid \langle follow \rangle(a, b) \mid \langle posted \rangle(a, \omega) \mid \neg\phi \mid \phi \wedge \phi \mid P\phi \mid F\phi \mid X\phi$$
$$\omega ::= p \mid \neg\omega \mid \omega \wedge \omega$$

where $p \in \mathsf{At}$ and $a, b \in \mathsf{Ag}$. We define propositional connectives like $\vee$, $\rightarrow$ and the formulas $\top$, $\bot$ as usual. Further, we define the duals $H := \neg P \neg$ and $G := \neg F \neg$ as standard.

All formulas of $\mathcal{L}_{TNL}$ have the separation property, which means that each formula can be rewritten as a Boolean combination of formulas, each of which depends only on the past, present, or future.[3] We use Ω to denote the set of all possible posts ω, which is the propositional fragment of $\mathcal{L}_{TNL}$. Since the set At is finite, the set Ω is finite up to equivalent formulas.

Intuitively, we read $\langle follow \rangle(a, b)$ as "agent a follows agent b" and $\langle posted \rangle(a, \omega)$ as "agent a posted ω" or "ω appears on agent a's profile". ω is a logical formula made up of atomic posts and standard logical connectives. In $\mathcal{L}_{TNL}$, we therefore have formulas such as $\langle posted \rangle(a, p)$ and $\langle posted \rangle(a, p \rightarrow q)$. We call $p, q, r, \ldots \in \mathsf{At}$ atomic posts. $P\phi$, $F\phi$ and $X\phi$ are temporal operators. We read $P\phi$ as "ϕ was the case at some past time", $F\phi$ as "ϕ will be the case at some future time" and $X\phi$ as "ϕ is the case next step, or next time".

4.2 Models and semantics

Before presenting the semantics, we give definitions of temporal network frames and models.

Definition 2 (*Temporal Network Frame and Model*) We define the following:

- $A \subseteq \mathsf{Ag}$ is a set of agents;
- T is a countably infinite set of time points;
- $<$ is a strict linear order with no end on T, obtaining an infinite chain;
- $N : (A \times T) \rightarrow \mathcal{P}(A)$ is a follower function specifying for each agent their followers at a specific time;
- $V : \mathsf{At} \rightarrow \mathcal{P}(T)$ is a valuation function deciding the truth value of atomic posts at each time point;
- $Posts : (A \times T) \rightarrow \Omega$ is a function outputting for each agent the posts they have on their profile at a specific time.

The tuple $\mathfrak{F} = (A, T, <)$ is a **temporal network frame**, whereas the tuple $\mathfrak{M} = (A, T, <, N, V, Posts)$ is a **temporal network model**. We define a **pointed temporal network model** to be $(\mathfrak{M}, t)$ where $\mathfrak{M}$ is a temporal network model and $t \in T$ its distinguished point, at which the evaluation takes place.

[3] For more information on the separation property, we advice the reader to turn to Hodkinson and Reynolds (2005).

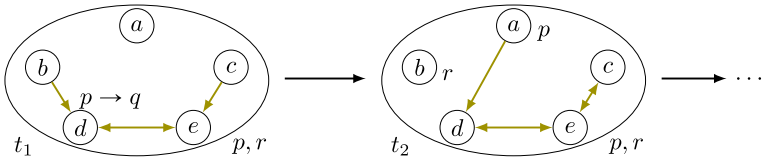

Fig. 1 A temporal network model $\mathfrak{M}$

For a temporal network frame $\mathfrak{F}$ and a formula $\phi \in \mathcal{L}_{TNL}$, we write $\mathfrak{F} \Vdash \phi$ when ϕ is valid in $\mathfrak{F}$: if ϕ is true at every time t in every temporal network model $\langle \mathfrak{F}, N, V, Posts \rangle$ for any follower function N, any valuation V and any post function $Posts$ on $\mathfrak{F}$.

A temporal network model represents the evolution of a social network over time: a timeline in which each time point hosts a social network, defined by the follower function N. Agents' posts at each time is modeled by the post function $Posts$. The intention of N and $Posts$ is that followers continue to follow until they actively unfollow, and that posts stay on an agents' profile until they are actively removed by the posting agent. If a post $\omega \in Posts(a, t)$ for agent a at t, we say that "a posted ω at t" or that "ω appears on agent a's profile at t". This is not intended to mean that ω is first posted at t, ω could have been posted earlier, but still appears on the agent's profile. To avoid confusion, we say "a actively posts ω at t" when $\omega \in Posts(a, t)$ and $\omega \notin Posts(a, t-1)$ for $t-1$ being the immediate predecessor of t.

See Fig. 1 for a simple example of a temporal network model. This network has only five agents $A = \{a, b, c, d, e\}$ and although the timeline has no end, we choose to concentrate on its first time step to the next: $T = \{t_1, t_2, \ldots\}$. Also note that $t_1, t_2 \in V(p)$ and $t_1, t_2 \in V(r)$, whereas $t_1, t_2 \notin V(q)$. Further, $Posts(d, t_1) = \{p \to q\}$, $Posts(a, t_2) = \{p\}$ and $Posts(b, t_2) = \{r\}$. The edges within the time points correspond to the information captured in the follower function: for instance "c follows e" is represented by an edge from c to e.

We now define the semantics of TNL.

Definition 3 (*Semantics*) Let $\mathfrak{M} = (A, T, <, N, V, Posts)$ be a temporal network model, $t \in T$ be a time point, $a, b \in A$ be agents, and $\omega \in \Omega$ be a post. Truth conditions for TNL are defined as follows:

$$
\begin{aligned}
&\mathfrak{M}, t \Vdash p \text{ iff } t \in V(p)\\
&\mathfrak{M}, t \Vdash \langle follow \rangle (a, b) \text{ iff } a \in N(b, t)\\
&\mathfrak{M}, t \Vdash \langle posted \rangle (a, \omega) \text{ iff } \omega \in Posts(a, t)\\
&\mathfrak{M}, t \Vdash \neg\phi \text{ iff } \mathfrak{M}, t \nVdash \phi\\
&\mathfrak{M}, t \Vdash \phi \wedge \psi \text{ iff } \mathfrak{M}, t \Vdash \phi \text{ and } \mathfrak{M}, t \Vdash \psi\\
&\mathfrak{M}, t \Vdash P\phi \text{ iff } \exists s \in T \text{ such that } s < t \text{ and } \mathfrak{M}, s \Vdash \phi\\
&\mathfrak{M}, t \Vdash F\phi \text{ iff } \exists v \in T \text{ such that } t < v \text{ and } \mathfrak{M}, v \Vdash \phi\\
&\mathfrak{M}, t \Vdash X\phi \text{ iff } \mathfrak{M}, t+1 \Vdash \phi,\\
&\qquad \text{where } t+1 \text{ is the immediate successor of } t
\end{aligned}
$$

4.3 Expressing agent properties

An important motivation in the still emerging field of social network logics is to explore what network properties we can formalize with a limited logical syntax. In this section, we present some properties that can be expressed in $\mathcal{L}_{TNL}$.

1. Agent a actively posts ω in the next step: $\neg\langle posted\rangle(a,\omega) \wedge X\langle posted\rangle(a,\omega)$
2. Agent a removes post ω in the next step: $\langle posted\rangle(a,\omega) \wedge X\neg\langle posted\rangle(a,\omega)$
3. Agent b starts to follow agent a back: $P\langle follow\rangle(a,b) \wedge \neg P\langle follow\rangle(b,a) \wedge \langle follow\rangle(b,a)$
4. Agent a posted an original post ω, i.e. they are the first in the network to post ω: $\langle posted\rangle(a,\omega) \wedge \bigwedge_{b\in A} \neg P\langle posted\rangle(b,\omega)$
5. At the current time, agent a is a *lurker*, an agent in the network that only observes (follows at least one other agent), but has not yet posted: $\bigvee_{b\in A} P\langle follow\rangle(a,b) \wedge (\bigwedge_{\omega\in\Omega} \neg P\langle posted\rangle(a,\omega) \wedge \neg\langle posted\rangle(a,\omega))$

We return to the example in Fig. 1 and see the following:

1. $\mathfrak{M}, t_2 \Vdash P\langle follow\rangle(c,e) \wedge \neg P\langle follow\rangle(e,c) \wedge \langle follow\rangle(e,c)$
   In t_2, agent e starts to follow agent c back.
2. $\mathfrak{M}, t_1 \Vdash \langle posted\rangle(d, p\to q) \wedge \neg(p\to q) \wedge X\neg\langle posted\rangle(d, p\to q)$
   Agent d posted false information in t_1, but deleted it in the next step (t_2).
3. $\mathfrak{M}, t_2 \Vdash (\bigvee_{b\in A} \bigvee_{\omega\in\Omega} \neg P(\langle follow\rangle(a,b) \vee \langle follow\rangle(b,a) \vee \langle posted\rangle(a,\omega))) \wedge \langle follow\rangle(a,d)$
   Agent a's account was created in t_2.
4. $\mathfrak{M}, t_1 \Vdash \langle follow\rangle(b,d) \wedge X\neg\langle follow\rangle(b,d)$
   In t_2, agent b unfollows agent d.
5. $\mathfrak{M}, t_2 \Vdash \langle posted\rangle(b,r) \wedge \bigwedge_{a\in A} \neg P\langle posted\rangle(a,r)$
   Agent b posted an original post r in t_2.

Note that we here define "account creation" as the first action, in other words posting or following, of the agent in the network. The intuition is that an agent must follow other agents to view the other agents' content. One could alternatively allow for a different setting in which we expand or shrink the set A of agents itself, which would require an extension of the logical framework.

4.4 Soundness and completeness

TNL is a logic with both future, past and next-time operators, interpreted over a strict linear order with no end. A strict linear order is a binary relation that is transitive, irreflexive and trichotomous, the latter being the property such that for all $s, t \in T$: exactly one of $s < t$, $t < s$ or $t = s$ is true. Irreflexivity entails that we read the $P\phi$ and $F\phi$ modalities as strict future and past which does not include the current time point. The formulas $\langle follow\rangle(a,b)$ and $\langle posted\rangle(a,\omega)$ are original in TNL.

The language of TNL is the basic tense logic language with an additional next-time operator and follow- and posted-formulas. Tense logic, originating from Prior (1957), classically includes a past and future operator, but not next-time. There are known

Table 1 Axiomatization of TNL

1. (CT)	All classical tautologies
2. (K_G)	$\vdash G(\phi \to \psi) \to (G\phi \to G\psi)$
3. (K_H)	$\vdash H(\phi \to \psi) \to (H\phi \to H\psi)$
4. (GP)	$\vdash \phi \to GP\phi$
5. (HF)	$\vdash \phi \to HF\phi$
6. (K_X)	$\vdash X(\phi \to \psi) \to (X\phi \to X\psi)$
7. $(FUNC)$	$\vdash X\neg\phi \leftrightarrow \neg X\phi$
8. (FP_G^+)	$\vdash G\phi \to (X\phi \wedge XG\phi)$
9. (IND^+)	$\vdash G(\phi \to X\phi) \to (X\phi \to G\phi)$
10. (**MP**)	If $\vdash \phi$ and $\vdash \phi \to \psi$ then $\vdash \psi$
11. (**US**)	If $\vdash \phi$ then $\vdash \phi^\sigma$, for σ a substitution
12. $(\mathbf{TG}_G)$	If $\vdash \phi$ then $\vdash G\phi$
13. $(\mathbf{TG}_H)$	If $\vdash \phi$ then $\vdash H\phi$

axiomatic systems for tense logic which are complete with respect to the class of strict linear orders with no end (Blackburn et al., 2001; Venema, 2001).

$\mathcal{L}_{TNL}$ can also be seen as an the language of linear-time temporal logic (LTL) without the until operator and with the additional past operator and follow- and posted-formulas. Additionally, models of LTL are reflexive with respect to the future operator. In one of the seminal works that first introduce LTL, Gabbay et al. (1980) present an axiomatic system called DX which is weakly complete with respect to logical models representing the natural numbers and their natural, and irreflexive, ordering. These models are essentially TNL models as the natural numbers and their ordering are strict linear orders with no end. The axiomatization for TNL is the axioms of DX in addition to axioms representing the relationship between past and future, as well as axioms for the past operators from axiomatizations of tense logic. The axiomatic system for TNL is given in Table 1.

