Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

11-2016

# Designing minimal effective normative systems with the help of lightweight formal methods

Jianye HAO

Eunsuk KANG

Jun SUN
*Singapore Management University*, junsun@smu.edu.sg

Daniel JACKSON

## Citation

HAO, Jianye; KANG, Eunsuk; SUN, Jun; and JACKSON, Daniel. Designing minimal effective normative systems with the help of lightweight formal methods. (2016). *FSE 2016: Proceedings of the 24th ACM SIGSOFT Symposium on the Foundations of Software Engineering, Seattle, November 13-18, 2016*. 50-60. Research Collection School Of Information Systems.
**Available at:** https://ink.library.smu.edu.sg/sis_research/4947

# Designing Minimal Effective Normative Systems with the Help of Lightweight Formal Methods

Jianye Hao[1], Eunsuk Kang[2], Jun Sun[3], Daniel Jackson[4]
[1]School of Computer Software, Tianjin University, China
[2] University of California, Berkeley, USA
[3]ISTD Pillar, Singapore University of Technology and Design, Singapore
[4]CSAIL, Massachusetts Institute of Technology, USA

## ABSTRACT

Normative systems (i.e., a set of rules) are an important approach to achieving effective coordination among (often an arbitrary number of) agents in multiagent systems. A normative system should be *effective* in ensuring the satisfaction of a desirable system property, and *minimal* (i.e., not containing norms that unnecessarily overconstrain the behaviors of agents). Designing or even automatically synthesizing minimal effective normative systems is highly nontrivial. Previous attempts on synthesizing such systems through simulations often fail to generate normative systems which are both minimal and effective. In this work, we propose a framework that facilitates designing of minimal effective normative systems using lightweight formal methods. Given a minimal effective normative system which coordinates many agents must be minimal and effective for a small number of agents, we start with automatically synthesizing one such system with a few agents. We then increase the number of agents so as to check whether the same design remains minimal and effective. If it is, we manually establish an induction proof so as to lift the design to an arbitrary number of agents. We show the effectiveness of the framework by using it to design road junction traffic rules and population protocols. The protocols designed using our framework are shown to be as good as those manually designed and published ones.

## CCS Concepts

•**Security and privacy** → **Logic and verification;** •**Theory of computation** → *Branch-and-bound;*

## Keywords

Lightweight formal methods, Minimal effective, Norm Synthesis

## 1. INTRODUCTION

Normative systems have garnered much attention in the multiagent systems (MAS) literature as an effective technique for regulating the behaviors of agents [28, 15, 24, 13]. Roughly, a normative system is a set of rules or *norms* imposed on an MAS to ensure that a desirable global property is satisfied. Each norm constrains the behaviors of agents by forbidding them from performing one or more actions under certain circumstances. For example, to avoid collision, two autonomous trains traveling through a common tunnel may implement a norm of "if another train is observed inside the tunnel, the action of move should not be allowed".

This paradigm of normative system in MAS naturally maps to the law system implemented in our human system. The normative system paradigm was first proposed by Shoham and Tennenholtz [28, 29], and the problem of *norm synthesis* has been a major research topic since then. Van der Hoek *et al.* [15] identified three major computational problems related to normative systems: effectiveness, feasibility and synthesis. In order to avoid over-regulate the behaviors of agents, Morales *et al.* [24, 25] later proposed that a desirable normative system should be both effective and *necessary*

Designing or synthesizing normative systems which are both effective and necessary are highly non-trivial. For instance, Van der Hoek *et al.* [15] proposed to solve the computational problems related to normative systems, i.e., effectiveness, feasibility and synthesis, in the context of an Alternating-time Temporal Logic (ATL) [4] model checking problem. Their framework may, however, synthesize unnecessary norms that over-regulate the behaviors of agents. One particular challenge in relying on model checking is that normative systems are often designed for a large or arbitrary number of agents, which often result in state-space explosion. Morales *et al.* [24, 25] developed a mechanism to automatically synthesize normative systems that are "effective and necessary". However, effectiveness and necessity are determined in their scheme using simulations, and there is thus no guarantee that the normative system synthesized is effective or necessary.

