MANCaLog: A Logic for Multi-Attribute Network Cascades (Technical Report)

Paulo Shakarian
Network Science Center and
Dept. of Electrical Engineering
and Computer Science
U.S. Military Academy
West Point, NY 10996
paulo@shakarian.net

Gerardo I. Simari
Network Science Center and
Dept. of Computer Science
University of Oxford
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
gerardo.simari@cs.ox.ac.uk

Robert Schroeder
CORE Lab
Defense Analysis Dept.
Naval Postgraduate School
Monterey, CA 93943
rcschroe@nps.edu

ABSTRACT

The modeling of cascade processes in multi-agent systems in the form of complex networks has in recent years become an important topic of study due to its many applications: the adoption of commercial products, spread of disease, the diffusion of an idea, etc. In this paper, we begin by identifying a desiderata of seven properties that a framework for modeling such processes should satisfy: the ability to represent attributes of both nodes and edges, an explicit representation of time, the ability to represent non-Markovian temporal relationships, representation of uncertain information, the ability to represent competing cascades, allowance of non-monotonic diffusion, and computational tractability. We then present the MANCaLog language, a formalism based on logic programming that satisfies all these desiderata, and focus on algorithms for finding minimal models (from which the outcome of cascades can be obtained) as well as how this formalism can be applied in real world scenarios. We are not aware of any other formalism in the literature that meets all of the above requirements.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Representation Languages*

General Terms

Languages, Algorithms

Keywords

Complex Networks, Cascades, Logic Programming

1. INTRODUCTION AND RELATED WORK

An epidemic working through a population, cascading electrical power failures, product adoption, and the spread of a mutant gene are all examples of diffusion processes that can happen in multi-agent systems structured as complex networks. These network processes have been studied in a variety of disciplines, including computer science [10], biology [11], sociology [8], economics [12], and physics [15]. Much existing work in this area is based on pre-existing models in sociology and economics – in particular the work of [8, 12]. However, recent examinations of social networks – both analysis of large data sets and experimental – have indicated that there may be additional factors to consider that are not taken into account by these models. These include the attributes of nodes and edges, competing diffusion processes, and time. In this paper, we outline seven design criteria (Section 1.1) for such a framework and introduce MANCaLog (Section 2), which is to the best of our knowledge the first logical language for modeling diffusion in complex networks that meets these criteria. MANCaLog is a rule-based framework (inspired by logic programming) that can richly express how agents adopt or fail to adopt certain behaviors, and how these behaviors cascade through a network. We also introduce fixed-point based algorithms that allow for the calculation of the result of the diffusion process in Section 3. Note that these algorithms are proven not only to be correct, but also to run in polynomial time. Hence, our approach can not only better express many aspects of cascades in complex networks, but it can do so in a reasonable amount of time. We conclude by discussing applications of MANCaLog in Section 4.

Proofs of all results stated in this paper can be found in the appendix.

This is an extended version of *MANCaLog: A Logic for Multi-Attribute Network Cascades*, which appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May 6–10, 2013, Saint Paul, Minnesota, USA.*

1.1 Desiderata of Properties

We begin by identifying a set of criteria that we believe a framework for reasoning about cascades in complex networks should satisfy.

1. Multiply labeled and weighted nodes and edges. Many existing frameworks for studying diffusion in complex networks assume that there is only one type of vertex that may become "active" [10] or may "mutate" [11, 15] and only one possible relationship between nodes. In reality, nodes and edges often have different properties. For instance, labels on edges can be used to differentiate between strong and weak ties (edge types) – a concept that is well studied [7]. Recently, such attributes of nodes have been shown to impact influence in a network [1].

2. Explicit Representation of Time. Most work in the literature assumes static models, with the exception of the recent developments in [4, 5, 6], which assume the existence of a timestamped log referring to actions taken

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2013		**00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
MANCaLog: A Logic for Multi-Attribute Network Cascades	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Dept. Electrical Engineering and Computer Science and,Network Science Center,U.S. Military Academy,West Point,NY,10996**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**11**	

in the network in order to learn how nodes influence each other. Though [4] tackles the problem of predicting the time at which a certain node will take an action, the authors make several simplifying assumptions such as monotonicity of probability functions, probabilistic independence, sub-modularity and, most importantly for this criterion, a modeling of time solely based on temporal decay of influence. We seek a richer model of temporal relationships between conditions in the network structure, the current state of the cascades in process, and how influence propagates.

3. Non-Markovian Temporal Relationships. Apart from time being explicitly represented, the temporal dependencies should be able to span multiple units of time. Hence, the "memoryless" mode of a standard Markov process, where only the information of the current state is required, is insufficient. Here, we strive to create a framework where dependencies can be from other earlier time steps. This issue has been previously studied with respect to more general logic programming frameworks such as [13], but to our knowledge has not been applied to social networks.

4. Representation of Uncertainty. As in practice it is not always possible to judge the attributes of all individuals in a network, an element of uncertainty must be included. However, in connection with point 7, this should not be at the expense of tractability. For instance, the probabilistic models of [10] are normally addressed with simulation (and hence do not scale well) as the computation of the expected number of activated nodes is a $\#P$-hard problem [3].

5. Competing Cascades. Often, in real-world situations there will be competing cascading processes. For example, in evolutionary graph theory [11], "mutants" and "residents" compete for nodes in the network – the success of one hinges on the failure of the other.

6. Non-Monotonic Cascades. In much existing work on cascades in complex networks, the number of nodes attaining a certain property at each time step can only increase. However, if we allow for competing cascades in the same model, we cannot have such a strong restriction as the success of one cascade may come at the expense of another.

7. Tractability. The social networks of interest in today's data mining problems often have millions of nodes. It is reasonable to expect that soon billion-node networks will be commonplace. Any framework for dealing with these problems must be solvable in a reasonable amount of time and offer areas for practical improvement for further scalability.

1.2 Related Work

The above criteria can be summarized as the desire to design the most expressive language for network cascades possible while still allowing computation of the outcome of a diffusion process to be completed in a tractable amount of time. As a comparison, let us briefly describe some relevant related work. Perhaps the best known general model for representing diffusion in complex networks is the independent cascade/linear threshold (IC/LT) model of [10]. However, although this framework was shown to be capable of expressing a wide variety of sociological models, it assumes the Markov property and does not allow for the representation of multiple attributes on vertices and edges. A more recent framework, social network optimization problems (SNOPs) [14] uses logic programming to allow for the representation of attributes, but this framework does not allow for competing processes or non-monotonic cascades. A related logic programming framework, competitive diffusion (CD) [2] allows for competitive diffusion and non-monotonic processes but does not explicitly represent time and also makes Markovian assumptions. Further, we also note that the semantics of CD yields a "most probable interpretation" that is not a unique solution. Hence, a given model in that framework can lead to multiple and possibly contradictory, outcomes to a cascade (this problem is avoided in MANCaLog). Another popular class of models is Evolutionary Graph Theory (EGT) [11], which is highly related to the voter model (VM) [15]. Although this framework allows for competing processes and non-monotonic diffusion, it also makes Markovian assumptions while not explicitly representing time. Further, determining the outcome of a cascade in those models is NP-hard, while determining the outcome in MANCaLog can be accomplished in polynomial time. Table 1 lists how these models compare to MANCaLog when considering our design criteria.

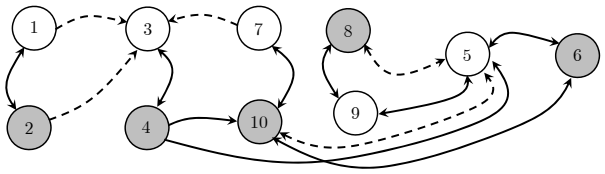

Figure 1: Simple online social network G_{soc}. Solid edges are labeled with *strTie*, while dashed edges are labeled with *wkTie*. White nodes are labeled with *male*, while gray nodes are labeled with *fem*. Arrows represent the direction of the edge; double-headed edges represent two edges with the same label.