Like DX, the axiomatic system for TNL is weakly complete. TNL is not compact and can therefore not be strongly axiomatized. Consider the set of sentences:

$$\Gamma := \{X^n p \mid n \in \mathbb{N}\} \cup \{\neg Gp\}$$

Γ is the set consisting of $\neg Gp$, Xp, XXp, $XXXp$ and so on. Every finite subset of Γ is satisfiable, because we can always construct a model in which the necessary point on the timeline does not satisfy p. However Γ itself is not satisfiable: if $X^n p$ is satisfied for all $n \in \mathbb{N}$, then Gp is true and consequently $\neg Gp$ is false.

Theorem 1 *TNL is sound and weakly complete with respect to temporal network models.*

Proof The proof closely follows the proof of completeness for DX by Gabbay et al. (1980) with details from standard proofs of completeness for various tense logics (Blackburn et al., 2001; Venema, 2001). The full proof is included in Appendix A. □

5 Detection formulas

Our approach to bot detecting is to define a set of formulas corresponding to particular behaviors of a social bot. Checking satisfiability of these formulas in a temporal network model gives us information on whether we should expect bots in the social network.

As was seen in the empirical research overview in Sect. 2, a recurring feature of bot behavior is to act aggressively, or bursty. To model these traits, we need to define what it means to do something a lot and in a short or long period of time. We decide to rely on an external source of our system: a program or person that in each network and at each time gives us a specific value for a lot, depending on the individual situation in each network, and for each agent. This is implemented in our language as the constant $alot$. The same approach goes for a long period of time denoted with the constant $long$. The constants $alot$ and $long$ can be seen as thresholds. That is, for any specific number x in place of, say, $alot$, it means that $alot$ is *at least* x. Anything that happens more than x times, also happens x times. For a short period of time, we argue that in any case, a single time point is a short period of time. We now present the formulas corresponding to bot behaviors.

5.1 Posting false information

Recall that our models include a valuation function $V : \mathsf{At} \to \mathcal{P}(T)$ with an objective truth value for each atomic post in the network, at each time point. This enables us to capture the existence of misinformation in the network with this simple formula **FI**.[4]

$$\bigvee_{a \in A} \bigvee_{\omega \in \Omega} \langle posted \rangle (a, \omega) \wedge \neg \omega \tag{FI}$$

We cannot claim that there are bots in the network based solely on whether the **FI** formula is forced at a time point in the model, as human behavior has been shown to contribute as much to misinformation spread as bots (Vosoughi et al., 2018). The **FI** formula should therefore be checked together with other bot detecting formulas to see whether it is likely that there are social bots, and false information, in the network.

5.2 Bursty nature

To characterize the bursty nature of a social bot, we want to define a formula that describes agents that post a lot in a short period of time and have longer periods of inactivity. We remind the reader that the intended interpretation of the formula $\langle posted \rangle (a, \omega)$ is that the post ω appears on agent a's profile, and therefore, if $\langle posted \rangle (a, \omega)$ holds in consecutive time points, it is intended to mean that the post stays on the agent's profile. We first define a formula representing agent a being

[4] Since the set At of atomic posts is finite, the set Ω of all posts is finite up to equivalent formulas. As mentioned by one of the reviewers, technically this does not make disjunction or conjunction over Ω finite. For the formula to be finite one needs to define a finite subset of formulas that contain one element of each equivalence class of equivalent formulas, and do disjunction and conjunction over this subset. For brevity, we keep the current notation, but note that the aforementioned method is the correct way.

inactive for a long time in $inactive\_long(a) :=$

$$\bigwedge_{\omega\in\Omega}\bigwedge_{b\in A}((X\langle posted\rangle(a,\omega)\leftrightarrow\langle posted\rangle(a,\omega))\wedge(X\langle follow\rangle$$

$$(a,b)\leftrightarrow\langle follow\rangle(a,b))\wedge\cdots\wedge(X^{long}\langle posted\rangle(a,\omega)\leftrightarrow$$

$$\langle posted\rangle(a,\omega)\wedge X^{long}\langle follow\rangle(a,b)\leftrightarrow\langle follow\rangle(a,b)))$$

X^{long} is the standard notation $X^n := XXX\ldots X_n$, in this case for $n = long$. We read $inactive\_long(a)$ as "for all posts and all agents, in the next steps for a long time, agent a posted something, or followed someone, if and only if they have already done this in the current step". We also abbreviate the formula $active\_post(a,\omega) := \neg\langle posted\rangle(a,\omega)\wedge X\langle posted\rangle(a,\omega)$ to be read as "agent a actively posts ω in the next time point". We now define the formula **BN**.

$$\bigvee_{a\in A}\Big(\bigvee_{\substack{\omega_1,\ldots,\omega_m\in\Omega\\ \forall n\neq k:\omega_n\not\equiv\omega_k}}(\underbrace{active\_post(a,\omega_1)\wedge\cdots\wedge}_{}\underbrace{active\_post(a,\omega_m)}_{alot})\wedge Xinactive\_long(a)\Big) \tag{BN}$$

The notation $\forall n\neq k:\omega_n\not\equiv\omega_k$ means that for any $n\neq k$: ω_n and ω_k are not logically equivalent formulas. The **BN** formula states that "there exists an agent, called a for reference, and there exists m (a lot of) inequivalent posts such that the agents posts all these posts at the next time point, and then they are inactive for a long time in the future".

Recall that we formalized account creation as the first action of an agent in the network in Sect. 4.3. This definition allows us to formalize bursty account creation: that a lot of accounts are created in a short period of time. For simplification, we refer to the creation-formula for an agent a as $created(a) :=$

$$\begin{aligned}&(\bigvee_{b\in A}\bigvee_{\omega\in\Omega}\neg P(\langle follow\rangle(a,b)\vee\langle follow\rangle(b,a)\\&\vee\langle posted\rangle(a,\omega)))\wedge(\bigvee_{b\in A}\bigvee_{\omega\in\Omega}\langle follow\rangle(a,b)\vee\langle follow\rangle(b,a)\\&\vee\langle posted\rangle(a,\omega))\end{aligned}$$

We define bursty account creation with the use of the external classifier for $alot$ in the formula **BAC**.

$$\bigvee_{\substack{a_1,\ldots,a_m\in A\\ \forall n\neq k:a_n\neq a_k}}\underbrace{created(a_1)\wedge\cdots\wedge created(a_m)}_{alot} \tag{BAC}$$

5.3 Targeting

We did not extend the syntax to express and directly identify a hashtag within a post in $\mathcal{L}_{TNL}$. But, we can detect the posting of a lot of posts that relates to a particular atomic post p. If we rely on a person or an external natural language processing algorithm that can recognize a hashtag in a post, we can denote p for a given hashtag. We use the abbreviation $\langle posted\rangle(a, p + alot) :=$

$$\bigvee_{\substack{\omega_1,\ldots,\omega_m\in\Omega \\ \forall n\neq k:\omega_n\not\equiv\omega_k}} \underbrace{\langle posted\rangle(a, p\wedge\omega_1)\wedge\cdots\wedge\langle posted\rangle(a, p\wedge\omega_m)}_{alot}$$

for "agent a posted a lot of formulas on the form $p \wedge \omega$ for a given p" and present the formula **HT**.

$$\bigvee_{a\in A}\langle posted\rangle(a, p + alot) \tag{HT}$$

An observation made by one of the reviewers is that hashtags on social media platforms are usually not true or false. One way to tackle this is to set the truth values of hashtags to always be true. Alternatively, we could add a separate set of atomic posts that are hashtags, which cannot be true or false.

Another property we did not explicitly specify in the language $\mathcal{L}_{TNL}$ is whether a post is relevant or irrelevant to a specific hashtag. An important feature of hashtag targeting is that irrelevant information is posted under a particular hashtag. It is therefore arguable whether **HT** captures hashtag targeting.

Subgroup targeting is characterized by a user posting a lot on one topic related to a subgroup, to then start posting about another irrelevant topic, often political. Although we cannot express relevance, we can formalize an agent posting a lot on one topic p before posting a lot on another topic q. This action is described in the formula **ST**.

$$\bigvee_{a\in A} P(\langle posted\rangle(a, p + alot))\wedge\langle posted\rangle(a, q + alot) \tag{ST}$$

5.4 Aggressive following and unfollowing

We define aggressive following or unfollowing as an action that happens a lot, repeatedly, in a short time span. In this context, that is a combination of the proposition *alot* in a short period of time. For simplification, we abbreviate

$startfollow(a,b) := \neg\langle follow\rangle(a,b) \wedge X\langle follow\rangle(a,b)$ and define the formula **AggU**.

$$\bigvee_{a\in A}\ \bigvee_{\substack{b_1,\ldots,b_m\in A\\ \forall n\neq k: b_n\neq b_k}} \underbrace{startfollow(a,b_1) \wedge F\neg\langle follow\rangle(a,b_1)}$$
$$\underbrace{\wedge\cdots\wedge startfollow(a,b_m) \wedge F\neg\langle follow\rangle(a,b_m)}_{alot} \qquad \textbf{(AggU)}$$

We read the formula as "there exists an agent a, and a lot of other agents $b_1,\ldots,b_m$ who all start being followed by a in the next step and are unfollowed by a in the future".[5]

6 Bot detecting: model checking

We define the problem of bot detecting in a social network as a model checking problem in TNL. Model checking is an automated decision procedure for establishing whether a finite model of a system satisfies a formal specification expressed as a logical formula. Let $\phi \in \mathcal{L}_{TNL}$ be a given formula specifying a property of a bot and let $(\mathfrak{M},t)$ be a given finite pointed temporal network model. The model checking problem is the problem to determine whether ϕ is satisfied in $(\mathfrak{M},t)$.

In this section, we discuss complexity results and the computational efficiency of model checking for bot detecting. We propose two alternative options for model checking. First, we give a model checking algorithm for finite fragments of TNL and show that it runs in polynomial time. Then, we show that the model checking problem for TNL can be translated to the model checking problem for linear-time temporal logic with past (PLTL). The reason for including this translation when we already present a model checking algorithm for TNL is to reason about whether we can use these results to leverage existing model checkers such as SPIN and NuSMV which has native support for linear-time temporal logic.

6.1 Finite fragments

The main proposal in this paper is to use model checking to detect social bots in a network of agents. We envision this to be done for a real-life network by using data from a social network to form a corresponding temporal network model. Further details on how we propose to build TNL models from social network data can be found in Sect. 8.

Recall that TNL models are strict linear orders with no end. When taking existing information about a social network, it is clear that this will always be data represented on a finite timeline. Translating this information to a strict linear order with no end is therefore not entirely straightforward. To bypass this problem, we introduce finite fragments of temporal network models.

[5] This formula can also be expanded such that the agent follows a lot of other agents in some boundedly many steps and then eventually unfollows them.

Definition 4 (*Finite Fragments of Temporal Network Models*) A **finite fragment** of a temporal network model $\mathfrak{M}_f = (A, T, \sim_f, N, V, Posts)$ is a tuple where all elements except $\sim_f$ are defined as in a temporal network model. $\sim_f$ is a binary relation on T which is a finite linear order in which all points are irreflexive except the last reflexive point.

A finite fragment of a temporal network model is essentially a finite strict linear order with a final deadlock state: a last state that loops to itself to end the timeline. This construction is needed to obtain a well-defined semantics for the X operator for all formulas of $\mathcal{L}_{TNL}$ on finite fragments. The semantics for $\mathfrak{M}_f$ is defined as the semantics for TNL on classical temporal network models.

6.2 A model checking algorithm for TNL

The model checking algorithm for finite fragments of TNL is based on the model checking algorithm for Computation Tree Logic (CTL), a known result which can be found in logic textbooks (Huth & Ryan, 2004), and originates from influential work on model checking by Clarke and Emerson (1981). We provide pseudocode for the algorithm below.

Algorithm 1 Function $SAT(\phi)$ determining the set of time points that satisfy ϕ

Input: Finite fragment $\mathfrak{M}_f = (A, T, \sim_f, N, V, Posts)$, a formula $\phi \in \mathcal{L}_{TNL}$
Output: The set sat of all time points in $\mathfrak{M}_f$ where ϕ is satisfied

1: $sat := \emptyset$
2: **case**
3: ϕ is $\bot$: $sat = \emptyset$
4: ϕ is p: $sat = \{t \in T \mid t \in V(p)\}$
5: ϕ is $\langle follow \rangle(a, b)$: $sat = \{t \in T \mid a \in N(b, t)\}$
6: ϕ is $\langle posted \rangle(a, \omega)$: $sat = \{t \in T \mid \omega \in Posts(a, t)\}$
7: ϕ is $\neg\psi$: $sat = T \setminus SAT(\psi)$
8: ϕ is $\psi \wedge \chi$: $sat = SAT(\psi) \cap SAT(\chi)$
9: ϕ is $P\psi$: $sat = SAT_P(\psi)$
10: ϕ is $F\psi$: $sat = SAT_F(\psi)$
11: ϕ is $X\psi$: $sat = SAT_X(\psi)$
12: **end case**
13: **return** sat

Algorithm 1 shows a labeling algorithm that labels the time points of $\mathfrak{M}_f$ in a case analysis. The function goes through the subformulas of ϕ from the shortest, working upwards in increasing order of length. It is recursive on the structure of ϕ (lines 7,8,9,10 and 11), referring to the function $SAT(\phi)$ itself in the cases of negation and conjunction. The algorithm relies on three subroutines SAT_P, SAT_F and SAT_X (lines 8–10) found in Algorithms 8, 9 and 10 in Appendix B. For a formula ϕ, $|\phi|$ denotes the size of ϕ, i.e. the number of distinct subformulas in ϕ.