In this work, we show that the problem can be solved, to certain extent, with techniques developed in the software engineering community. In particular, we propose a framework which aims to facilitate designing *minimal effective* normative systems based on lightweight formal methods [17]. One simple observation is that given a homogenous system, if a normative system is minimal effective for a large or arbitrary number of agents, it is highly likely to hold for a small number of agents as well. This observation matches the underlying principle of lightweight formal methods [17] and automated deduction philosophy [31] in software engineering. Based on the observation, we propose a framework which starts with automatically searching for candidate effective and necessary normative systems for an MAS with as few as 2 agents. Next, we check if the candidate normative systems (or any specialization of them) remain effective and necessary with an increasing number of agents. We keep eliminating candidates until the candidates stabilize. The effectiveness of a candidate normative system is automatically checked using an analysis engine (e.g., Alloy Analyzer [18] or other model checkers) that exhaustively ex-

*FSE'16*, November 13–18, 2016, Seattle, WA, USA
© 2016 ACM. 978-1-4503-4218-6/16/11...$15.00
http://dx.doi.org/10.1145/2950290.2950307

plores all possible behaviors of the MAS. *Because we work with a small number of agents, scalability is less an issue in our framework.* Once a minimal effective normative system is identified and withstands with an increasing number of agents, we then manually show, often with an induction proof, that the same normative system can be applied to an arbitrary number of agents and remain *minimal effective* in satisfying the property.

To demonstrate that we can design non-trivial normative systems this way, we apply our framework to a road junction example which is a typical example (which is often the only example [28, 29, 24, 25]) used to showcase normative system synthesis methods in MAS literature. In addition, we apply our framework to design a number of population protocols, which are a group of self-regulating network protocols designed for agent-like network nodes. We show that the protocols we synthesis are as good as those manually designed and published ones [7, 21, 6].

The remainders of the paper is organized as follows. We start with an overview of related work in Section 2, and then introduce the background and problem in Section 3. Our norm synthesis framework is described in Section 4, and then are evaluated using three representative case studies in Section 5. Section 6 concludes the paper and points out future directions.

## 2. RELATED WORK

Shoham and Tennenholtz [28, 29] first defined the problem of synthesizing useful social laws in multi-agent systems, and investigated the required properties of the synthesis process and its computational complexity. They argued that a social law should not restrict too much of the individual agents' freedom, and also restrictive enough such that the agents can cooperate towards desirable system goals. Specifically the authors defined the notion of useful social laws, which guarantee that each agent can come up with a plan ensuring him to move from any one focal state to another focal state. They showed that deriving useful social laws in general is NP-complete, and identified conditions under which the problem of deriving useful social laws can be solved in a polynomial time.

Later this idea was generalized in [15]. It was shown that the objective of social laws can be expressed as an Alternating-time Temporal Logic (ATL) [5] formula, and the social law problem is transformed to an ATL model-checking problem. In their framework, the system is modeled as an Action-based Alternating Transition System (AATS) and a social law consists of two components: the objective of the system and the behavioral constraints. The authors identified three major computational problems related to social laws, i.e., effectiveness, feasibility and synthesis, and investigated the corresponding computational complexities. The authors showed that the complexity of the feasibility problem is no more complex than the complexity of the corresponding problem defined by Shoham and Tennenholtz [28], which is NP-complete. Christelis and Rovatsos [8] proposed a novel norm synthesis approach using traditional AI planning techniques to synthesize effective norms in the planning domains. However, no notion of optimality of a social law is considered in both of the work and an effective social law may over regulate the behaviors of the agents.