2. FRAMEWORK

2.1 Syntax and Semantics

In this paper we assume that agents are arranged in a directed graph (or network) $G = (V, E)$, where the set of nodes corresponds to the agents, and the edges model the relationships between them. We also assume a set of labels $\mathcal{L}$, which is partitioned into two sets: *fluent* labels $\mathcal{L}_f$ (labels that can change over time) and *non-fluent* labels $\mathcal{L}_{nf}$ (labels that do not); labels can be applied to both the nodes and edges of the network. We will use the notation $\mathcal{G} = V \cup E$ to be the set of all *components* (nodes and edges) in the network. Thus, $c \in \mathcal{G}$ could be either a node or an edge.

EXAMPLE 2.1. *We will use the sample online social network G_{soc} shown in Figure 1 as the running example; $\mathcal{G}_{soc}$ is used to denote the set of components of G_{soc}. Here we have $\mathcal{L}_{nf} = \{male, fem, strTie, wkTie\}$ representing male, female, strong ties and weak ties, respectively. Additionally, we have $\mathcal{L}_f = \{visPgA, visPgB\}$ representing visiting webpage A and visiting webpage B, respectively.* ∎

In this paper, we present a logical language where we use atoms, referring to labels and weights, to describe properties of the nodes and edges. Though labels themselves could be modeled as atoms instead of predicates (to model non-ground labelings that allow for greater expressibility), for simplicity of presentation we leave this to future work. The first piece of the syntax is the **network atom**.

Table 1: Comparison with other models

Criterion	MANCaLog	IC/LT [10]	SNOP [14]	CD [2]	EGT/VM [11]
1. Labels	Yes	No	Yes	Yes	No
2. Explicit Representation of Time	Yes	No	Yes	No	Yes
3. Non-Markovian Temporal Relationships	Yes	No	No	No	No
4. Uncertainty	Yes	Yes	Yes	Yes	Yes
5. Competing Cascades	Yes	No	No	Yes	Yes
6. Non-monotonic Cascades	Yes	No	No	Yes	Yes
7. Tractablity	PTIME	#P-hard	PTIME	PTIME	NP-hard

Definition 2.1 (Network Atom). *Given label $L \in \mathcal{L}$ and weight interval $bnd \subseteq [0,1]$, then $\langle L, bnd\rangle$ is a **network atom**. A network atom is fluent (resp., non-fluent) if $L \in \mathcal{L}_f$ (resp., $L \in \mathcal{L}_{nf}$). We use NA to denote the set of all possible network atoms.*

Network atoms describe properties of nodes and edges. The definition is intuitive: L represents a property of the vertex or edge, and associated with this property is some weight that may have associated uncertainty – hence represented as an interval *bnd*, which can be open or closed. An invalid bound is represented by $\emptyset$, which is equivalent to all other invalid bounds.

Definition 2.2 (World). *A world W is a set of network atoms such that for each $L \in \mathcal{L}$ there is no more than one network atom of the form $\langle L, bnd\rangle$ in W.*

A **network formula** over NA is defined using conjunction, disjunction, and negation in the usual way. If a formula contains only non-fluent (resp., fluent) atoms, it is a non-fluent (resp., fluent) formula.

Definition 2.3 (Satisfaction of Worlds). *Given a world W and network formula f, **satisfaction** of W by f (denoted $W \models f$) is defined:*

- *If $f = \langle L, [0,1]\rangle$ then $W \models f$.*
- *If $f = \langle L, \emptyset\rangle$ then $W \not\models f$.*
- *If $f = \langle L, bnd\rangle$, with $bnd \neq \emptyset$ and $bnd \neq [0,1]$, then $W \models f$ iff there exists $\langle L, bnd'\rangle \in W$ s.t. $bnd' \subseteq bnd$.*
- *If $f = \neg f'$ then $W \models f$ iff $W \not\models f'$.*
- *If $f = f_1 \wedge f_2$ then $W \models f$ iff $W \models f_1$ and $W \models f_2$.*
- *If $f = f_1 \vee f_2$ then $W \models f$ iff $W \models f_1$ or $W \models f_2$.*

For some arbitrary label $L \in \mathcal{L}$, we will use the notation $\mathsf{Tr} = \langle L, [0,1]\rangle$ and $\mathsf{F} = \langle L, \emptyset\rangle$ to represent a tautology and contradiction, respectively. For ease of notation (and without loss of generality), we say that if there does not exist some *bnd* s.t. $\langle L, bnd\rangle \in W$, then this implies that $\langle L, [0,1]\rangle \in W$.

Example 2.2. *Following from Example 2.1, the network atom $\langle female, [1,1]\rangle$ can be used to identify a node as a woman. Likewise, the world $W =$*

$$\Big\{\langle fem, [1,1]\rangle, \langle male, [0,0]\rangle, \langle visPgA, [1,1]\rangle, \langle visPgB, [0,0]\rangle\Big\}$$

might be used to identify a woman who visits webpage A. Clearly, we have that $W \models$

$$\langle fem, [1,1]\rangle \wedge \neg\langle visPgA, [0.5, 0.9]\rangle \wedge \neg\langle visPgB, [0.1, 0.7]\rangle$$

Note that the network atoms formed with strTie and wkTie are not present; this could be due to the fact that such a world is used to describe a node and not an edge, and hence there is no information about those two labels. As such is the case, $W \models \langle strTie, [0,1]\rangle \wedge \langle wkTie, [0,1]\rangle$. ∎

The idea is to use MANCaLog to describe how properties (specified by labels) of the nodes in the network change over time. We assume that there is some natural number t_{max} that specifies the total amount of time we are considering, and we use $\tau = \{t \mid t \in [0, t_{max}]\}$ to denote the set of all time points. How well a certain property can be attributed to a node is based on a *weight* (to which the *bnd* bound in the network atom refers). As time progresses, a weight can either increase/decrease and/or become more/less certain. We now introduce the MANCaLog fact, which states that some network atom is true for a node or edge during certain times.

Definition 2.4 (MANCaLog Fact). *If $[t_1, t_2] \subseteq [0, t_{max}]$, $c \in \mathcal{G}$, and $a \in NA$, then $(a, c) : [t_1, t_2]$ is a **MANCaLog fact**. A fact is fluent (resp., non-fluent) if atom a is fluent (resp., non-fluent). All non-fluent facts must be of the form $(a, c) : [0, t_{max}]$. Let $\mathcal{F}$ be the set of all facts and $\mathcal{F}_{nf}, \mathcal{F}_f$ be the set of all non-fluent and fluent facts, respectively.*

Example 2.3. *Following from Example 2.2, the following facts are based on Figure 1:*

$$\begin{aligned}
F_1 &= (\langle male, [1,1]\rangle, 1) : [0, t_{max}]\\
F_2 &= (\langle fem, [1,1]\rangle, 1) : [0, t_{max}]\\
F_3 &= (\langle male, [1,1]\rangle, 3) : [0, t_{max}]\\
F_4 &= (\langle strTie, [1,1]\rangle, (1,2)) : [0, t_{max}]\\
F_5 &= (\langle strTie, [1,1]\rangle, (2,1)) : [0, t_{max}]\\
F_6 &= (\langle wkTie, [1,1]\rangle, (2,3)) : [0, t_{max}]\\
F_7 &= (\langle visPgA, [0.8, 1.0]\rangle, 1) : [0, t_{max}]\\
F_8 &= (\langle visPgA, [0.5, 1.0]\rangle, 2) : [0, t_{max}]
\end{aligned}$$

For instance, agent 1 is male, and has a strong tie to agent 2, who is female. ∎

Next, we introduce integrity constraints (ICs).

Definition 2.5. *Given fluent network atom a and conjunction of network atoms b, an integrity constraint is of the form $a \hookleftarrow b$.*

Intuitively, integrity constraint $\langle L, bnd\rangle \hookleftarrow b$ means that if at a certain time point a component (vertex or edge) of the network has a set of properties specified by conjunction b, then at that same time the component's weight for label L must be in interval bnd.