Proposition 2 *The complexity of the model checking algorithm for finite fragments of TNL, Algorithm 1, is* $\mathcal{O}(|\phi| \times |A| \times |T|)$.

Proof To construct sat, each visit of line 4 checks $|V|$ times, each visit of line 5 checks $|N|$ times and each visit of line 6 checks $|Posts|$ times. Since V depends on T, and N and $Posts$ depend on $A \times T$, the complexity of checking lines 4, 5, 6 is $\mathcal{O}(|A| \times |T|)$. We imagine a worst case scenario where one would need to check every distinct node in the parse tree of ϕ exactly once. The search through the parse tree is depth-first and has complexity $\mathcal{O}(|\phi|)$. The complexity of the whole algorithm is therefore $\mathcal{O}(|\phi| \times |A| \times |T|)$. □

We should note that our function is in fact a global model checking algorithm: the output gives us all time points in which the formula is satisfied. Model checking for a single given state only entails to check whether the state is a member of the output set $SAT(\phi)$.

6.3 From TNL to PLTL

We discuss how we can reduce Temporal Network Logic to a fragment of linear-time temporal logic with past (PLTL) (Schnoebelen, 2002) with respect to model checking. That is, we show that for a pointed TNL model $(\mathfrak{M}, t)$ and a formula ϕ in the language of TNL, there exists a pointed PLTL model $(\mathcal{M}, t)$ and a formula φ in the language of PLTL, such that instead of checking whether $(\mathfrak{M}, t) \Vdash \phi$, we can rather check whether $(\mathcal{M}, t) \Vdash \varphi$. The reason we include a reduction of the model checking problem is to open for the possibility of utilizing existing model checkers for LTL, such as NuSMV (Cimatti et al., 2002), as an alternative to the model checking algorithm for TNL presented in the previous section.

PLTL is classical linear-time temporal logic with a past P and a previous step, or yesterday, X^{-1} operator in addition to the standard future F and next-time X operator. In PLTL, the semantics of $X^{-1}\phi$ is analogous to $X\phi$ in which $X^{-1}\phi$ is forced at the current time point if and only if ϕ holds at the previous time point.

Models of PLTL are tuples $\mathcal{M} = (T, R, V)$, where T and V are as defined in Definition 2 of temporal network models. R is usually not a strict linear order in PLTL, and is defined as having no beginning in addition to having no end. The translation from models of TNL to models of PLTL strips down the TNL model to a 3-tuple. We also include a reflexive deadlock state at the beginning of the timeline, so that X^{-1} will be well-defined.

The reduction from any well-formed formula of TNL to a formula of PLTL is shown in the following translation.

The translation $t : \mathcal{L}_{TNL} \to \mathcal{L}_{PLTL}$ is defined as follows.

$$t(p) = p \qquad t(\neg\phi) = \neg t(\phi)$$
$$t(\phi \wedge \psi) = t(\phi) \wedge t(\psi) \qquad t(P\phi) = X^{-1}Pt(\phi)$$
$$t(F\phi) = XFt(\phi) \qquad t(X\phi) = Xt(\phi)$$
$$t(\langle follow\rangle(a,b)) = follow\_ab \in \mathsf{At} \qquad t(\langle posted\rangle(a,\omega)) = post\_a\omega \in \mathsf{At}$$

The semantics of the P and F operators in TNL are defined as strict past and future, without including the present. Thus, the translation into PLTL, where the standard

semantics of past and future includes the present, requires an additional X^{-1} and X operator in the translation of $P\phi$ and $F\phi$, respectively. The set of agents A is finite and the set of posts Ω is finite up to equivalent formulas. Therefore, the set of formulas on the form $\langle follow\rangle(a, b)$ and $\langle posted\rangle(a, \omega)$ for any a, b, ω is finite too and can be translated into atomic posts in the set At.

It is well known that the model checking problem for LTL is in PSPACE (Sistla & Clarke, 1985) and that adding the past operators to LTL does not increase the expressivity of LTL[6]. Namely the model checking problem for LTL with past is also in PSPACE (Markey, 2004).

Despite being in PSPACE, model checking LTL formulas can be done computationally efficiently in practice (Schnoebelen, 2002) and numerous efficient model checking tools have been developed. It has to be noted that current model checking algorithms are exponential in size of the formula (describing the property). In many of our formulas, the size (number of atoms) depends on the number of agents and number of posts we would like to consider. These are numbers we can control and in most cases we would be looking at a specific small number of posts and a part of the social network. We might also be able to use results on model checking LTL on finite traces (Fionda & Greco, 2016). Experiments are needed to establish true practical efficacy of model checking with PLTL in our examples. It has been shown that for PLTL formulas that satisfy the separation property, the SPIN model checker can be used efficiently (Pradella et al., 2003). The NuSMV model checker (Cimatti et al., 2002) has a native support for PLTL formulas.

7 Extending TNL: from an agent's perspective

We look at an extension of Temporal Network Logic in which we switch our perspective from the global view of a time point to the local view of a singular agent in the social network, within the time point. The advantage of this extended version of TNL is that it increases our expressive power and allows us to express specific properties from each agent's point of view such as "this agent is likely a bot" or "this agent is likely following a bot". The model checking algorithm for TNL given in Sect. 6.2 labels time points t in the model where a given formula is forced. Given a specific agent, it is possible to check at what time points, if at all, this agent exhibited a particular behavior. In this section, we give a model checking algorithm for the extended TNL. This model checker labels pairs (a, t) of agents and time points. Therefore, checking a formula representing a property such as "this agent is likely a bot" outputs not only the time points, but also the agents for which this property is true. Model checking these types of formulas might help identify bots in the network.

[6] By the Gabbay theorem, any PLTL formula can be written as an LTL formula (Gabbay, 1989; Schnoebelen, 2002) however some properties can be more succinctly expressed in PLTL.

7.1 Syntax and semantics

This version of TNL where we evaluate formulas at agents in addition to time points, we name Temporal Network Agent Logic, TNL_a. The language of TNL_a includes elements from hybrid logic (Areces & ten Cate, 2007) such as a separate set Nom of atomic propositions called nominals. In contrast to the set of posts At, an element of Nom can only be true at one agent in a time point, and can therefore be taken to represent the agent's unique name. We also include a set of nominal variables Var.

Definition 5 (*Syntax of TNL_a*) Let At, Nom and Var be sets of propositional atoms, all finite and pairwise disjoint. We define the well-formed formulas of the language $\mathcal{L}_{TNL_a}$ to be generated by the following grammar:

$$\begin{aligned}\phi ::= p \mid s \mid \langle follow\rangle(i,i) \mid \langle posted\rangle(s,\omega) \mid \neg\phi \mid \phi \wedge \phi \mid \Diamond\phi \mid \Diamond^{-1}\phi \\ P\phi \mid F\phi \mid X\phi \mid @_s\phi \mid \downarrow x.\phi \\ \omega ::= p \mid \neg\omega \mid \omega \wedge \omega\end{aligned}$$

where $p \in \mathsf{At}$, $s \in \mathsf{Nom} \cup \mathsf{Var}$ and $i \in \mathsf{Nom}$. We define propositional connectives like $\vee$, $\rightarrow$ and the formulas $\top$, $\bot$ as usual.

The syntax of TNL_a is the syntax of TNL with nominals, the hybrid operators @ and $\downarrow$, and the diamonds $\Diamond$ and $\Diamond^{-1}$. Intuitively, we read $@_s\phi$ as "ϕ is true at the agent who is called s". The other hybrid operator $\downarrow x$ names the current agent x, such that we can speak generally about this agent without knowing the actual nominal that is satisfied there. We can read $\downarrow x.\phi$ intuitively as "the current agent is called some specific nominal such as i, but we refer to it as the unique generic x, and ϕ is true at x". We read $\Diamond\phi$ as "the current agent is being followed by an agent where ϕ holds" and $\Diamond^{-1}\phi$ as "the current agent follows someone where ϕ is true". Additionally, the operators $\langle follow\rangle$ and $\langle posted\rangle$ are now defined with nominals and nominal variables. The reason behind this change is to be able to characterize the necessary axioms for the relationship between the global $\langle follow\rangle$-operator and the new local $\Diamond$ and $\Diamond^{-1}$. We intuitively read $\langle follow\rangle(i,j)$ as "the agent called i follows the agent called j" and $\langle posted\rangle(s,\omega)$ as "the agent called s posted ω" or "ω appears on the agent called s' profile".

The models of Temporal Network Logic already include two distinct binary relations on separate sets: the temporal relation $<$ on the set of time points T and the follower function N on the set of agents A. Yet, so far we have restricted our semantics to evaluate formulas only at time points and not at agents. Taking advantage of this existing model structure, we need to make minimal changes to the temporal network model to be able to evaluate formulas also at agents. The only component that differs is that the valuation function V_a now needs to be accommodated to nominals.

Definition 6 (*Temporal Network Agent Model*) A **temporal network agent model** is a tuple $\mathfrak{M}_a = (A, T, <, N, V_a, Posts)$ where all items except V_a are defined as in the case of a standard temporal network model. V_a is defined as follows:

$$V_a : \mathsf{At} \cup \mathsf{Nom} \to \mathcal{P}(A \times T) \text{ where } \forall a \in A, \forall t \in T : \exists i \in \mathsf{Nom} \text{ such that } (a,t) \in V_a(i)$$
$$\text{and } \forall i \in \mathsf{Nom}, \forall t \in T, \forall a, b \in A : \text{ if } (a,t) \in V_a(i) \text{ and } (b,t) \in V_a(i), \text{ then } a = b.$$

We call the tuple $\mathfrak{F}_a = (A, T, <, N)$ a **temporal network agent frame**.

V_a is a valuation function that takes both atomic propositions and nominals as input. The temporal network agent model is *named*, that is, for any agent in a time point, there is a nominal satisfied there. Furthermore, this nominal is unique: the same nominal cannot be true at two distinct agents in the same time point.

Before presenting the semantics of TNL_a, we introduce the function $g : \mathsf{Var} \to A$ which assigns agents to nominal variables. We define an x-variant of g as $g^x_a(x) = a$ and $g^x_a = g(y)$ for all $y \neq x$. Also, for $s \in \mathsf{Nom} \cup \mathsf{Var}$, let $[s]^{\mathfrak{M}_a,g}$ denote the agent whose name is s. For $i \in \mathsf{Nom}$, $[i]^{\mathfrak{M}_a,g}$ is the unique a in t such that $(a,t) \in V_a(i)$. For $x \in \mathsf{Var}$, $[x]^{\mathfrak{M}_a,g} = g(x)$. We continue to call elements in $\mathsf{At} = \{p, q, \ldots\}$, as well as $\mathsf{Nom} = \{i, j, \ldots\}$ and $\mathsf{Var} = \{x, y, \ldots\}$.