The problem of synthesizing effective social laws was further refined in [2] which takes into consideration the fact that implementing social laws may have cost, and that the system designer may have multiple goals of varying value. In their framework, the system is modeled using Kripke structures and the desirable objectives are expressed using Computational Tree Logic (CTL). Social laws are modeled as restrictions on Kripke structures, and the costs on edges are used to model the cost of implementing different norms. The utility of a social law is defined as the benefit of the desirable objective brought by this law minus the cost of implementing it. The optimal social law is the one that owns the highest utility. The authors solved the problem of designing an optimal social law by formulating it as an integer linear program and also investigated possible ways of reducing the computational complexity. A common problem of the above approaches are that due to the state space explosion problem which is often observed when model checking techniques are employed, these methods are often limited to synthesize MAS for a small number of agents.

Morales *et al.* [24, 25] proposed that a normative system should not only be effective in guaranteeing the global property, but also avoid over-regulating the behaviors of the agents. They developed a simulation-based mechanism to synthesize both effective and necessary normative systems, and applied their mechanism to the road junction example to illustrate its effectiveness. However, their approach does not provide a theoretical guarantee that the synthesized normative system is effective and minimal. Furthermore, their approach may result in non-effective normative systems that cannot be detected through extensive simulation. In Section 5, we provide a more detailed comparison of our approach against theirs using the same benchmark system.

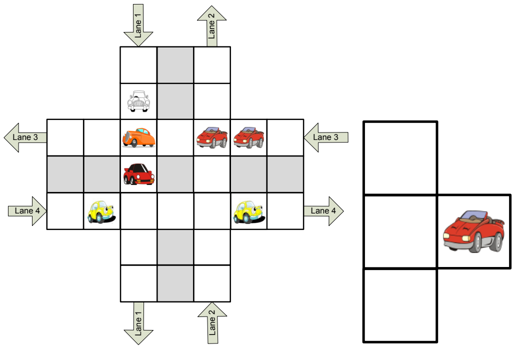
The work in [1] is remotely related to ours. Ågotnes *et al.* [1] proposed the notion of conservative social laws, which are those effective social laws making minimal degree of changes to the original system based on the criterion of the distance metric. The concept of conservative social laws is to model the least change principle inspired from social laws in human society, where the laws with minimal change/effect on people's habits would be easily accepted by the public. While this idea of minimal social laws has been explored previously, most efforts have been focused on exploring the theoretical boundaries of this notion and not on designing practical algorithms for synthesizing minimal norms.

Compared to the above work, our approach is complementary as we not only formally define minimal effective normative systems, but also provide a practical algorithm based on lightweight formal methods to automatically synthesize them. Wamberto *et al.* [30] proposed an approach using first-order constraint solving techniques for norm conflicts detection and resolution in a normative system, however, their goal is simply to resolve conflicts among norms, i.e., there is no global properties to be satisfied or the notion of minimality and effectiveness of norms.

This work is also related to work using formal method based synthesis techniques to solve goal operationalisation problem in requirement engineering [22, 3]. In [14], Degiovanni *et al.* proposed an approach using interpolation and SAT solving techniques to automatically computes required preconditions and required triggering conditions for operations such that the resulting operations establish the (safety) goals. Their idea of iteratively refining operational specifications based on the counterexamples from the model checker is similar to our iterative approach of synthesizing norms from small-size MAS. However, our work is different from theirs as we also consider the minimality requirement of norms. Specifically, their overall idea of iteratively refining the operations based on counter-examples is similar to the first phase (initial norm synthesis) in our approach. In the second phase (norm refinement) of our approach, we seeks to find the minimal normative system through iterative verification and heuristic search based on the results from the first phase.

## 3. PROBLEM DEFINITION

In this section, We define our problem formally.



(a) (b)

**Figure 1: Traffic Network Example**

**Normative Systems** A *multi-agent system* (MAS) is composed of a set $G$ of interacting agents, where each agent $i \in G$ can perform actions from a finite, non-empty set $A_i$ of actions. Formally, a multi-agent system $\mathcal{M}$ is a tuple $\mathcal{M} = \langle S, G, \{A_i\}, R, \Psi, \pi \rangle$, where $S$ is a finite set of global states; $G$ is a finite set of agents; $A_i$ for each $i \in G$ is a finite set of actions of agent $i$; $\Psi$ is a finite set of atomic propositions; and

- $R : S \times J_G \to S$ is a partial system transition function, which defines how the system state changes given the set of actions that the agents choose to perform. Here, $J_G$ is the set of joint actions of the set G of agents, i.e., $J_G = \Pi_{i \in G} A_i$.
- $\pi : S \to 2^{\Psi}$ is an interpretation function, which maps each system state to the set of primitive propositions that hold in that state. That is, $p \in \pi(q)$ where $q \in S$ denotes that the atomic proposition $p$ is true in state $q$.