EXAMPLE 2.4. *Following from the previous examples, the integrity constraint* $\langle male, [0,0]\rangle \hookleftarrow \langle fem, [1,1]\rangle$ *would require any node designated as a female to not be male.* ■

We now define MANCaLog rules. The idea behind rules is simple: an agent that meets some criteria is influenced by the set of its neighbors who possess certain properties. The amount of influence exerted on an agent by its neighbors is specified by an *influence function*, whose precise effects will be described later on when we discuss the semantics. As a result, a rule consists of four major parts: (i) an influence function, (ii) neighbor criteria, (iii) target criteria, and (iv) a target. Intuitively, (i) specifies how the neighbors influence the agent in question, (ii) specifies which of the neighbors can influence the agent, (iii) specifies the criteria that cause the agent to be influenced, and (iv) is the property of the agent that changes as a result of the influence.

We will discuss each of these parts in turn, and then define rules in terms of these elements. First, we define influence functions and neighbor criteria.

DEFINITION 2.6 (INFLUENCE FUNCTION). *An* ***influence function*** *is a function* $ifl : \boldsymbol{N} \times \boldsymbol{N} \to [0,1] \times [0,1]$ *that satisfies the following two axioms:*

1. *ifl can be computed in constant* ($O(1)$) *time.*
2. *For* $x' > x$ *we have* $ifl(x', y) \subseteq ifl(x, y)$.

We use IFL to denote the set of all influence functions.

Intuitively, an influence function takes the number of qualifying influencers and the number of eligible influencers and returns a bound on the new value for the weight of the property of the target node that changes. In practice, we expect the time complexity of such a function to be a polynomial in terms of the two arguments. However, as both arguments are naturals bounded by the maximum degree of a node in the network, this value will be much smaller than the size of the network – we thus treat it as a constant here.

EXAMPLE 2.5. *The well-known "tipping model" originally introduced in [8, 12] states that an agent adopts a behavior when a certain fraction of his incoming neighbors do so. A common tipping function is the* ***majority threshold*** *where at least half of the agent's neighbors must previously adopt the behavior. We can represent this using the following influence function:*

$$tip(x,y) = \begin{cases} [1.0, 1.0] & \text{if } x/y \geq 0.5 \\ [0.0, 1.0] & \text{otherwise} \end{cases}$$

This function says that an agent adopts a certain behavior if at least half of his incoming neighbors have some property (strong ties, weak ties, meet some requirement of gender, income, etc.) and that we have no information otherwise. In our framework, we can leverage the bounds associated with the influence function to create a "soft" tipping function:

$$sftTp(x,y) = \begin{cases} [0.7, 1.0] & \text{if } x/y \geq 0.5 \\ [0.0, 1.0] & \text{otherwise} \end{cases}$$

Intuitively, the above function says that an agent adopts a behavior with a weight of at least 0.7 *if half of the incoming neighbors that have some attribute and meet some criteria, and we have no information otherwise. Another possibility is to have an influence function that may reduce the weight that an agent adopts a certain behavior:*

$$ngTp(x,y) = \begin{cases} [0.0, 0.2] & \text{if } x = y \\ [0.0, 1.0] & \text{otherwise} \end{cases}$$

The ngTp function says that an agent will adopt a behavior with a weight no greater than 0.2 *if all of the incoming neighbors possessing some property meet some criteria, and that we have no information otherwise.* ■

DEFINITION 2.7 (NEIGHBOR CRITERION). *If* g_{edge}, g_{node} *are non-fluent network formulas (formed over edges and nodes, respectively),* h *is a conjunction of network atoms, and ifl is an influence function, then* $(g_{edge}, g_{node}, h)_{ifl}$ *is a neighbor criterion.*

Formulas g_{node} and h in a neighbor criterion specify the (non-fluent and fluent, respectively) criteria on a given neighbor, while formula g_{edge} specifies the non-fluent criteria on the directed edge from that neighbor to the node in question.

The next component is the "target criteria", which are the criteria that an agent must satisfy in order to be influenced by its neighbors. Ideas such as "susceptibility" [1] can be integrated into our framework via this component. We represent these criteria with a formula of non-fluent network atoms. The final component, the "target", is simply the label of the target agent that is influenced by its neighbors. Hence, we now have all the pieces to define a rule.

DEFINITION 2.8 (RULE). *Given fluent label* L, *natural number* Δt, *target criteria* f *and neighbor criteria* $(g_{edge}, g_{node}, h)_{ifl}$, *a MANCaLog Rule is of the form:*

$$r = L \overset{\Delta t}{\leftarrow} f, (g_{edge}, g_{node}, h)_{ifl}$$

We will use the notation $head(r)$ *to denote* L.

Note that the target (also referred to as the head) of the rule is a single label; essentially, the body of the rule characterizes a set of nodes, and this label is the one that is modified for each node in this set. More specifically, the rule is essentially saying that when certain conditions for an agent and its neighbors are met, the *bnd* bound for the network atom formed with label L on that agent changes. Later, in the semantics, we introduce network interpretations, which map components (nodes and edges) of the network to worlds at a given point in time. The rule dictates how this mapping changes in the next time step.

DEFINITION 2.9 (MANCaLog PROGRAM). *A program* P *is a set of rules, facts, and integrity constraints s.t. each non-fluent fact* $F \in \mathcal{F}_{nf}$ *appears no more than once in the program. Let* $\mathbf{P}$ *be the set of all programs.*

EXAMPLE 2.6. *Following from the previous examples, we can have a MANCaLog program that leverage the sftTp and ngTp influence functions in rules that are more expressive*

Table 2: Example network interpretation, NI_1.

Comp.	*male*	*fem*	*strTie*	*wkTie*	*visPgA*	*visPgB*
1	$[1,1]$	$[0,0]$	-	-	$[0.9,1.0]$	$[0.8,1.0]$
2	$[0,0]$	$[1,1]$	-	-	$[0.0,0.3]$	$[0.0,0.2]$
3	$[1,1]$	$[0,0]$	-	-	$[0.6,1.0]$	$[0.0,0.2]$
4	$[0,0]$	$[1,1]$	-	-	$[0.0,0.2]$	$[0.9,1.0]$
5	$[1,1]$	$[0,0]$	-	-	$[0.0,0.2]$	$[0.7,1.0]$
(1,2)	-	-	$[1,1]$	$[0,0]$	-	-
(2,1)	-	-	$[1,1]$	$[0,0]$	-	-
(1,3)	-	-	$[0,0]$	$[1,1]$	-	-
(2,3)	-	-	$[0,0]$	$[1,1]$	-	-
(3,4)	-	-	$[1,1]$	$[0,0]$	-	-
(4,3)	-	-	$[1,1]$	$[0,0]$	-	-
(4,5)	-	-	$[1,1]$	$[0,0]$	-	-

than previous models. Consider the following rules:

$$R_1 = visPgA \overset{2}{\leftarrow} \langle fem,[1,1]\rangle, (\langle strTie,[0.9,1]\rangle, \mathsf{Tr}, \langle visPgA,[0.9,1.0]\rangle)_{sftTp}$$

$$R_2 = visPgB \overset{1}{\leftarrow} \langle male,[1,1]\rangle, (\mathsf{Tr}, \mathsf{Tr}, \langle visPgB,[0.8,1.0]\rangle)_{sftTp}$$

$$R_3 = visPgA \overset{3}{\leftarrow} \langle male,[1,1]\rangle, (\mathsf{Tr}, \langle fem,[1,1]\rangle, \neg\langle visPgA,[0.7,1.0]\rangle)_{ngTp}$$

Rule R_1 says that a female agent in the network visits page A with a weight of at least 0.7 (this is specified in the sftTp influence function) if at least half of her strong ties (with weight of at least 0.9) visited the page (with a weight of at least 0.9) two days ago. The rest of the rules can be read analogously. ■

We now introduce our first semantic structure: the **network interpretation**.

Definition 2.10 (Network Interpretation). *A network interpretation is a mapping of network components to sets of network atoms, $NI : \mathcal{G} \to 2^{NA}$. We will use* **NI** *to denote the set of all network interpretations.*

We note that not all labels will necessarily apply to all nodes and edges in the network. For instance, certain labels may describe a relationship while others may only describe a property of an individual in the network. If a given label L does not describe a certain component c of the network, then in a valid network interpretation NI, $\langle L,[0,1]\rangle \in NI(c)$.