Definition 7 (*Semantics of TNL$_a$*) Let $\mathfrak{M}_a = (A, T, <, N, V_a, Posts)$ be a temporal network agent model, $a \in A$ an agent, $t \in T$ a time point and $g : \mathsf{Var} \to A$ an assignment function. Truth conditions for TNL_a are defined as follows:

$\mathfrak{M}_a, g, a, t \Vdash p$ iff $(a,t) \in V(p)$

$\mathfrak{M}_a, g, a, t \Vdash s$ iff $a = [s]^{\mathfrak{M}_a,g}$ for $s \in \mathsf{Nom} \cup \mathsf{Var}$

$\mathfrak{M}_a, g, a, t \Vdash \langle follow \rangle(i,j)$ iff $[i]^{\mathfrak{M}_a,g} \in N([j]^{\mathfrak{M}_a,g}, t)$

$\mathfrak{M}_a, g, a, t \Vdash \langle posted \rangle(s,\omega)$ iff $\omega \in Posts([s]^{\mathfrak{M}_a,g}, t)$ for $s \in \mathsf{Nom} \cup \mathsf{Var}$

$\mathfrak{M}_a, g, a, t \Vdash \neg\phi$ iff $\mathfrak{M}_a, g, a, t \nVdash \phi$

$\mathfrak{M}_a, g, a, t \Vdash \phi \wedge \psi$ iff $\mathfrak{M}_a, g, a, t \Vdash \phi$ and $\mathfrak{M}_a, g, a, t \Vdash \psi$

$\mathfrak{M}_a, g, a, t \Vdash \Diamond\phi$ iff $\exists b \in A$ such that $b \in N(a,t)$ and $\mathfrak{M}_a, g, b, t \Vdash \phi$

$\mathfrak{M}_a, g, a, t \Vdash \Diamond^{-1}\phi$ iff $\exists c \in A$ such that $a \in N(c,t)$ and $\mathfrak{M}_a, g, c, t \Vdash \phi$

$\mathfrak{M}_a, g, a, t \Vdash P\phi$ iff $\exists s \in T$ such that $s < t$ and $\mathfrak{M}_a, g, a, s \Vdash \phi$

$\mathfrak{M}_a, g, a, t \Vdash F\phi$ iff $\exists v \in T$ such that $t < v$ and $\mathfrak{M}_a, g, a, v \Vdash \phi$

$\mathfrak{M}_a, g, a, t \Vdash X\phi$ iff $\mathfrak{M}_a, g, a, t+1 \Vdash \phi$
where $t+1$ is the immediate successor of t

$\mathfrak{M}_a, g, a, t \Vdash @_s\phi$ iff $\mathfrak{M}_a, g, [s]^{\mathfrak{M}_a,g}, t \vdash \phi$

$\mathfrak{M}_a, g, a, t \Vdash\downarrow x.\phi$ iff $\mathfrak{M}_a, g^x_a, a, t \Vdash \phi$

7.2 Relations between the global and the local view

The extended logic TNL_a naturally has to come with some principles establishing the relationship between the old global and the new local formulas. We discuss a selection of these principles characterized with axioms that must be valid on temporal network agent frames to narrow down our class of frames to the ones we are interested in.

Posting. An immediate concern is to make sure that we cannot have a formula stating that the current agent posted ω, while the agent with the unique name of the current agent did not. That is, we do not allow $\downarrow x.\langle posted\rangle(x, \omega)$ and $\neg\langle posted\rangle(i, \omega)$ both to be true when x and i refer to the same agent. Therefore this axiom must be valid for all $\omega \in \Omega$ and all $i \in \mathsf{Nom}$.[7].

$$@_i \downarrow x.\langle posted\rangle(x, \omega) \leftrightarrow \langle posted\rangle(i, \omega)$$

Following. Another concern occurs in the context of the $\langle follow\rangle$ operator and the $\Diamond$ and $\Diamond^{-1}$ operators: we exclude formulas such that an agent called i follows an agent called j, when $\Diamond i$ is false at the agent called j. We have a similar concern for the other diamond. These axioms for all $i, j \in \mathsf{Nom}$ characterize the properties that we want.

$$\langle follow\rangle(i, j) \leftrightarrow @_j \Diamond i$$
$$\langle follow\rangle(j, i) \leftrightarrow @_j \Diamond^{-1} i$$

Global truth. The aim of extending our framework is to preserve the original timeline in TNL, and to be able to evaluate formulas at agents. Therefore, we need to ensure that atomic posts in At either hold at none or at all agents in the same time point. Since our original interpretation of the atomic posts in $\mathcal{L}_{TNL}$ are true facts at each time point, we do not want it to be the case in our new model that a fact is true for some agent, but not for another. This property can be characterized with the following axiom for all $p \in \mathsf{At}$ and all $i \in \mathsf{Nom}$.

$$p \rightarrow @_i p$$

Names. Another important feature we want to preserve from the original timeline is to characterize an agent unfollowing another with a formula such as $\langle follow\rangle(i, j) \wedge X\neg\langle follow\rangle(i, j)$. The X operator switches the evaluation of $\neg\langle follow\rangle(i, j)$ to the next time step, and so it must be crucial to us that the names i and j refer to the same agents for all time steps. If not, we cannot guarantee that it is the same agent called i who unfollows the same agent called j as in $\langle follow\rangle(i, j)$ in the previous time point. The property that all agents have the same name in all time points can be characterized with this simple axiom for all $i \in \mathsf{Nom}$.

$$i \rightarrow Gi$$

[7] Although we will not include a complete axiomatic system for TNL_a in this paper, it is likely that an axiomatization would include standard hybrid axioms for the hybrid logic $\mathbf{K}_{\mathcal{H}(@,\downarrow)}$ (Blackburn & ten Cate, 2006) The axiom presented here also follows from the (DA)-axiom in this known axiomatic system.

We conjecture that TNL_a is complete, but leave a complete axiomatic system for TNL_a to future work. More specifically, we conjecture completeness both with respect to temporal network agent frames, and also the subclass of frames that have the desirable relationships between global and local, with the use of the previously mentioned axioms. The reason for this assumption is, mainly, the observation that our frames are *indexed* (Balbiani & Fernández González, 2020). Indexed frames are relational structures (W^1, W^2, R^1, R^2) with two sets of worlds and two binary relations such that R^1 is a family of relations $\{R^1_w : w \in W_1\}$ on W_2 and R^2 is a family of relations $\{R^2_w : w \in W_2\}$ on W_1. The frames of TNL_a are essentially such structures: the following-relation N is a family of relations on the set of agents A, indexed for each time point in T. The temporal relation $<$ is a relation on the time points, and could be formalized as being a family of relations indexed by each agent in A, although for all agents a, $<_a$ would refer to the same set. Completeness results for logics on indexed frames, also for hybrid logics, have been extensively researched by Balbiani and Fernández González (2020, 2021) and Fernández González (2021), and will provide valuable input for our work on TNL_a.

7.3 Formulas in TNL_a

In this section we present some of the formulas in TNL_a related to the existence of social bots in a network and the powerful positions they may hold. With the use of the local diamond modalities and the hybrid operators, we can begin to analyze properties that, by zooming in on a specific agent, holds either of *me*, or the agents I am following or being followed by.

1. I posted p for the first time now:

$$\downarrow x.\langle posted\rangle(x, p) \wedge \neg P\langle posted\rangle(x, p)$$

2. Someone who follows me now, has posted false information in the past:

$$\Diamond P \bigvee_{\omega\in\Omega} \downarrow x.\langle posted\rangle(x, \omega) \wedge \neg\omega$$

3. I am following someone now who will have a bursty nature in the future (and therefore likely is a bot):

$$\Diamond^{-1} F \downarrow x.\Big(\bigvee_{\substack{\omega_1,\ldots,\omega_m\in\Omega \\ \forall n\neq k:\omega_n\neq\omega_k}} \underbrace{(\langle posted\rangle(x, \omega_1) \wedge \cdots \wedge \langle posted\rangle(x, \omega_m))}_{alot} \wedge inactive\_long(x)\Big)$$

It is interesting to note that the hybrid language lets us analyze particular properties of the network structure, and likely powerful positions that agents inhabit. One such position is called a *local gatekeeper* (Easley & Kleinberg, 2010). In our context, an agent a is a local gatekeeper if and only if there are two other agents, called b and

c, who both follow and are being followed by a, but does not have any follower-relationship with each other. The reason a is called a local gatekeeper between b and c is that it is likely that information from b to c, or vice versa, would go through a. It is therefore reasonable that we would be interested in whether or not a is a trustworthy agent. With the language of TNL_a we can formalize properties such as the following.

4. I am following someone now who posted false information in the past, and is a local gatekeeper between me and another agent:

$$\downarrow x.\Diamond^{-1} \downarrow y.(\neg x \wedge (P \bigvee_{\omega \in \Omega} \langle posted \rangle (y, \omega) \wedge \neg \omega)$$
$$\wedge \Diamond^{-1} x \wedge \Diamond^{-1} \downarrow z.(\neg x \wedge \neg y \wedge \Diamond^{-1} y \wedge \neg \Diamond x \wedge @_x \neg \Diamond z))$$

Another property we can define with formulas in $\mathcal{L}_{TNL_a}$ is a least number of followers for the current agent or any of their relations. When analyzing the network, knowing the number of followers is crucial to be able to discuss a particular agent's power to be heard.

5. I am being followed by someone who has at least three followers:

$$\Diamond \downarrow x.\Diamond \downarrow y.(\neg x \wedge @_x \Diamond \downarrow z.(\neg x \wedge \neg y \wedge @_x \Diamond (\neg x \wedge \neg y \wedge \neg z)))$$

The list of sentences we have included is only a minor selection of possible properties we could have mentioned. The aim is rather to show some examples of properties we can express in TNL_a which we could use a model checker to check for in the network.

7.4 Model checking TNL_a

We introduce a model checking algorithm for TNL_a. As in the case of TNL, we start by assuming that we have gathered data from a real-life social network and can use this to develop, in this case, a finite fragment of a TNL_a model. The TNL_a model is, as the TNL model, an infinite timeline whereas gathered social network data will always be finite. To depict this finite information as a TNL_a model, we therefore define an equivalent to the finite fragments of TNL.

Definition 8 (*Finite Fragments of Temporal Network Agent Models*) A **finite fragment** of a temporal network agent model $\mathfrak{M}_{f_a} = (A, T, \sim_f, N, V_a, Posts)$ is a tuple where all elements except $\sim_f$ are defined as in a temporal network agent model. $\sim_f$ is a binary relation on T which is a finite linear order in which all points are irreflexive except the last reflexive point.

Finite fragments of temporal network agent models are finite timelines with social network information, in which the last irreflexive point is a reflexive deadlock point. The reason for this final deadlock point is to keep the semantics of the X operator well-defined. The semantics for formulas in $\mathcal{L}_{TNL_a}$ on finite fragments are defined as in the case of temporal network agent models.

The model checking algorithm for TNL_a builds on the model checking algorithm MCFULL for hybrid logic given by Franceschet and de Rijke (2006), with some substantive differences accommodated for our setting. Classical hybrid logic models only have one relational structure, as opposed to the two relations $\sim_f$ and N in finite fragments of temporal network agent models. This difference is also reflected in $\mathcal{L}_{TNL_a}$, which is richer than the tense hybrid logic language given by Franceschet and de Rijke (2006) and includes both $X\phi$, $\Diamond\phi$, $\Diamond^{-1}\phi$ as well as the $\langle follow\rangle$- and $\langle posted\rangle$-formulas.

For the standard hybrid language without the binder $\downarrow$, Franceschet and de Rijke (2006) introduce a model checker MCLITE that takes a finite hybrid model $\mathcal{M} = \langle M, R, V\rangle$, an assignment function g and a formula ϕ as input. After termination of the procedure, each state in the model is labeled with all the subformulas of ϕ forced at that state. Its algorithm updates a table whose elements are bits. We also begin by defining a model checker for the language of TNL_a without $\downarrow$, from here on called $\mathcal{L}_{TNL_{a@}}$. Our procedure is based on a similar method, but the table of bits needs to be three-dimensional as opposed to the two-dimensional table of MCLITE. The subroutines introduced in the following also all include details that are uniquely defined for TNL_a.

The model checker $\text{MC}_@$ receives a finite fragment $\mathfrak{M}_{fa} = (A, T, \sim_f, N, V_a, Posts)$, an assignment function g and a formula ϕ in $\mathcal{L}_{TNL_{a@}}$. It outputs the finite fragment where all states are labeled by the subformulas of ϕ which hold at the respective states. Like the model checking algorithm MCLITE, $\text{MC}_@$ goes through the subformulas of ϕ starting with the shortest and working upwards in increasing order of length. Boolean connectives are handled in a standard way.

$\text{MC}_@$ updates a table L of bits of size $|\phi| \times |A| \times |T|$. Note that the size of L is finite, as both A and T are defined as finite in $\mathfrak{M}_{fa}$. We denote $sub(\phi) = \{\alpha, \beta, \ldots\}$ for the set of subformulas of ϕ. Initially, for each element in the table $L(\alpha, a, t) = 1$ if and only if:

- α is an atomic post in $sub(\phi)$ such that $(a, t) \in V_a(\alpha)$;
- α is a nominal in $sub(\phi)$ such that $a = [i]^{\mathfrak{M}_{fa},g}$;
- α is a formula $\langle follow\rangle(i, j)$ in $sub(\phi)$ such that $[i]^{\mathfrak{M}_{fa},g} \in N([j]^{\mathfrak{M}_{fa},g}, t)$; or,
- α is a formula $\langle posted\rangle(s, \omega)$ in $sub(\phi)$ such that $\omega \in Posts([s]^{\mathfrak{M}_{fa},g}, t)$.