Example 2.7. *Consider G'_{soc}, the induced subgraph of G_{soc} that has only nodes $\{1,2,3,4,5\}$. Table 2 shows the contents of NI_1, an example network interpretation.* ■

We define a MANCaLog interpretation as follows.

Definition 2.11 (Interpretation). *A MANCaLog interpretation I is a mapping of natural numbers in the interval $[0,t_{max}]$ to network interpretations, i.e., $I : \boldsymbol{N} \to \mathbf{NI}$. Let $\mathcal{I}$ be the set of all possible interpretations.*

2.2 Satisfaction

First, we define what it means for an interpretation to satisfy a fact and a rule.

Definition 2.12 (Fact Satisfaction). *An interpretation I **satisfies** MANCaLog fact $(a,c) : [t_1,t_2]$, written $I \models (a,c) : [t_1,t_2]$, iff $\forall t \in [t_1,t_2]$, $I(t)(c) \models a$.*

Example 2.8. *Consider interpretation I_1, where $I_1(0) = NI_1$ (from Example 2.7), and MANCaLog facts F_7 and F_8 from Example 2.3. In this case, $I_1 \models F_7$ and $I_1 \not\models F_8$.* ■

For non-fluent facts, we introduce the notion of strict satisfaction, which enforces the bound in the interpretation to be set to exactly what the fact dictates.

Definition 2.13 (Strict Fact Satisfaction). *Interpretation I **strictly satisfies** MANCaLog fact $(c,a) : [t_1,t_2]$ iff $\forall t \in [t_1,t_2]$, $a \in I(t)(c)$.*

Next, we define what it means for an interpretation to satisfy an integrity constraint.

Definition 2.14 (IC Satisfaction). *An interpretation I **satisfies** integrity constraint $a \hookleftarrow b$ iff for all $t \in \tau$ and $c \in \mathcal{G}$, $I(t)(c) \models \neg b \vee a$.*

Before we define what it means for an interpretation to satisfy a rule, we require two auxiliary definitions that are used to define the bound enforced on a label by a given rule, and the set of time points that are affected by a rule.

Definition 2.15 (*Bound* function). *For a given rule $r = L \overset{\Delta t}{\leftarrow} f, (g_{edge}, g_{node}, h)_{ifl}$, node v, and network interpretation NI, $Bound(r,v,NI) =$*

$$ifl\Big(\Big|Qual\big(v,g_{edge},g_{node},h,NI\big)\Big|, \Big|Elig\big(v,g_{edge},g_{node},NI\big)\Big|\Big),$$

where $Elig(v,g_{edge},g_{node},NI) =$

$$\Big\{v' \in V \mid NI(v') \models g_{node} \wedge (v',v) \in E \wedge NI\big((v',v)\big) \models g_{edge}\Big\}$$

and $Qual(v,g_{edge},g_{node},h,NI) =$

$$\Big\{v' \in Elig(v,g_{edge},g_{node},NI) \mid NI(v') \models h\Big\}$$

Intuitively, the bound returned by the function depends on the influence function and the number of qualifying and eligible nodes that influence it.

Definition 2.16 (Target Time Set). *For interpretation I, node v, and rule $r = L \overset{\Delta t}{\leftarrow} f, (g_{edge}, g_{node}, h)_{ifl}$, the target time set of I, v, r is defined as follows:*

$$TTS(I,v,r) = \Big\{t \in [0,t_{max}] \mid I(t-\Delta t)(v) \models f\Big\}$$

We also extend this definition to a program P, for a given $c \in \mathcal{G}$ and $L \in \mathcal{L}$, as follows; $TTS(I,c,L,P) =$

$$\bigcup_{r \in P, head(r)=L} TTS(I,c,r) \cup \big\{t \in [t_1,t_2] \mid (\langle L,bnd\rangle,c) : [t_1,t_2] \in P\big\}$$

$$\cup \Big\{t \mid \langle L,bnd\rangle \hookleftarrow b \in P \wedge I(t)(c) \models b\Big\}$$

We can now define satisfaction of a rule by an interpretation.

Definition 2.17. *An interpretation I **satisfies** a rule $r = L \overset{\Delta t}{\leftarrow} f, (g_{edge}, g_{node}, h)_{ifl}$ iff for all $v \in V$ and $t \in TTS(I,v,r)$ it holds that*

$$I(t)(v) \models \Big\langle L, Bound\big(r,v,I(t-\Delta t)\big)\Big\rangle.$$

EXAMPLE 2.9. *Let I_1 be the interpretation from Example 2.8. Suppose that $\langle visPgB, [0.8, 1.0]\rangle \in I(1)(5)$. In this case, $I_1 \models R_2$. Let I_2 be equivalent to I_1 except that we have $\langle visPgB, [0.0, 0.5]\rangle \in I_2(1)(3)$. In this case, $I_2 \not\models R_2$.* ∎

We now define satisfaction of programs, and introduce *canonical interpretations*, in which time points that are not "targets" retain information from the last time step.

DEFINITION 2.18. *For interpretation I and program P:*

I is a **model** *for P iff it satisfies all rules, integrity constraints, and fluent facts in that program, strictly satisfies all non-fluent facts in the program, and for all $L \in \mathcal{L}$, $c \in \mathcal{G}$ and $t \notin TTS(I, c, L, P)$, $\langle L, [0, 1]\rangle \in I(c)(t)$.*

I is a ***canonical model*** *for P iff it satisfies all rules, integrity constraints, and fluent facts in P, strictly satisfies all non-fluent facts in P, and for all $L \in \mathcal{L}$, $c \in \mathcal{G}$, and $t \notin TTS(I, c, L, P)$, $\langle L, [0, 1]\rangle \in I(c)(t)$ when $t = 0$ and $\langle L, bnd\rangle \in I(t)(c)$ where $\langle L, bnd\rangle \in I(t-1)(c)$, otherwise.*

EXAMPLE 2.10. *Following from previous examples, if we consider interpretation I_1 and program $P = \{F_7, R_2\}$, we have that $\langle visPgB, [0.0, 0.2]\rangle$ must be in $I_1(1)(2)$ in order for I_1 to be canonical.* ∎

2.3 Consistency and Entailment

In this section we discuss consistency and entailment in MANCaLog programs, and explore the use of minimal models towards computing answers to these problems.

DEFINITION 2.19 (CONSISTENCY). *A MANCaLog program P is (canonically) consistent iff there exists a (canonical) model I of P.*

DEFINITION 2.20 (ENTAILMENT). *A MANCaLog program P (canonically) entails MANCaLog fact F iff for all (canonical) models I of P, it holds that $I \models F$.*

Now we define an ordering over models and define the concept of minimal model. We then show that if we can find a minimal model then we can answer consistency, entailment, and tight entailment queries. To do so, we first define a pre-order over interpretations.

DEFINITION 2.21 (PREORDER OVER INTERPRETATIONS). *Given interpretations I, I' we say $I \sqsubseteq^{pre} I'$ if and only if for all t, v, L if there exists $\langle L, bnd\rangle \in I(t)(v)$ then there must exist $\langle L, bnd'\rangle \in I'(t)(v)$ s.t. $bnd' \subseteq bnd$.*

Next, we define an equivalence relation for interpretations denoted with $\sim$; we will use the notation $[I]$ for the set of all interpretations equivalent to I w.r.t. $\sim$. This allows us to define a partial ordering.

DEFINITION 2.22. *Two interpretations I, I' are* ***equivalent*** *(written $I \sim I'$) iff for all $P \in \mathbf{P}$, $I \models P$ iff $I' \models P$.*

DEFINITION 2.23 (PARTIAL ORDERING). *Given classes of interpretations $[I], [I']$ that are equivalent w.r.t. $\sim$, we say that $[I]$ precedes $[I']$, written $[I] \sqsubseteq [I']$, iff $I \sqsubseteq^{pre} I'$.*

The partial ordering is clearly reflexive, antisymmetric, and transitive. Note that we will often use $I \sqsubseteq I'$ as shorthand for $[I] \sqsubseteq [I']$. We define two special interpretations, $\bot$ and $\top$, such that $\forall t \in \tau, c \in \mathcal{G}$, $\bot(t)(c) = \emptyset$ and there exists network atom $\langle L, \emptyset\rangle \in \top(t)(c)$. Clearly, no other interpretation can be below $\bot$ as the $[\ell, u]$ bound on all network atoms for each time step and each component is $[0, 1]$; similarly, no other interpretation is above $\top$, since for any interpretation I for which there exists $\langle L, bnd\rangle \in I(t)(c)$ where $bnd \neq \emptyset$, we have $\emptyset \subseteq bnd$. We can prove (see the full version of the paper for details) that with $\top$ and $\bot$, $\langle \mathcal{I}, \sqsubseteq\rangle$ is a complete lattice. We can now arrive at the notion of *minimal model* for a MANCaLog program.

DEFINITION 2.24 (MINIMAL MODEL). *Given program P, the minimal model of P is a (canonical) interpretation I s.t. $I \models P$ and for all (canonical) interpretation I' s.t. $I' \models P$, we have that $I \sqsubseteq I'$.*

Suppose we have some algorithm A that takes as input a program P and returns an interpretation I (where I does not necessarily satisfy P) s.t. for all I' where $I' \models P$, $I \sqsubseteq I'$. It is easy to show that if $A(P) \models P$ then P is consistent. Likewise, if $A(P) = \top$ then P is inconsistent, as all models must then have a tighter weight bound for the network atoms than an invalid interpretation (hence, making such an interpretation invalid as well). Clearly, any such algorithm A would provide a sound and complete answer to the consistency problem. Likewise, if we consider the entailment problem, it is easy to show that for fact $F = (\langle L, bnd\rangle, c) : [t_1, t_2]$, P (canonically) entails F iff the minimal model of P (canonically) satisfies F. This is because for minimal model $A(P)$ of P, for any time $t \in [t_1, t_2]$, if $A(P)(t)(c) \models \langle L, bnd\rangle$ then there is network atom $\langle L, bnd'\rangle \in A(P)(t)(c)$ s.t. $bnd' \subseteq bnd$. We note that for any other interpretation I of P with $\langle L, bnd''\rangle \in I(t)(c)$ we have that $bnd' \supseteq bnd''$. Hence, having a minimal model allows us to solve any entailment query. We can think of a minimal model of a MANCaLog program as the outcome of a diffusion process in a multi-agent system modeled as a complex network. Hence, a question such as "how many agents will adopt the product with a weight of at least 0.9 in two months?" can be easily answered once the minimal model is obtained.

3. FIXED POINT MODEL COMPUTATION

In this section we introduce a fixed-point operator that produces the non-canonical minimal model of a MANCaLog program in polynomial time. This is followed by an algorithm to find a canonical minimal model also in polynomial time. First, we introduce three preliminary definitions.

DEFINITION 3.1. *For a given MANCaLog program P, $c \in \mathcal{G}$, $L \in \mathcal{L}$, and $t \in \tau$ we define function $FBnd(P, c, t, L) =$*

$$\bigcap_{(\langle L, bnd\rangle, c):[t_1,t_2]\in P \text{ s.t. } t\in[t_1,t_2]} bnd$$

DEFINITION 3.2. *For a given MANCaLog program P, $c \in \mathcal{G}$, $L \in \mathcal{L}$, and $t \in \tau$ we define function $IBnd(P, c, t, L) =$*

$$\bigcap_{\langle L, bnd\rangle \leftarrow a \in P \text{ s.t. } I(t)(c) \models a} bnd$$

DEFINITION 3.3. *Given MANCaLog program P, interpretation I, $v \in V$, $L \in \mathcal{L}$, and $t \in \tau$, we define $RBnd(P, I, v, t, L) =$*

$$\bigcap_{r\in P \text{ s.t. } t\in TTS(I,v,L,P)\cap TTS(I,v,r)} Bound(r, v, I(t - \Delta t))$$

We can now introduce the operator.

DEFINITION 3.4 (Γ OPERATOR). *For a given MANCaLog program P, we define the operator $\Gamma_P : \mathcal{I} \to \mathcal{I}$ as follows: For a given I, for each $t \in \tau$, $c \in \mathcal{G}$, and $L \in \mathcal{L}$, add $\langle \mathcal{L}, bnd \rangle$ to $\Gamma_P(I)(t)(c)$ where bnd is defined as:*

$$bnd = bnd_{prv} \cap FBnd(P, c, t, L) \cap IBnd(P, I, c, t, L) \cap RBnd(P, I, c, t, L)$$

where $\langle L, bnd_{prv} \rangle \in I(t)(c)$.

It is easy to show that Γ can be computed in polynomial time (the proof is in the full version). Next, we introduce notation for repeated applications of Γ.

DEFINITION 3.5 (ITERATED APPLICATIONS OF Γ). *Given natural number $i > 0$, interpretation I, and program P, we define $\Gamma_P^i(I)$, the multiple applications of Γ, as follows:*

$$\Gamma_P^i(I) = \begin{cases} \Gamma_P(I) & \text{if } i = 1 \\ \Gamma_P(\Gamma_P^{i-1}(I)) & \text{otherwise} \end{cases}$$

We can prove that the iterated Γ operator converges after a polynomial number of applications:

THEOREM 3.1. *Given interpretation I and program P, there exists a natural number k s.t. $\Gamma_P^k(I) = \Gamma_P^{k+1}(I)$, and*

$$k \in O\Big(|P| \cdot d_*^{in} \cdot t_{max} \cdot |E|\Big)$$

where d_*^{in} is the maximum in-degree in the network.

PROOF (SKETCH). For a given vertex $i \in V$, we will use the notation d_i^{in} to denote the number of incoming neighbors (of any edge type). First note that for a given $t \in \tau, i \in V$, and $L \in \mathcal{L}$, a given rule r can tighten the bound on a network atom formed with L no more than $(d_i^{in}+1) \cdot (d_*^{in}+1)$ times. At each application of Γ, at least one network atom must tighten. Hence, as there are only $O\Big(|P| \cdot d_*^{in} \cdot t_{max} \cdot |E|\Big)$ tightenings possible, this is also the bound on the number of applications of Γ. □

In the following, we will use the notation Γ_P^* to denote the iterated application of Γ after a number of steps sufficient for convergence; Theorem 3.1 means that we can efficiently compute Γ_P^*. We also note that as a single application of Γ can be computed in polynomial time, this implies that we can find a minimal model of a MANCaLog program in polynomial time. We now prove the correctness of the operator. We do this first by proving a key lemma that, when combined with a claim showing that for consistent program P, Γ_P^* is a model of P, tells us that Γ_P^* is a minimal model for P. Following directly from this, we have that P is inconsistent iff $\Gamma_P^* = \top$.

LEMMA 3.2. *If $I \models P$ and $I' \sqsubseteq I$ then $\Gamma(I') \sqsubseteq I$.*

THEOREM 3.3. *If program P is consistent then Γ_P^* is a minimal model for P.*

These results, when taken together, prove that tight entailment and consistency problems for MANCaLog can be solved in polynomial time, which is precisely what we set out to accomplish as part of our desiderata described in Section 1.1. Next, we develop an algorithm for the *canonical* versions of consistency and tight entailment, and show that we can bound the running time of the algorithm with a polynomial. We also note that subsequent runs of the convergence of Γ will likely complete quicker in practice, as the initial interpretation is the last interpretation calculated (cf. line 11). We also show that the interpretation produced by the algorithm is a canonical minimal model. Following from that, a program is inconsistent iff the algorithm returns $\top$.