When $\text{MC}_@$ terminates, $L(\alpha, a, t) = 1$ if and only if $\mathfrak{M}_{fa}, g, a, t \Vdash \alpha$ for all $a \in A$, $t \in T$ and $\alpha \in sub(\phi)$.

For formulas on the form $\Diamond\alpha$, $\Diamond^{-1}\alpha$, $P\alpha$, $F\alpha$, $X\alpha$ and $@_s\alpha$, $\text{MC}_@$ relies on subroutines $mc_\Diamond$, $mc_{\Diamond^{-1}}$, mc_P, mc_F, mc_X and $mc_@$, respectively. We give the subroutines $mc_\Diamond$, mc_F and $mc_@$, and explain how the other routines can be defined. For subroutine $mc_\Diamond$ in Algorithm 2, define the set N^{-1} such that for any $a, b \in A$ and $t \in T$: $b \in N(a, t)$ iff $a \in N^{-1}(b, t)$. Also define $L(\alpha)$ as the set of pairs (a, t) in the table L where $L(\alpha, a, t) = 1$.

The procedure $mc_{\Diamond^{-1}}$ can be defined similarly as in the case of $mc_\Diamond$, where $N^{-1}(a, t)$ is exchanged with $N(a, t)$ and line 3 states $L(\Diamond^{-1}\alpha, b, t) \leftarrow 1$ instead. For subroutine mc_F in Algorithm 3 define the set $< (t) = \{s \in T \mid s < t\}$.

The procedures mc_P and mc_X can be defined similarly as mc_F. In mc_P, $< (t)$ should be exchanged with the set $> (t) = \{v \in T \mid t < v\}$ and line 3 should rather state $L(P\alpha, a, s) \leftarrow 1$. In the case of mc_X, instead of $< (t)$, define the set $<^1 (t)$ to be

Algorithm 2 Procedure $mc_{\Diamond}$ $(\mathfrak{M}_{f_a}, g, \alpha)$

1: **for** $(a, t) \in L(\alpha)$ **do**
2: **for** $b \in N^{-1}(a, t)$ **do**
3: $L(\Diamond\alpha, b, t) \leftarrow 1$
4: **end for**
5: **end for**

Algorithm 3 Procedure mc_F $(\mathfrak{M}_{f_a}, g, \alpha)$

1: **for** $(a, t) \in L(\alpha)$ **do**
2: **for** $s \in <(t)$ **do**
3: $L(F\alpha, a, s) \leftarrow 1$
4: **end for**
5: **end for**

the set of states reachable one step in the past, in other words the set of the immediate predecessors of t. Recall that this set would be defined as $pre(\{t\})$ using the notation from the model checking algorithm of TNL in Sect. 6. In mc_X, line 3 should state $L(X\alpha, a, s) \leftarrow 1$ instead of $L(F\alpha, a, s) \leftarrow 1$.

Algorithm 4 Procedure $mc_{@}(\mathfrak{M}_{f_a}, g, s, \alpha)$

1: **for** $t \in T$ **do**
2: **let** $\{a\} = [s]^{\mathfrak{M}_{f_a}, g}$
3: **if** $L(\alpha, a, t) = 1$ **then**
4: **for** $b \in A$ **do**
5: $L(@_s\alpha, b, t) \leftarrow 1$
6: **end for**
7: **end if**
8: **end for**

Proposition 3 *Let $\mathfrak{M}_{f_a} = (A, T, \sim_f, N, V_a, Posts)$ be a finite fragment of TNL_a, g an assignment function and ϕ a formula in $\mathcal{L}_{TNL_{a@}}$, the language of TNL_a without the binder $\downarrow$. The model checker $MC_{@}(\mathcal{M}_{f_a}, g, \phi)$ terminates in time $\mathcal{O}(|A|^2 \times |T|^2 \times | \sim_f | \times |\phi|)$.*

Proof The proof follows the similar proof of the complexity of MCLITE, Theorem 4.3 (Franceschet & de Rijke, 2006). $MC_{@}$ checks all subformulas of ϕ, in total $|sub(\phi)| = |\phi|$ formulas. To determine the initial state of the table, one needs to check $|V_a|$ times if α is an atomic post or a nominal, $|N|$ times if α is on the form $\langle follow\rangle(i, j)$ and $|Posts|$ times if α is on the form $\langle posted\rangle(s, \omega)$. Since V_a, N and $Posts$ depend on $A \times T$, the complexity of determining the initial state is $\mathcal{O}(|A| \times |T|)$. The complexity of checking each subformula α depends on the main operator in α. Recall that $\sim_f$ is the binary relation in $\mathfrak{M}_{f_a}$ which is a finite linear order in which all points are irreflexive except the last reflexive point. The cardinality $| \sim_f |$ is the number of elements in $\sim_f$, i.e. the number of transitions from a time point to another in $\mathfrak{M}_f$, plus 1 for the last reflexive point. Subroutine $mc_{\Diamond}$ is the routine with the highest complexity to check, and is checked in time $\mathcal{O}(|A| \times |T| \times | \sim_f |)$. Therefore, $MC_{@}(\mathcal{M}_{f_a}, g, \phi)$ terminates in time $\mathcal{O}(|A|^2 \times |T|^2 \times | \sim_f | \times |\phi|)$. □

We introduce the model checker $\text{MC}_{\downarrow}$ for the full language of TNL_a. $\text{MC}_{\downarrow}$ uses the same combination of bottom-up and top-down strategy as the model checker MCFULL (Franceschet & de Rijke, 2006). It is a recursive model checker, which, as $\text{MC}_{@}$, updates a three-dimensional table L of bits. For each formula ϕ, all subformulas where the main operator is not the hybrid binder $\downarrow$ is checked using the subroutines from $\text{MC}_{@}$. If the main operator is the hybrid binder $\downarrow$, the formula is on the form $\downarrow x.\alpha$. First, for each time point t and for each agent a, we assign a to x. Then, we check α at t and a, using the subroutines from $\text{MC}_{@}$, with the new assignment for x. $\text{MC}_{\downarrow}$ uses the same procedures $check_*$ for $* \in \{\Diamond, \Diamond^{-1}, P, F, X, @\}$ as in MCFULL, shown in Algorithm 5.

Algorithm 5 Procedure $check_* (\mathfrak{M}_{f_a}, g, \alpha)$

1: $\text{MC}_{\downarrow}(\mathfrak{M}_{f_a}, g, \alpha)$
2: $mc_*(\mathfrak{M}_{f_a}, g, \alpha)$

The procedure $check_{\downarrow}$ is shown in Algorithm 6. $check_{\downarrow}$ uses a new subroutine $clear(L, x, t)$ which resets all values of $L(\alpha)$, where x is free in α, in t. As in $\text{MC}_{@}$, Boolean operators are treated as usual.

Algorithm 6 Procedure $check_{\downarrow}(\mathfrak{M}_{f_a}, g, x, \alpha)$

1: **for** $t \in T$ **do**
2: **for** $a \in A$ **do**
3: $g(x) \leftarrow a$
4: $\text{MC}_{\downarrow}(\mathfrak{M}_{f_a}, g, \alpha)$
5: **if** $(a, t) \in L(\alpha)$ **then**
6: $L(\downarrow x.\alpha, a, t) \leftarrow 1$
7: **end if**
8: $clear(L, x, t)$
9: **end for**
10: **end for**

Proposition 4 *Let $\mathfrak{M}_{f_a} = (A, T, \sim_f, N, V_a, Posts)$ be a finite fragment of TNL_a, g an assignment function and ϕ a formula in $\mathcal{L}_{TNL_a}$. Let $r_{\downarrow}$ be the nesting degree of $\downarrow$. The model checker $MC_{\downarrow}(\mathcal{M}_{f_a}, g, \phi)$ terminates in time $\mathcal{O}(|A|^{2+r_{\downarrow}} \times |T|^{2+r_{\downarrow}} \times | \sim_f | \times |\phi|)$ and uses polynomial space.*

Proof The proof follows closely the proof of the complexity of MCFULL, Theorem 4.5 (Franceschet & de Rijke, 2006). The procedure $check_*$ runs in time $C_\alpha + C_*$ where C_α is the cost of checking α and C_* is the cost of the subroutine mc_* for each $* \in \{\Diamond, \Diamond^{-1}, P, F, X, @\}$. The procedure $check_{\downarrow}$ runs in time $|A| \times |T| \times C_\alpha$. Let $C_{MC_@}$ denote the complexity of $\text{MC}_{@}$. The worst case time complexity is $\mathcal{O}(C_{MC_@} \times (|A| \times |T|)^{r_{\downarrow}}) = \mathcal{O}(|\phi| \times | \sim_f | \times (|A| \times |T|)^{2+r_{\downarrow}})$. As in the case of MCFULL, the height of the recursion stack for $\text{MC}_{\downarrow}$ is at most the length of the formula ϕ, thus $\text{MC}_{\downarrow}$ uses polynomial space. □

8 From real-life networks to TNL models

In this section, we propose a method to build temporal network logic specifications of social networks. The motivation is to take a set of snapshots of an existing social network, and translate this into a TNL model. With this model, we can check our bot detecting formulas to see if it is likely that there are bots in the network. The method relies on some initial assumptions about the network which we discuss before introducing an algorithm that renders snapshots of a network into a temporal network model.

8.1 Assumptions

Information about social networks can either be gathered directly or acquired as a dataset with already existing social network data. To be able to build a temporal network model from social network data, we make some assumptions about the network which we want to gather the data from.

1. *The social network is a set of agents that can post and have a relation to each other*. We do not put restrictions on how posting is implemented, or whether the relation is an asymmetric follower-relation or a symmetric friend-relation. A symmetric friendship in the real-life network can be translated into mutual followership in the TNL model.
2. *Time transition is based on a given interval*. We define transition of the network from one time state to another based on a set time interval, such as one second or twenty-four hours. Input of the algorithm is therefore a set of ordered snapshots of the social network. We define a snapshot to be a dataset containing information about the agents, their posts and relations at given moments in time.
3. *External natural language processing algorithm*. To represent natural language posts as logical ω-formulas, we need to rely on human analysis, or an external program which can translate natural language into logical formulas with Boolean operators.
4. *External fact-checking algorithm*. The set of true propositions at every time step, determined by the valuation function V in a temporal network model, needs to be established by an external fact-checking algorithm. Work towards this end has been explored in the literature, as for instance shown by Thorne and Vlachos (2018). In this context, such an algorithm would have to work in collaboration with the natural language processing algorithm mentioned above.

8.2 Building a TNL model

The translation algorithm, shown in Algorithm 7, takes as input ordered snapshots of a social network, as well as the valuation function V from an external fact-checking program, and outputs a finite fragment of a temporal network model $\mathfrak{M}_f = (A, T, \sim_f$

, N, V, $Posts$). Note that the output here is a finite fragment of TNL, and not a finite fragment of TNL$_a$. To build a finite fragment of TNL$_a$, the algorithm would take a nominal valuation V_a instead of V as part of the input. The rest of the algorithm would remain the same.

T is a set of natural numbers of the same size as the set of snapshots. Recall that a snapshot is a dataset that contains information about agents in the network, their posts and relations at given moments in time. For each element t in T, the ordered pair of $(t, t+1)$ is added to $\sim_f$, to construct a strict linear order (line 3). From each snapshot, every agent that is a node in the social network snapshot is added to A (line 5), relations are added to N (line 7) and posts are added to $Posts$ (line 10). Then the reflexive deadlock state is added (line 16) and the program returns the model (line 17). The algorithm runs in linear time on the number of agents, connections and posts in the social network.