Algorithm 1 CANON_PROC

Require: Program P
Ensure: Interpretation I

1: $cur\_interp = \Gamma_P^*(\bot)$;
2: Initialize matrix array $cur\_free[\cdot][\cdot]$ where for $v \in V$, and $L \in \mathcal{L}$, $cur\_free[v][L] = \tau - TTS(cur\_interp, v, L, P) - \{0\}$;
3: Initialize array $vl\_pr[\cdot]$ where for each $t \in [1, t_{max}]$, $vl\_pr[t] = \{(v, L) \mid t \in cur\_free[v][L]\}$;
4: **for** $t = 1, \ldots, t_{max}$ **do**
5: **if** $vl\_pr[t] \neq \emptyset$ **then**
6: **for** $(v, L) \in vl\_pr[t]$ **do**
7: Remove $\langle L, bnd \rangle$ from $I(t)(v)$;
8: Let a be the atom in $I(t-1)(v)$ of the form $\langle L, bnd' \rangle$;
9: Add a to $I(t)(v)$;
10: **end for**
11: Set $cur\_interp = \Gamma_P^*(cur\_interp)$;
12: **end if**
13: For $v \in V$, and $L \in \mathcal{L}$, $cur\_free[v][L] = \tau - TTS(cur\_interp, v, L, P) - \{0, \ldots, t\}$
14: For each $t \in [t+1, t_{max}]$, $vl\_pr[t] = \{(v, L) \mid t \in cur\_free[v][L]\}$
15: **end for**
16: **return** I

PROPOSITION 3.1. *Algorithm CANON_PROC performs no more than $1 + t_{max} \cdot \min(|\mathcal{L}|, |P|) \cdot |V|$ calculations of the convergence of Γ.*

THEOREM 3.4. *If P is consistent, then CANON_PROC(P) is the minimal canonical model of P.*

4. APPLICATIONS

In this section, we will briefly discuss work in progress on how MANCaLog can be applied in real world settings.

It is widely acknowledged that modeling influence in multi-agent systems (most usefully modeled as complex networks) is highly desirable for many practical problems as varied as viral marketing, prevention of drug use, vaccination, and power plant failure. Though MANCaLog programs are a rich model to work with, the acquisition of rules is the principal hurdle to overcome; this is mainly due to this richness of representation, since for each rule we must provide a set of conditions on the agents being influenced, conditions on their neighbors and their ties to their neighbors, and how capable these neighbors are of influencing them. A domain expert is likely able to provide important insights into these components, but the best way to obtain these rules is undoubtedly to leverage the presence of large amounts of data in domains like Twitter (with about 340M messages sent per day, available through public APIs), Facebook (over 950M users with more complex information; not publicly available, but data can be requested through apps), and blogging and photo hosting sites such as Blogger and Flickr (which have millions of users as well).

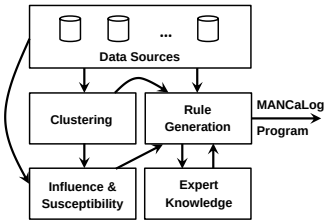

Figure 2: An architecture for obtaining MANCaLog programs from available data sources.

Concretely, we have begun working towards this goal by extracting several time-series, multi-attribute network data sets on which to apply MANCaLog. For instance, to study the proliferation of research on different topics, we looked at research on "niacin" indexed by Thomson Reuters Web of Knowledge (`http://wokinfo.com`). This topic was chosen due to its interest to a variety of disciplines, such as medicine, biology, and chemistry; this gives the data more variety compared with more discipline-specific topics. We extracted an author-paper bipartite network consisting of 3,790 papers with 10,465 authors and 16,722 edges (cf. Figure 3); from this data we can easily focus on various kinds of networks (co-author, citation, etc.). We have also collected attribute and time-series data for this network, as well as the subjects of the papers; the propagation of these subjects is a good starting point to test methods for the acquisition of MANCaLog rules. We are harvesting larger datasets from various online social networks. Further details can be found in the full version of the paper.

A proposed learning architecture. We are currently developing a MANCaLog learning architecture (depicted in Fig. 2) based on the use of state-of-the-art data analysis, clustering, and influence learning techniques as building blocks for the acquisition of MANCaLog rules from data sets. The key question is not just the identification of the best techniques to adopt, but how to adapt them and combine them in such a way as to produce meaningful and useful outputs.

Consider the diagram in Fig. 2: the data first flows from raw data sources to the *cluster identification* component, which has the goal of identifying sets of agents behaving as groups (for instance, teens influencing other teens of the same sex in the consumption of music, or scientists of a certain field influencing the research topics of others in a related field) [16, 9]; the main output here is a set of conditions on nodes and edges that characterize groups of nodes. Once clusters are identified, the *influence recognition* component will make use of both the clusters and the data sources to recognize what kind of influence is present in the system [1, 5, 6]; the main output of this component is the influence function to be used in the MANCaLog rules. The *rule generation* component then takes the output of the cluster identification and influence recognition components, along with the raw data (*e.g.*, to analyze time stamps) and produces MANCaLog rules; the output of this component is involved in a refinement cycle with experts who can provide feedback on the rules being produced (such as possible combinations of rules, identification of cases of overfitting, etc.).

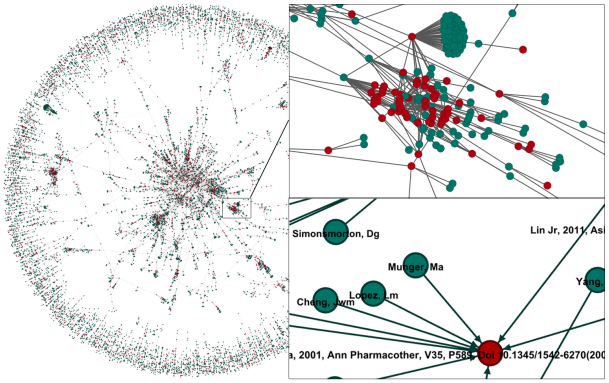

Figure 3: *(Left)* Visualization of a multi-attribute time-series author-paper network from 1952 to 2012. *(Top-Right)* Close-up of the data inside the small box in the main figure. *(Bottom-Right)* Close-up showing node attributes. In all cases, authors are colored green and papers are colored red. Data extracted from Thomson Reuters Web of Knowledge.

5. CONCLUSION

In this paper, we presented the MANCaLog language for modeling cascades in multi-agent systems organized in the form of complex networks. We started by establishing seven criteria in the form of desiderata for such a formalism, and proved that MANCaLog meets all of them; to the best of our knowledge, this has not been accomplished by any previous model in the literature. We also note that MANCaLog is the first language of its kind to consider network structure in the semantics, potentially opening the door for algorithms that leverage features of network topology in more efficiently answering queries. Our current work involves implementing the algorithms described in this paper, as well as the real-world applications described in Section 4; though our algorithms have polynomial time complexity, it is likely that further optimizations will be needed in practice to ensure scalability for very large data sets.

In the near future, we shall also explore various types of queries that have been studied in the literature, such as finding agents of maximum influence, identifying agents that cause a cascade to spread more quickly, and identifying agents that can be influenced in order to halt a cascade.

6. ACKNOWLEDGMENTS

P.S. is supported by the Army Research Office (project 2GDATXR042). G.I.S. is supported under (UK) EPSRC grant EP/J008346/1 – PrOQAW. The opinions in this paper are those of the authors and do not necessarily reflect the opinions of the funders, the U.S. Military Academy, or the U.S. Army.

7. REFERENCES

[1] S. Aral and D. Walker. Identifying Influential and Susceptible Members of Social Networks. *Science*, 337(6092):337–341, 2012.

[2] M. Broecheler, P. Shakarian, and V. Subrahmanian. A scalable framework for modeling competitive diffusion in social networks. In *Proc. of SocialCom*. IEEE, 2010.

[3] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proc. of KDD '10*, pages 1029–1038. ACM, 2010.

[4] A. Goyal, F. Bonchi, and L. Lakshmanan. Discovering leaders from community actions. In *Proc. of CIKM*, pages 499–508. ACM, 2008.

[5] A. Goyal, F. Bonchi, and L. Lakshmanan. Learning influence probabilities in social networks. In *Proc. of WSDM*, pages 241–250. ACM, 2010.

[6] A. Goyal, F. Bonchi, and L. Lakshmanan. A data-based approach to social influence maximization. *Proc. of VLDB*, 5(1):73–84, 2011.

[7] M. Granovetter. The Strength of Weak Ties. *The American Journal of Sociology*, 78(6):1360–1380, 1973.

[8] M. Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.

[9] A. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[10] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. of KDD '03*, pages 137–146. ACM, 2003.

[11] E. Lieberman, C. Hauert, and M. A. Nowak. Evolutionary dynamics on graphs. *Nature*, 433(7023):312–316, 2005.

[12] T. C. Schelling. *Micromotives and Macrobehavior*. W.W. Norton and Co., 1978.

[13] P. Shakarian, A. Parker, G. I. Simari, and V. S. Subrahmanian. Annotated probabilstic temporal logic. *ACM Trans. on Comp. Logic*, 12(2), 2011.

[14] P. Shakarian, V. Subrahmanian, and M. L. Sapino. Using Generalized Annotated Programs to Solve Social Network Optimization Problems. In *Proc. of ICLP (tech. comm.)*, 2010.

[15] V. Sood, T. Antal, and S. Redner. Voter models on heterogeneous networks. *Physical Review E*, 77(4):041121, 2008.

[16] T. Warren Liao. Clustering of time series data–a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

8. APPENDIX

8.1 Set of interpretations form a complete lattice

With top interpretation $\top$ and bottom interpretation $\bot$, $\langle \mathcal{I}, \sqsubseteq \rangle$ is a complete lattice.

PROOF. Let $\mathcal{I}'$ be a subset of $\mathcal{I}$. We can create $inf(\mathcal{I}')$ as follows. We build interpretation I'. For each $t \in \tau, c \in \mathcal{G}, L \in \mathcal{L}$, let ℓ_1 be the least of the set $\cup_{I \in \mathcal{I}'} \{\ell | \langle L, [\ell, u] \rangle \in I(t)(c), \langle L, [\ell, u) \rangle \in I(t)(c)\}$ and ℓ_2 be the least of the set $\cup_{I \in \mathcal{I}'} \{\ell | \langle L, (\ell, u] \rangle \in I(t)(c), \langle L, (\ell, u) \rangle \in I(t)(c)\}$. Then, for each $t \in \tau, c \in \mathcal{G}, L \in \mathcal{L}$ let u_1 be the greatest element of the set $\cup_{I \in \mathcal{I}'} \{u | \langle L, [\ell, u] \rangle \in I(t)(c), \langle L, (\ell, u] \rangle \in I(t)(c)\}$ and u_2 be the greatest of the set $\cup_{I \in \mathcal{I}'} \{u | \langle L, [\ell, u) \rangle \in I(t)(c), \langle L, (\ell, u) \rangle \in I(t)(c)\}$. If there is any interpretation I in $\mathcal{I}$ where there is not some bnd s.t. $\langle L, bnd \rangle \in I(t)(c)$ then add $\langle L, [0, 1] \rangle$ to $I'(t)(c)$. If $\ell_2 \leq \ell_1$ and $u_1 \geq u_2$ then add $\langle L, (\ell_2, u_1] \rangle$ to $I'(t)(c)$. If $\ell_2 \leq \ell_1$ and $u_2 > u_1$ then add $\langle L, (\ell_2, u_2) \rangle$ to $I'(t)(c)$. If $\ell_2 > \ell_1$ and $u_2 > u_1$ then add $\langle L, [\ell_1, u_2) \rangle$ to $I'(t)(c)$. Finally, if $\ell_2 > \ell_1$ and $u_1 \geq u_2$ then add $\langle L, [\ell_1, u_1] \rangle$ to $I'(t)(c)$. Clearly, $I' = inf(\mathcal{I}')$.

In the next part of the proof, we show we can create $sup(\mathcal{I}')$ as follows. We build interpretation I'. For each $t \in \tau, c \in \mathcal{G}, L \in \mathcal{L}$ let ℓ_1 be the greatest of the set $\cup_{I \in \mathcal{I}'} \{\ell | \langle L, [\ell, u] \rangle \in I(t)(c), \langle L, [\ell, u) \rangle \in I(t)(c)\}$ and ℓ_2 be the greatest of the set $\cup_{I \in \mathcal{I}'} \{\ell | \langle L, (\ell, u] \rangle \in I(t)(c), \langle L, (\ell, u) \rangle \in I(t)(c)\}$. Then, for each $t \in \tau, c \in \mathcal{G}, L \in \mathcal{L}$ let u_1 be the least element of the set $\cup_{I \in \mathcal{I}'} \{u | \langle L, [\ell, u] \rangle \in I(t)(c), \langle L, (\ell, u] \rangle \in I(t)(c)\}$ and u_2 be the least of the set $\cup_{I \in \mathcal{I}'} \{u | \langle L, [\ell, u) \rangle \in I(t)(c), \langle L, (\ell, u) \rangle \in I(t)(c)\}$. If $\max(\ell_1, \ell_2) > \min(u_1, u_2)$ or $(\ell_2 > \ell_1) \wedge (u_2 < u_1) \wedge (\ell_2 = u_2)$ then add $\langle L, \emptyset \rangle$ to $I'(t)(c)$. If $\ell_2 > \ell_1$ and $u_1 \leq u_2$ then add $\langle L, (\ell_2, u_1] \rangle$ to $I'(t)(c)$. If $\ell_2 > \ell_1$ and $u_2 < u_1$ then add $\langle L, (\ell_2, u_2) \rangle$ to $I'(t)(c)$. If $\ell_2 \leq \ell_1$ and $u_2 < u_1$ then add $\langle L, [\ell_1, u_2) \rangle$ to $I'(t)(c)$. Finally, if $\ell_2 \leq \ell_1$ and $u_1 \leq u_2$ then add $\langle L, [\ell_1, u_1] \rangle$ to $I'(t)(c)$. Clearly, $I' = sup(\mathcal{I}')$.

As both $inf(\mathcal{I}')$ and $sup(\mathcal{I}')$ exist and are clearly in $\mathcal{I}$ then the statement follows. □

8.2 A single application of Γ can be computed in polynomial time

For interpretation I, $\Gamma(I)$ can be computed by conducting $O(|P| \cdot |V| \cdot t_{max} \cdot d_*^{in})$ satisfaction checks where d_*^{in} is the maximum in-degree of a node in the network. (This combined with the assumption that the influence function is computed in constant time results in polynomial time computation for a single application of Γ.)

PROOF. We note that a given rule will require the most satisfaction checks, as a rule will potentially affect a network atom of a certain label for each vertex-time point pair. By the definition of $RBnd$, a given rule clearly causes no more than $O(d_*^{in})$ satisfaction checks. As the number of rules is no more than $|P|$, the statement follows. □

8.3 Proof of Theorem 3.1

Given interpretation I and program P, there exists a natural number k s.t. $\Gamma_P^k(I) = \Gamma_P^{k+1}(I)$, and

$$k \in O\Big(|P| \cdot d_*^{in} \cdot t_{max} \cdot |E|\Big)$$

where d_*^{in} is the maximum in-degree in the network.

PROOF. For a given vertex $i \in V$, we will use the notation d_i^{in} to denote the number of incoming neighbors (of any edge type) and $d_*^{in} = \max_i d_i^{in}$. First we show that for a given $t \in \tau, i \in V$, and $L \in \mathcal{L}$, a given rule r can tighten the bound on a network atom formed with L no more than $(d_i^{in} + 1) \cdot (d_*^{in} + 1)$ times. This is because a given rule adjusts the bound on a network atom based on the number of eligible and qualifying neighbors, which are bounded by d_i^{in}, d_*^{in} respectively. At each application of Γ, at least one network atom must tighten. Hence, as there are only $O\Big(|P| \cdot d_*^{in} \cdot t_{max} \cdot \sum_i d_i^{in}\Big) = O\Big(|P| \cdot d_*^{in} \cdot t_{max} \cdot |E|\Big)$ tightenings possible, this is also the bound on the number of applications of Γ. □

8.4 Proof of Lemma 3.2

If $I \models P$ and $I' \sqsubseteq I$ then $\Gamma(I') \sqsubseteq I$.