Algorithm 7 Building a finite fragment of a temporal network model from snapshots of a social network

Input: Ordered snapshots of a social network, enumerated from 1 to n, V
Parameter: Set of natural numbers $T = \{1, \ldots, n\}$ of size n with its standard ordering, each element referring to a snapshot
Output: $\mathfrak{M}_f = (A, T, \sim_f, N, V, Posts)$

1: $t = 1$
2: **for** $t \in T$ **do**
3: **if** $t \neq n$ **then**
4: add $(t, t+1)$ to $\sim_f$
5: **end if**
6: **if** a is a node in snapshot of network **then**
7: add a to A
8: **if** a follows node b **then**
9: add a to $N(b, t)$
10: **end if**
11: **if** post ω is on a's profile **then**
12: add ω to $Posts(a, t)$
13: **end if**
14: **end if**
15: **end for**
16: Add $n+1$ to T and add $(n+1, n+1)$ to $\sim_f$.
17: **return** $(A, T, \sim_f, N, V, Posts)$

9 Conclusion and future directions

In this paper, we explored social bot detection as a model checking problem. We represented social networks as logical models and characterized bot behavior as formulas; thus, checking whether the formula is satisfied in the model detects whether to expect bots in the network. We first presented examples of bot behavior based on empirical literature and existing bot detectors. Then, we reviewed related work before we introduced Temporal Network Logic and showed some of the network and agent

properties we can formalize in the language. We also showed that TNL is sound and weakly complete. Then, we presented bot behavior formulas corresponding to the earlier assessed properties of social bots. We gave a model checking algorithm for finite fragments of TNL models, and showed that it runs in polynomial time. We also showed the reduction of the model checking problem for TNL to the model checking problem for past linear-time temporal logic and discussed the complexity of model checking for bot detecting through this translation. We then presented an extension of TNL, named TNL_a, where we explored the expressive power of adding hybrid elements to the language and evaluating formulas not only at time points, but also at agents in the network. We introduced a model checking algorithm for finite fragments of TNL_a and showed that its complexity is in PSPACE. The paper ended by a proposal for a way to build finite fragments of TNL models from real social network data.

Completeness for TNL_a is still an open problem. In future work, we want to use the axiomatic system for TNL as well as literature on indexed frames (Balbiani & Fernández González, 2020, 2021; Fernández González, 2021) to provide a complete axiomatization of TNL_a.

Temporal network models keep the whole social network through the entire timeline. As was mentioned by one of the reviewers, an alternative to TNL models is to rather have a framework that defines an initial model and only track the necessary changes. We believe this could be done in a Dynamic Epistemic Logic-style setting by using event models to track changes from a given outset. It will be interesting to study the model checking problem for the language of TNL on this type of models. A proper exploration of this idea is left to future work.

We also hope to be able to empirically evaluate our detection method, and check its efficiency vis-a-vis standard machine learning methods. The verification can be done in at least one of two ways: either by implementing the model checking algorithms for finite fragments of TNL and TNL_a models, or by using an LTL model checker such as SPIN (Holzmann, 1997), LTSmin (Kant et al., 2015) or NuSMV (Cimatti et al., 2002). One of the challenges is to substitute human input for the natural language and fact checking tasks.

Acknowledgements We would like to thank the anonymous reviewers for insightful comments and useful suggestions for improvements. This also includes the reviewers of the conference proceedings paper which this paper is an extension of. A big thank you to Rustam Galimullin, Valentin Goranko and Thomas Ågotnes for discussion and expertise.

Funding Open access funding provided by University of Bergen (incl Haukeland University Hospital). Funding was provided by L. Meltzers Høyskolefond.

Declarations

Conflict of interest There is no conflict of interest to declare.

Research invloving human and animal rights The research did not involve human and animal participants.

Appendix A: Proof of Theorem 1

See Table 2.

Table 2 Axiomatization of TNL

1. (CT)	All classical tautologies
2. (K_G)	$\vdash G(\phi \to \psi) \to (G\phi \to G\psi)$
3. (K_H)	$\vdash H(\phi \to \psi) \to (H\phi \to H\psi)$
4. (GP)	$\vdash \phi \to GP\phi$
5. (HF)	$\vdash \phi \to HF\phi$
6. (K_X)	$\vdash X(\phi \to \psi) \to (X\phi \to X\psi)$
7. $(FUNC)$	$\vdash X\neg\phi \leftrightarrow \neg X\phi$
8. (FP_G^+)	$\vdash G\phi \to (X\phi \wedge XG\phi)$
9. (IND^+)	$\vdash G(\phi \to X\phi) \to (X\phi \to G\phi)$
10. **(MP)**	If $\vdash \phi$ and $\vdash \phi \to \psi$ then $\vdash \psi$
11. **(US)**	If $\vdash \phi$ then $\vdash \phi^\sigma$, for σ a substitution
12. $(\mathbf{TG}_G)$	If $\vdash \phi$ then $\vdash G\phi$
13. $(\mathbf{TG}_H)$	If $\vdash \phi$ then $\vdash H\phi$

Theorem 1 *TNL is sound and weakly complete with respect to temporal network models.*

Proof The proof closely follows the proof of completeness for DX by Gabbay et al. (1980) with details from standard proofs of completeness for various tense logics (Blackburn et al., 2001; Venema, 2001). Soundness is established by showing validity of the axioms, and that the rules preserve validity. We omit details on proof of soundness.

A classic proposition in modal logic states that a logic L is weakly complete with respect to a class of frames $\mathfrak{F}$ if and only if every L-consistent formula is satisfiable on a frame $\mathcal{F} \in \mathfrak{F}$ (Blackburn et al., 2001). Let ϕ be an arbitrary consistent formula. We want to show that ϕ is satisfiable on some TNL frame. We define the canonical model $\mathfrak{M}^{TNL} = (W^{TNL}, R_G^{TNL}, R_H^{TNL}, R_X^{TNL}, V^{TNL})$ where:

- W^{TNL} is the set of all maximal consistent sets of formulas in TNL;
- R_G^{TNL} is a binary relation on W^{TNL} such that $(\Delta, \Theta) \in R_G^{TNL}$ iff for all ψ : $G\psi \in \Delta$ implies $\psi \in \Theta$;
- R_H^{TNL} is a binary relation on W^{TNL} such that $(\Delta, \Theta) \in R_H^{TNL}$ iff for all ψ : $H\psi \in \Theta$ implies $\psi \in \Delta$;
- R_X^{TNL} is a function from W^{TNL} to the power set of $\mathcal{L}_{TNL}$ defined such that $R_X^{TNL}(\Delta) = \{\psi \mid X\psi \in \Delta\}$;
- V^{TNL} is the valuation defined by
  - $V^{TNL}(p) = \{\Delta \in W^{TNL} \mid p \in \Delta\}$,
  - $V^{TNL}(\langle follow\rangle(a, b)) = \{\Delta \in W^{TNL} \mid \langle follow\rangle(a, b) \in \Delta\}$,
  - $V^{TNL}(\langle posted\rangle(a, \omega)) = \{\Delta \in W^{TNL} \mid \langle posted\rangle(a, \omega) \in \Delta\}$.

For simplicity, we write $\Delta < \Theta$ if $(\Delta, \Theta) \in R_G^{TNL}$ and $\Delta > \Theta$ if $(\Delta, \Theta) \in R_H^{TNL}$. $\mathfrak{M}^{TNL}$ is identical to the canonical model S by Gabbay et al. (1980), except for the added relation R_H^{TNL} and that the valuation function in S only takes standard atoms as input. Using the axioms (GP) and (HF) we can prove that for any $\Delta, \Theta \in W^{TNL}$: $\Delta < \Theta$ iff $\Theta > \Delta$. This is a standard proof in tense logic (Blackburn et al., 2001). The result implies that R_G^{TNL} and R_H^{TNL} is the same relation, and we only need to define R_G^{TNL} and R_X^{TNL} in the canonical model. Therefore, the frame structure of $\mathfrak{M}^{TNL}$ is now the same as S. From now on, the proof follows Gabbay et al. (1980). It is shown that for every $\Delta \in W^{TNL}$: $R_G^{TNL}(\Delta) \in W^{TNL}$, and that for all $\Delta, \Theta, \Gamma \in W^{TNL}$: if $\Delta < \Theta$, $\Delta < \Gamma$ and $\Theta \neq \Gamma$, then $\Theta < \Gamma$ or $\Gamma < \Theta$. It is also shown that if $F\psi \in \Delta$, then for some Θ : $\Delta < \Theta$ and $\psi \in \Theta$. The canonical relation R_G^{TNL} on the set W^{TNL} is therefore a linear order consistent with F, but it is not irreflexive. We will use the closure of a formula to build a finitely representable model that represents the natural numbers and their ordering. Proving completeness by constructing a model based on the closure of a formula is also used in weak completeness proofs for other non-compact logics such as S5 with common knowledge (van Ditmarsch et al., 2008) and propositional dynamic logic (Blackburn et al., 2001).

We define the closure $cl(\phi)$ of ϕ to be the smallest set such that for any ψ:

- $\phi \in cl(\phi)$;

- if $\psi \in cl(\phi)$, then $Sub(\psi) \subseteq \phi$, where $Sub(\psi)$ is the set of subformulas of ψ;
- $cl(\phi)$ is closed under Boolean operators;
- if $F\psi \in cl(\phi)$, then $X\psi, XF\psi \in cl(\phi)$;
- if $X\psi \in cl(\phi)$, then $X\neg\psi \in cl(\phi)$.

Now, define the set $\overline{W^{TNL}} = \{\Delta \cap cl(\phi) \mid \Delta \in W^{TNL}\}$ and the binary relations ρ_X, ρ_G on $\overline{W^{TNL}}$ such that:

- $(t, s) \in \rho_X$ iff for some $\Delta \in W^{TNL} : t = \Delta \cap cl(\phi)$ and $s = R_X^{TNL}(\Delta) \cap cl(\phi)$;
- $(t, s) \in \rho_G$ iff for some $\Delta, \Theta \in W^{TNL} : \Delta < \Theta$ and $t = \Delta \cap cl(\phi)$ and $s = \Theta \cap cl(\phi)$.

Gabbay et al. (1980) show that ρ_G is the transitive closure of ρ_X using the axiom (IND^+).

Then, we define our final model. Let s_0 be an element of $\overline{W^{TNL}}$ such that $\phi \in s_0$. Let S_n be a sequence of states from $\overline{W^{TNL}}$ such that for any $t \in \overline{W^{TNL}}$, if $t = s_i$ for infinitely many i, then every u such that $(t, u) \in \rho_X$ is also equal to infinitely many s_i. That is, if a state appears infinitely many times in the sequence, then so does each of its successors. Our final model is the sequence S_n with the valuation function $\overline{V^{TNL}}$ which is defined as V^{TNL} but with respect to elements in $\overline{W^{TNL}}$ instead of W^{TNL}. S_n is a strict linear order with no end. A truth lemma is proved by induction by Gabbay et al. (1980): that in the language of DX, for any $\psi \in cl(\phi)$: $(S_n, \overline{V^{TNL}}), s_n \Vdash \psi$ iff $\psi \in s_n$. For the language of TNL, we must make sure the proof also holds for the past operator and the $\langle follow\rangle$- and $\langle posted\rangle$ formulas. For formulas on the form $\langle follow\rangle(a, b)$ and $\langle posted\rangle(a, \omega)$, the induction proof can be extended to handle two additional separate base cases. We prove the case for ψ is $P\chi$.

Induction hypothesis: for any $\chi \in cl(\phi) : (S_n, \overline{V^{TNL}}), s_n \Vdash \chi$ iff $\chi \in s_n$.

ψ is $P\chi$: ($\Rightarrow$) Suppose that $(S_n, \overline{V^{TNL}}), s_n \Vdash P\chi$ for $P\chi \in cl(\phi)$. Then $\exists s_m \in \overline{W^{TNL}}$ such that $(s_m, s_n) \in \rho_G$ and $(S_n, \overline{V^{TNL}}), s_m \Vdash \chi$. Thus, for some $\Delta, \Theta \in W^{TNL}, \Delta < \Theta$ and $s_m = \Delta \cap cl(\phi)$ and $s_n = \Theta \cap cl(\phi)$. By the induction hypothesis, $\chi \in s_m$. Hence, $\chi \in \Delta$. Then, $P\chi \in \Theta$. Since $P\chi \in cl(\phi)$ by assumption, it follows that $P\chi \in s_n$.

($\Leftarrow$) Suppose that $P\chi \in s_n$. $s_n = \Delta \cap cl(\phi)$ for some $\Delta \in W^{TNL}$, and therefore $P\chi \in \Delta$. As mentioned earlier in the proof, Gabbay et al. (1980) show that for any ψ, any $\Gamma \in W^{TNL}$: if $F\psi \in \Gamma$, then for some $\Theta : \Gamma < \Theta$ and $\psi \in \Theta$. We assume that by the same reasoning, we have that for any ψ, any $\Gamma \in W^{TNL}$: if $P\psi \in \Gamma$, then for some $\Theta : \Theta < \Gamma$ and $\psi \in \Theta$. This step is common in standard completeness proofs for tense logics, see for instance (de Jongh et al., 2004). It follows that for some $\Theta : \Theta < \Delta$ and $\chi \in \Theta$. Since $cl(\phi)$ is closed under subformulas and $P\chi \in cl(\phi)$, we have that $\chi \in cl(\phi)$. Thus, $\chi \in \Theta \cap cl(\phi)$. By the induction hypothesis $(S_n, \overline{V^{TNL}}), \Theta \cap cl(\phi) \Vdash \chi$. Hence, $(S_n, V^{TNL}), s_n \Vdash P\chi$.