PROOF. Suppose, BWOC, that $\Gamma(I') \sqsupset I$. Then, there exists some $L \in \mathcal{L}$, $t \in \tau$ and $c \in \mathcal{G}$ s.t. $\langle L, bnd\rangle \in I(t)(c)$, $\langle L, bnd'\rangle \in I'(t)(c)$, and $\langle L, bnd''\rangle \in \Gamma(I')(t)(c)$ s.t. $bnd \supset bnd''$ and $bnd' \supseteq bnd''$. There are four things that affect bnd'': facts, rules, integrity constraints and bnd'. Clearly, we need not consider the effect that either facts or bnd' have on bnd'', as I satisfies all facts and $I' \sqsubseteq I$. We also note that a given integrity constraint imposed by Definition 3.2 can tighten bnd'' no more than the associated bound in any model. Hence, there must be some rule $r = L \overset{\Delta t}{\Leftarrow} f, (g_{edge}, g_{node}, h)_{ifl}$ that causes bnd'' to become less than bnd. As $bnd'' \neq bnd'$, we know that $t \in TTS(\Gamma(I'), c, r) \cap TTS(I', c, r)$. As a result, we have $\Gamma(I')(t - \Delta t)(c) \models f$ and $I'(t - \Delta t)(c) \models f$. Further, as $I \models P, I' \sqsubseteq I$, and no rule can modify a non-fluent atom, we have

$$|Elig(v, g_{edge}, g_{node}, \Gamma(I')(t - \Delta t)| =$$
$$|Elig(v, g_{edge}, g_{node}, I'(t - \Delta t)| =$$
$$|Elig(v, g_{edge}, g_{node}, \Gamma(I')(t - \Delta t)|.$$

Further, we know that as $I' \sqsubseteq I$, it must be the case that

$$|Qual(v, g_{edge}, g_{node}, h, I(t - \Delta t))| \geq$$
$$|Qual(v, g_{edge}, g_{node}, h, I'(t - \Delta t))|.$$

This implies, by Axiom 2 that, $Bound(r, v, I(t - \Delta t)) \subseteq Bound(r, v, I'(t - \Delta t))$. This then implies that $bnd \subseteq bnd''$, which is a contradiction. □

8.5 Proof of Theorem 3.3

Γ_P^* is a minimal model for P.

PROOF. Claim: If program P is consistent then Γ_P^* is a model of P.
Suppose, BWOC, that there is a fact in P that Γ_P^* does not satisfy. However, by the definition of Γ and the definition of a fact, Γ_P^* must satisfy all facts as the bound on the weight associated with each fact is included in the intersection. Further, we can also see by the definition of Γ that Γ_P^* strictly satisfies all non-fluent facts in P. We also note that the final application of the Γ operator ensures that all integrity constraints are satisfied by Γ_P^*. Now, Suppose, BWOC, that there is a rule in P that Γ_P^* does not satisfy. However, with each application of Γ, for each rule, we include the bound on the weight returned by the *Bound* function for each time step in the target time step associated with that rule. As Γ is applied to convergence, and new bounds are intersected with each application, then we know that all time points in any associated target time set are considered in the intersection.
Proof of Theorem: The above claim tells us that $\Gamma_P^* \models P$. Now consider interpretation I s.t. $I \models P$. As $\bot \sqsubseteq I$, multiple applications of Lemma 3.2 tell us that $\Gamma_P^* \sqsubseteq I$. Hence, the statement follows. □

8.6 Proof of Theorem 3.4

If P is consistent, then CANON_PROC(P) is the minimal canonical model of P.

PROOF. CLAIM 1: If P is consistent, then CANON_PROC(P) is a canonical model of P.
Clearly, I = CANON_PROC(P) satisfies all facts and integrity constraints in P. Hence, we shall consider programs that only consist of rules in this proof. We say I L-canonically satisfies P iff I canonically satisfies $\{r \in P \mid head(r) = L\}$. Clearly, I canonically satisfies P if for all $L \in \mathcal{L}$, P L-canonically satisfies by I. We say that I is an (L, c, q)-canonically consistent interpretation if for $c \in \mathcal{G}$, for the first $t \in \tau - TTS(I, c, L, P) - \{0\}$, $I(t)(c) \models \langle L, bnd\rangle$ where $\langle L, bnd\rangle \in I(t-1)(c)$. Consider some $L \in \mathcal{L}$ and $c \in \mathcal{G}$. Clearly, I is an $(L, c, 0)$-model for P. Let us assume, for some value q, that I is an $(L, c, q-1)$ model for P. Let time point t be the q-th element of $\tau - TTS(I, c, L, P) - \{0\}$. Consider the time step before time t is considered in the for-loop at line 4 of CANON_PROC, which causes the condition at line 5 to be true. By line 13, $\tau - TTS(I, c, L, P) - \{0\} \subseteq cur\_free[c][L]$. This means that t is a member of both. Hence, when t is considered at line 4, the condition at line 5 is true, causing $\langle L, bnd\rangle \in I(t)(c) \cap I(t-1)(c)$ and as the element $\langle L, bnd\rangle \in I(t-1)(c)$ is not changed here, we have shown the I is an (L, c, q)-model for P. By the for-loop at line 6, for all $L \in \mathcal{L}$ and $c \in \mathcal{G}$, I is an (L, c, q)-model for P. Hence, at the for-loop at line 4, we can be assured that for $L \in \mathcal{L}$ and $c \in \mathcal{G}$ that I $(L, c, |\tau - TTS(I, c, L, P) - \{0\}|)$ satisfies P – which means that I canonically satisfies P
CLAIM 2: If I is a canonical model for P,
$cur\_interp \sqsubseteq I$ is an interpretation that also strictly satisfies all non-fluent facts in P, and cur_interp' is cur_interp after being manipulated in lines 6-10 of CANON_PROC, then $cur\_interp' \sqsubseteq I$.

We note that by the definition of satisfaction of a non-fluent fact, and the fact that both cur_interp and I must strictly satisfy all non-fluent facts in P, we know that for all $c \in \mathcal{G}$ and $L \in \mathcal{L}$ that:

$$\begin{aligned} TTS(I, c, L, P) &= TTS(cur\_interp, c, L, P) \\ &= TTS(cur\_interp', c, L, P) \end{aligned}$$

Let us assume that lines 6-10 of the algorithm are changing cur_interp when the outer loop is considering time t and that the condition at line 5 is true. Clearly,

$$t \in \tau - TTS(I, v', L', P) - \{0\}$$

As a result, for any (v, L) pair considered at this point by the algorithm, if $\langle L, bnd\rangle \in I(t)(v)$ and $\langle L, bnd'\rangle \in I(t-1)(v)$ then we have $bnd = bnd'$. By the algorithm, if we have $\langle L, bnd^*\rangle \in cur\_interp'(t)(v)$ and $\langle L, bnd^{**}\rangle \in cur\_interp'(t-1)(v)$ we have that $bnd^* = bnd^{**}$. As $\langle L, bnd^{**}\rangle \in cur\_interp(t-1)(v)$, we know that $bnd' \subseteq bnd^{**}$. As a result, we have $cur\_interp' \sqsubseteq I$, completing the claim.

Proof of theorem: As initially $cur\_interp = \Gamma_P^*$ and $\Gamma_P^* \sqsubseteq I$ by Theorem 3.3, we note that the algorithm changes *cur_interp* either by applying Γ or manipulating it in lines 6-10, which tells us (by claim 2) that for all models I of P that $\mathsf{CANON\_PROC}(P) \sqsubseteq I$. Since by claim 1 we know that $\mathsf{CANON\_PROC}(P) \models P$, the statement of the theorem follows. □

8.7 Details on the Extracted Dataset

One way in which MANCaLog can be used is looking at proliferation of research on different topics. We look at research conducted on niacin, an organic compound commonly used for increasing levels of high density lipoproteins (HDL). Using Thomson Reuters Web of Knowledge (`http://wokinfo.com`) we were able to extract information on 4,202 articles about niacin. This information was then processed using the Science of Science (Sci^2) Tool (`http://sci2.cns.iu.edu`) to extract numerous different networks such as author by paper networks, citation networks, and paper by subject networks. Each paper has attributes about when it was published, what journal it was published in, and what subjects the paper was about. During the first time period there is a total of 508 papers with 856 different authors and 1,231 connections based on an author being cited as an author of a given paper. During the second time period, there is a total of 3,790 papers with 10,465 different authors and 16,772 connections.