We have now proved that the truth lemma holds also for formulas in $\mathcal{L}_{TNL}$. Since $\phi \in s_0$, it follows that ϕ is satisfied in $(S_n, \overline{V^{TNL}})$. We can build a TNL model from

$(S_n, \overline{V^{TNL}})$ by letting $A = \mathsf{Ag}$ and by splitting the valuation function into three with respect to atoms, $\langle follow \rangle$- and $\langle posted \rangle$ formulas. Proof of completeness follows. □

Appendix B: Details on the model checking algorithm for TNL

In this section of the appendix, we give the subroutines SAT_P, SAT_F and SAT_X which Algorithm 1 calls in lines 8–10. The functions require the following definition.

Definition 9 (*Pre and Post-time*) Let $\mathfrak{M}_f = (A, T, \sim_f, N, V, Posts)$ be a finite fragment of a temporal network model and $S \subseteq T$. We define the **pre** and **post sets** of time points:

$$pre(S) = \{t \in T \mid \exists t' : t' = t + 1 \text{ and } t' \in S\}$$
$$post(S) = \{t \in T \mid \exists t' : t' = t - 1 \text{ and } t' \in S\}$$

Intuitively, $pre(S)$ is the set of time points that immediately precede each member of S and $post(S)$ the set of time points that immediately follow S. For example, if $\mathfrak{M}_f$ models natural numbers from 1 to 6 in increasing order and we let $S = \{3, 4\}$, then $pre(S) = \{2, 3\}$ and $post(S) = \{4, 5\}$. The subroutines of Algorithm 1 are defined in Algorithms 8, 9 and 10.

Algorithm 8 Function $SAT_P(\phi)$ determining the set of time points that satisfy $P\phi$

Input: Finite fragment $\mathfrak{M}_f = (A, T, \sim_f, N, V, Posts)$, formula $\phi \in \mathcal{L}_{TNL}$
Output: The set sat_P of time points in $\mathfrak{M}_f$ where $P\phi$ is satisfied

1: $sat_P := \emptyset$
2: $Y := post(SAT(\phi))$;
3: **repeat until** $Z = sat_P$
4: $\quad Z := sat_P$;
5: $\quad sat_P := Y \cup post(sat_P)$;
6: **end**
7: **return** sat_P

In Algorithm 8, we define function $SAT_P(\phi)$ which produces a set of time points sat_P in the given model $\mathfrak{M}_f$ where $P\phi$ is satisfied for a given formula ϕ. The algorithm relies on two local variables Y and Z. We call $SAT(\phi)$ from Algorithm 1 to define the post set of $SAT(\phi)$ (line 2). Then, we introduce the iterated steps of the function (lines 3–6). The function takes the post set of $SAT(\phi)$, all the time points strictly after each time point where ϕ is satisfied (line 4). In each iterative step, the variable includes the post set of that previous post set, until there is no change (line 5). The output returns the set sat_P of all post sets of post sets of ... of $SAT(\phi)$ (line 7). The next Algorithm 9 is similar, but uses the pre sets instead of post sets.

Algorithm 10 defines function $SAT_X(\phi)$ which determines the set sat_X of time points in $\mathfrak{M}_f$ where $X\phi$ is satisfied. Like the previous algorithms, this algorithm also calls $SAT(\phi)$ from Algorithm 1 (line 1). Then, it simply defines the pre set of $SAT(\phi)$ (line 2) and outputs this set (line 3).

Algorithm 9 Function $SAT_F(\phi)$ determining the set of time points that satisfy $F\phi$

Input: Finite fragment $\mathfrak{M}_f = (A, T, \sim_f, N, V, Posts)$, formula $\phi \in \mathcal{L}_{TNL}$
Output: The set sat_F of time points in $\mathfrak{M}_f$ where $F\phi$ is satisfied

$sat_F := \emptyset$;
$Y := pre(SAT(\phi))$;
repeat until $Z = sat_F$
  $Z := sat_F$;
  $sat_F := Y \cup pre(sat_F)$;
end
return sat_F

Algorithm 10 Function $SAT_X(\phi)$ determining the set of time points that satisfy $X\phi$

Input: Finite fragment $\mathfrak{M}_f = (A, T, \sim_f, N, V, Posts)$, formula $\phi \in \mathcal{L}_{TNL}$
Output: The set sat_X of time points in $\mathfrak{M}_f$ where $X\phi$ is satisfied

$X := SAT(\phi)$;
$sat_X := pre(X)$;
return sat_X

References

Areces, C., & ten Cate, B. (2007). Hybrid logics. Studies in logic and practical reasoning. In J. van Benthem, P. Blackburn, & F. Wolter (Eds.), *Handbook of modal logic* (Vol. 3, pp. 821–868). Elsevier.

Balbiani, P., & Fernández González, S. (2020). Indexed frames and hybrid logics. In *International Conference on Advances in Modal Logic (AiML 2020), Aug 2020* (pp. 56–72). Finland: University of Helsinki.

Balbiani, P., & Fernández González, S. (2021). Orthogonal frames and indexed relations. In A. Silva, R. Wassermann, & R. de Queiroz (Eds.), *Logic, Language, Information, and Computation. WoLLIC 2021*. Lecture Notes in Computer Science (Vol. 13038, pp. 219–234). Springer. https://doi.org/10.1007/978-3-030-88853-4_14

Baltag, A., Christoff, Z., Rendsvig, R. K., & Smets, S. (2019). Dynamic epistemic logics of diffusion and prediction in social networks. *Studia Logica, 107*(3), 489–531.

Belardinelli, F., & Grossi, D. (2015). On the formal verification of diffusion phenomena in open dynamic agent networks. In *Proc. of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), May 2015* (pp. 237–245). Istanbul, Turkey.

Belardinelli, G. (2019). *Gatekeepers in social networks: Logics for communicative actions*. Master of Logic Thesis Series, MoL-2019-20, ILLC, University of Amsterdam.

Bernardeschi, C., Mercaldo, F., Nardone, V., & Santone, A. (2019). Exploiting model checking for mobile botnet detection. *Procedia Computer Science, 159*, 963–972.

Blackburn, P., de Rijke, M., & Venema, Y. (2001). *Modal logic*. Cambridge University Press.

Blackburn, P., & ten Cate, B. (2006). Pure extensions, proof rules, and hybrid axiomatics. *Studia Logica, 84*(2), 277–322.

Cao, Q., Sirivianos, M., Yang, X., & Pregueiro, T. (2012). Aiding the detection of fake accounts in large scale social online services. In *Proc. of the 9th USENIX symposium on Networked Systems Design and Implementation (NSDI 12)* (pp. 197–210).

Chang, H.-C., & Iyer, H. (2012). Trends in twitter hashtag applications: Design features for value-added dimensions to future library catalogues. *Library Trends, 61*(1), 248–258.

Christoff, Z., & Grossi, D. (2017). Stability in binary opinion diffusion. In A. Baltag, J. Seligman, & T. Yamada (Eds.), *Logic, rationality, and interaction. LORI 2017*. Lecture Notes in Computer Science (Vol. 10455, pp. 166–180). Springer. https://doi.org/10.1007/978-3-662-55665-8_12

Christoff, Z., & Hansen, J. U. (2015). A logic for diffusion in social networks. *Journal of Applied Logic, 13*(1), 48–77.

Christoff, Z., Hansen, J. U., & Proietti, C. (2016). Reflecting on social influence in networks. *Journal of Logic, Language and Information, 25*(3), 299–333.

Chu, Z., Gianvecchio, S., Wang, H., & Jajodia, S. (2012). Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing, 9*(6), 811–824.

Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., & Tacchella, A. (2002). NuSMV 2: an opensource tool for symbolic model checking. In E. Brinksma & K. G. Larsen (Eds.), *Computer aided verification. CAV 2002*. Lecture Notes in Computer Science (Vol. 2404, pp. 359–364). Springer. https://doi.org/10.1007/3-540-45657-0_29

Clarke, E. M., & Emerson, E. A. (1981). Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logics of Programs. Logic of Programs 1981*. Lecture Notes in Computer Science (Vol. 131, pp. 52–71). Springer. https://doi.org/10.1007/BFb0025774

Cresci, S. (2020). A decade of social bot detection. *Communications of the ACM, 63*(10), 72–83.

Cristani, M., Olivieri, F., & Santacà, K. (2020). Social networks as communication channels: a logical approach. In M. Brambilla, C. Cappiello, & S. H. Ow (Eds.), *Current trends in Web Engineering. ICWE 2019*. Lecture Notes in Computer Science (Vol. 11609, pp. 61–73). Springer. https://doi.org/10.1007/978-3-030-51253-8_8

Dennis, L. A., Fu, Y., & Slavkovik, M. (2022). Markov chain model representation of information diffusion in social networks. *Journal of Logic and Computation, 32*(6), 1195–1211.

Dennis, L. A., & Slavkovik, M. (2020). Model-checking information diffusion in social networks with prism. In N. Bassiliades, G. Chalkiadakis, & D. de Jonge (Eds.), *Multi-agent Systems and Agreement Technologies. EUMAS 2020*. Lecture Notes in Computer Science (Vol. 12520, pp. 475–492). Springer. https://doi.org/10.1007/978-3-030-66412-1_30

Dennis, L. A., Slavkovik, M., & Fisher, M. (2017). "How did they know?"—Model-checking for analysis of information leakage in social networks. In S. Cranefield, S. Mahmoud, J. Padget, & A. P. Rocha (Eds.), *Coordination, Organizations, Institutions, and Norms in Agent Systems XII. COIN COIN 2016 2016*. Lecture Notes in Computer Science (Vol. 10315, pp. 42–59). Springer. https://doi.org/10.1007/978-3-319-66595-5_3

van Ditmarsch, H., van der Hoek, W., & Kooi, B. (2008). *Dynamic epistemic logic*. Springer.

Easley, D., & Kleinberg, J. (2010). *Networks, crowds and markets*. Cambridge University Press.

Fernández González, S. (2021). *Logics for social networks: Asynchronous announcements in orthogonal structures*. PhD thesis, Université de Toulouse.

Fernández González, S. (2022). Change in social networks: Some dynamic extensions of social epistemic logic. *Journal of Logic and Computation, 32*(6), 1212–1233.

Fionda, V., & Greco, G. (2016). The complexity of LTL on finite traces: Hard and easy fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 30, No. 1). https://doi.org/10.1609/aaai.v30i1.10104

Franceschet, M., & de Rijke, M. (2006). Model checking hybrid logics (with an application to semistructured data). *Journal of Applied Logic, 4*(3), 279–304.

Gabbay, D. (1989). The declarative past and imperative future. In B. Banieqbal, H. Barringer, & A. Pnueli (Eds.), *Temporal Logic in Specification*. Lecture Notes in Computer Science (Vol. 398, pp. 409–448). Springer. https://doi.org/10.1007/3-540-51803-7_36

Gabbay, D., Pnueli, A., Shelah, S., & Stavi, J. (1980). On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on principles of programming languages. POPL '80* (pp. 163–173). Association for Computing Machinery.

Galimullin, R., Pedersen, M. Y., & Slavkovik, M. (2022). Logic of visibility in social networks. In A. Ciabattoni, E. Pimentel, & R. J. G. B. de Queiroz (Eds.), *Logic, Language, Information, and Computation. WoLLIC 2022*. Lecture Notes in Computer Science (Vol. 13468, pp. 190–206). Springer. https://doi.org/10.1007/978-3-031-15298-6_12

Gilani, Z., Kochmar, E., & Crowcroft, J. (2017). Classification of twitter accounts into automated agents and human users. In *Proc. of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining* (pp. 489–496).

Gorodnichenko, Y., Pham, T., & Talavera, O. (2018). *Social media, sentiment and public opinions: Evidence from #brexit and #uselection*. National Bureau of Economic Research: Technical report.

Grandi, U., Lorini, E., Novaro, A., & Perrussel, L. (2017). Strategic disclosure of opinions on a social network. In *Proceedings of the 16th conference on autonomous agents and multiagent systems. AAMAS '17* (pp. 1196–1204). International Foundation for Autonomous Agents and Multiagent Systems.

Hodkinson, I., & Reynolds, M. (2005). Separation—past, present and future. In S. Artemov, H. Barringer, A. S. d'Avila Garcez, L. C. Lamb, & J. Woods (Eds.), *We will show them: Essays in Honour of Dov Gabbay* (Vol. 2, pp. 117–142). College Publications.

van der Hoek, W., Kuijer, L. B., & Wáng, Y. N. (2019). Who should be my friends? In P. Blackburn, E. Lorini, & M. Guo (Eds.), *Logic, Rationality, and Interaction. LORI 2019*. Lecture Notes in Computer Science (Vol. 11813, pp. 370–384). Springer. https://doi.org/10.1007/978-3-662-60292-8_27

van der Hoek, W., Kuijer, L. B., & Wáng, Y. N. (2020). Logics of allies and enemies: a formal approach to the dynamics of social balance theory. In C. Bessiere (Eds.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020* (pp. 210–216).

van der Hoek, W., Kuijer, L. B., & Wáng, Y. (2022). Who should be my friends? Social balance from the perspective of game theory. *Journal of Logic, Language and Information, 31*(2), 189–211.

Holzmann, G. J. (1997). The model checker spin. *IEEE Transactions on Software Engineering, 23*(5), 279–295.

Huth, M., & Ryan, M. (2004). *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press.

de Jongh, D., Veltman, F., & Verbrugge, R. (2004). *Completeness by construction for tense logics of linear time*. Liber Amicorum for Dick de Jongh. Institute of Logic, Language and Computation.

Kant, G., Laarman, A., Meijer, J., van de Pol, J., Blom, S., & van Dijk, T. (2015). Ltsmin: High-performance language-independent model checking. In C. Baier & C. Tinelli (Eds.), *Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2015*. Lecture Notes in Computer Science (Vol. 9035, pp. 692–707). Springer. https://doi.org/10.1007/978-3-662-46681-0_61

Khaund, T., Al-Khateeb, S., Tokdemir, S., & Agarwal, N. (2018). Analyzing social bots and their coordination during natural disasters. In R. Thomson, C. Dancy, A. Hyder, & H. Bisgin (Eds.), *Social, Cultural, and Behavioral Modeling. SBP-BRiMS 2018*. Lecture Notes in Computer Science (Vol. 10899, pp. 207–212). Springer. https://doi.org/10.1007/978-3-319-93372-6_23

Kudugunta, S., & Ferrara, E. (2018). Deep neural networks for bot detection. *Information Sciences, 467*, 312–322.

Lee, K., Eoff, B. D., & Caverlee, J. (2011). Seven months with the devils: A long-term study of content polluters on twitter. In *ICWSM* (pp. 185–192).

Lee, S., & Kim, J. (2014). Early filtering of ephemeral malicious accounts on twitter. *Computer Communications, 54*, 48–57.

Liu, F., & Li, D. (2022). Ten-year history of social network logics in China. *Asian Studies, 10*(2), 121–146.

Liu, F., Seligman, J., & Girard, P. (2014). Logical dynamics of belief change in the community. *Synthese, 191*(11), 2403–2431.

Lorini, E., & Sartor, G. (2016). A STIT logic for reasoning about social influence. *Studia Logica, 104*(4), 773–812.

Machado, V., & Benevides, M. (2022). Temporal logic for social networks. *Journal of Logic and Computation, 32*(6), 1088–1108.

Markey, N. (2004). Past is for free: On the complexity of verifying linear temporal properties with past. *Acta Informatica, 40*(6), 431–458.

Occhipinti Liberman, A., & Rendsvig, R. K. (2022). Reasoning about epistemic social network dynamics using dynamic term-modal logic. *Journal of Logic and Computation, 32*(6), 1067–1087.

O'Connor, C., & Weatherall, J. O. (2019). *The misinformation age: How false beliefs spread*. Yale University Press.

Pacuit, E. (2017). *Neighborhood semantics for modal logic*. Springer.

Pacuit, E., & Parikh, R. (2005). The logic of communication graphs. In J. Leite, A. Omicini, P. Torroni & P. Yolum (Eds.), *Declarative Agent Languages and Technologies II. DALT 2004*. Lecture Notes in Computer Science (Vol. 3476, pp. 256–269). Springer. https://doi.org/10.1007/11493402_15

Pardo, R., Sánchez, C., & Schneider, G. (2018). Timed epistemic knowledge bases for social networks. In K. Havelund, J. Peleska, B. Roscoe, & E. de Vink (Eds.), *Formal Methods. FM 2018*. Lecture Notes in Computer Science (Vol. 10951, pp. 185–202). Springer. https://doi.org/10.1007/978-3-319-95582-7_11

Pardo, R., & Schneider, G. (2017). Model checking social network models. In *Proc. of eighth international symposium on Games, Automata, Logics and Formal Verification (GandALF). EPTCS* (Vol. 256, pp. 238–252).

Pedersen, M. Y. (2019). *Polarization and echo chambers: A logical analysis of balance and triadic closure in social networks*. Master of Logic Thesis Series, MoL-2019-10, ILLC, University of Amsterdam.

Pedersen, M. Y., Slavkovik, M., & Smets, S. (2021). Social bot detection as a temporal logic model checking problem. In S. Ghosh & T. Icard (Eds.), *Logic, Rationality, and Interaction. LORI 2021*. Lecture Notes

in Computer Science (Vol. 13039, pp. 158–173). Springer. https://doi.org/10.1007/978-3-030-88708-7_13

Pedersen, M. Y., Smets, S., & Ågotnes, T. (2019). Analyzing echo chambers: A logic of strong and weak ties. In P. Blackburn, E. Lorini, & M. Guo (Eds.), *Logic, Rationality, and Interaction. LORI 2019*. Lecture Notes in Computer Science (Vol. 11813, pp. 183–198). Springer. https://doi.org/10.1007/978-3-662-60292-8_14

Pedersen, M. Y., Smets, S., & Ågotnes, T. (2020). Further steps towards a logic of polarization in social networks. In M. Dastani, H. Dong, & L. van der Torre (Eds.), *Logic and Argumentation. CLAR 2020*. Lecture Notes in Computer Science (Vol. 12061, pp. 324–345). Springer. https://doi.org/10.1007/978-3-030-44638-3_20

Pedersen, M. Y., Smets, S., & Ågotnes, T. (2021). Modal logics and group polarization. *Journal of Logic and Computation, 31*(8), 2240–2269.

Pedersen, T., & Slavkovik, M. (2017). Formal models of conflicting social influence. In B. An, A. Bazzan, J. Leite, S. Villata, & L. van der Torre (Eds.), *PRIMA 2017: Principles and Practice of Multi-agent Systems. PRIMA 2017*. Lecture Notes in Computer Science (Vol. 10621, pp. 349–365). Springer. https://doi.org/10.1007/978-3-319-69131-2_21

Perrotin, E., Galimullin, R., Canu, Q., & Alechina, N. (2019). Public group announcements and trust in doxastic logic. In P. Blackburn, E. Lorini, & M. Guo (Eds.), *Logic, Rationality, and Interaction. LORI 2019*. Lecture Notes in Computer Science (Vol. 11813, pp. 199–213). Springer. https://doi.org/10.1007/978-3-662-60292-8_15

Pradella, M., Pietro, P. S., Spoletini, P., & Morzenti, A. (2003). Practical model checking of LTL with past. In *1st International Workshop on Automated Technology for Verification and Analysis (ATVA03)*.

Prior, A. N. (1957). *Time and modality*. Oxford University Press.

Ruan, J., & Thielscher, M. (2011). A logic for knowledge flow in social networks. In D. Wang, & M. Reynolds (Eds.), *AI 2011: Advances in Artificial Intelligence—24th Australasian Joint Conference. AI 2011*. Lecture Notes in Computer Science (Vol. 7106, pp. 511–520). Springer. https://doi.org/10.1007/978-3-642-25832-9_52

Sano, K. (2017). Axiomatizing epistemic logic of friendship via tree sequent calculus. In A. Baltag, J. Seligman, & T. Yamada (Eds.), *Logic, Rationality, and Interaction. LORI 2017*. Lecture Notes in Computer Science (Vol. 10455, pp. 224–239). Springer. https://doi.org/10.1007/978-3-662-55665-8_16

Santos, Y. D. (2020). Social consolidations: rational belief in a many-valued logic of evidence and peerhood. Lecture Notes in Computer ScienceIn A. Herzig & J. Kontinen (Eds.), *Foundations of Information and Knowledge Systems. FoIKS 2020*. Lecture Notes in Computer Science (Vol. 12012, pp. 58–78). Springer. https://doi.org/10.1007/978-3-030-39951-1_4

Schnoebelen, P. (2002). The complexity of temporal logic model checking. *Advances in Modal Logic, 4*(35), 393–436.

Seligman, J., Liu, F., & Girard, P. (2011). Logic in the community. In M. Banerjee & A. Seth (Eds.), *Logic and Its Applications. ICLA 2011*. Lecture Notes in Computer Science (Vol. 6521, pp. 178–188). Springer. https://doi.org/10.1007/978-3-642-18026-2_15

Seligman, J., Liu, F., & Girard, P. (2013). Facebook and the epistemic logic of friendship. In *Proceedings of the 14th TARK conference*.

Shao, C., Ciampaglia, G. L., Varol, O., Yang, K.-C., Flammini, A., & Menczer, F. (2018). The spread of low-credibility content by social bots. *Nature Communications, 9*(1), 4787.

Sistla, A. P., & Clarke, E. M. (1985). The complexity of propositional linear temporal logics. *Journal of the ACM, 32*(3), 733–749.

Smets, S., & Velázquez-Quesada, F. R. (2017a). How to make friends: A logical approach to social group creation. Lecture Notes in Computer Science. In A. Baltag, J. Seligman, & T. Yamada (Eds.), *Logic, Rationality, and Interaction. LORI 2017*. Lecture Notes in Computer Science (Vol. 10455, pp. 377–390). Springer. https://doi.org/10.1007/978-3-662-55665-8_26

Smets, S., & Velázquez-Quesada, F. R. (2017b). The creation and change of social networks: A logical study based on group size. In A. Madeira, & M.R.F. Benevides (Eds.) *Dynamic logic. New trends and applications—first international workshop, DALI 2017*, Brasilia, Brazil, September 23–24, 2017, Proceedings. Lecture Notes in Computer Science, (Vol. 10669, pp. 171–184). Springer.

Smets, S., & Velázquez-Quesada, F. R. (2020a). A closeness- and priority-based logical study of social network creation. *Journal of Logic, Language and Information, 29*(1), 21–51.

Smets, S., & Velázquez-Quesada, F. R. (2020b). A logical analysis of the interplay between social influence and friendship selection. In L. S. Barbosa, & A. Baltag (Eds.), *Dynamic Logic. New Trends and Applications—Second International Workshop*, DaLí 2019, Porto, Portugal, October 7–11, 2019, Proceedings. Lecture Notes in Computer Science (Vol. 12005, pp. 71–87).

Thorne, J., & Vlachos, A. (2018) Automated fact checking: task formulations, methods and future directions. In *Proc. of the 27th international conference on computational linguistics* (pp. 3346–3359).

Venema, Y. (2001). Temporal logic. In L. Goble (Ed.), *The Blackwell guide to philosophical logic* (pp. 203–223). Blackwell Publishers.

Viana, H., Araújo, A., Leite, L., & Alcântara, J. (2014). Private dynamic epistemic friendship logic. In *2014 Brazilian conference on intelligent systems* (pp. 378–383).

Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. *Science, 359*(6380), 1146–1151.

Xiong, Z. (2017). *On the logic of multicast messaging and balance in social networks*. PhD thesis, Doctoral dissertation, University of Bergen.

Xiong, Z., & Ågotnes, T. (2020). On the logic of balance in social networks. *Journal of Logic, Language and Information, 29*(1), 53–75.

Xiong, Z., Ågotnes, T., Seligman, J., & Zhu, R. (2017). Towards a logic of tweeting. In A. Baltag, J. Seligman, & T. Yamada (Eds.), *Logic, Rationality, and Interaction. LORI 2017*. Lecture Notes in Computer Science (Vol. 10455, pp. 49–64). Springer. https://doi.org/10.1007/978-3-662-55665-8_4

Yang, S., Taniguchi, M., & Tojo, S. (2019). 4-valued logic for agent communication with private/public information passing. In *Proceedings of the 11th international conference on agents and artificial intelligence (ICAART 2019)* (pp. 54–61).

Zhen, L. (2020). *Towards axiomatisation of social epistemic logic*. PhD thesis, University of Auckland.