In this work, we assume that the agents share the same set of actions, i.e., $A_i = A_j$ for any agent $i \in G$. We do not distinguish the identifies of agents and thus the system can be considered as being agent-symmetric. Theoretically, this assumption can be lifted.

*Example* We use a traffic cross system as a running example to explain each step of our approach. As shown in Figure 1(a), a traffic network is composed of two orthogonal, intersecting roads. Each road has two 7-cell lanes, and the direction of each lane is indicated by the arrows in the figure. Each car travels in the indicated direction, and stays within the lane. However, once it arrives at the junction, it is free to switch to any one of the four lanes. We assume that cars have the same speed, and can perform one of two possible actions; move one cell in the indicated direction or stay in the same cell (i.e., $A = \{\mathsf{Move}, \mathsf{Stop}\}$). At any given state, a car is described to be in one of four orientations, depending on the direction that is is headed (east, west, north, and south). We assume that each car can observe the orientation of cars in three of its neighboring cells, as shown in Figure 1(b): top-left, top, and top-right cells.

Given an MAS $\mathcal{M}$, we can define a set of desirable states in $C$ as a property $\omega$, which can be represented as a *propositional logic formula* constituted by propositions in $\Psi$. For example, in the traffic cross system, one desirable property is the *non-collision* property: the system never enters a state in which one or more cars collide. One way to make sure that the system remains in a desirable state is to regulate the actions of the agents using a set of norms. A norm is a constraint on the actions that an agent is not allowed to perform under certain condition. Formally,

Definition 1. *A norm is a tuple $\langle \psi, A' \rangle$, where $\psi$ is the precondition of the norm, and $A' \subseteq A$ is the subset of actions that an agent is forbidden to perform when $\psi$ is satisfied.*

A norm is enforced on an agent only under certain circumstances, which are characterized by the *precondition* of the norm. In practice, each individual agent may not observe the global state of the system due to physical or communication constraints. Thus $\psi$ should not refer to the global state of the system. Instead, we assume that each agent $g$ can observe the states of agents within its neighborhood. The neighborhood of an agent is problem-specific and the neighborhood of a car agent in the running example is illustrated in Figure 1(b). We define the *local view* $\rho_g(s)$ of a global state $s$ from the perspective of agent $g$ as the set of predicates that are evaluated over the neighboring agents that agent $g$ can observe. We assume that the local view of an agent is consistent with the global state of the system. Thus, the norm $n = \langle \psi, A' \rangle$ is enforced on an agent $g$ whenever the agent's local view satisfies the precondition $\psi$, which is formally denoted as $\rho_g(s) \Rightarrow \psi$.

In the following, we present the exact syntax for norms, as shown in Figure 2. Without loss of generality, we restrict the set $A'$ to be a singleton, which is the action that the agent is forbidden to perform under the norm. Note that norms consisting of multiple actions can be naturally represented as multiple norms of single action. The precondition $\psi$ is a conjunction of atomic predicates that describe the characteristics of a single agent or relationship between multiple agents. Every $n$-ary predicate, $p^n \in P$, is applied over $n$ terms, each of which represents a particular agent, an integer value, or a user-defined symbol. In our running example, consider a norm $n = \langle \psi, \{\mathsf{Move}\} \rangle$, where $\psi$ is defined as: $\mathsf{dirNorth}(g) \wedge \mathsf{topLeft}(h, g) \wedge \mathsf{dirEast}(h)$ where $g$ and $h$ are variables representing agents. Enforcing this norm on all agents means that any car $g$ is not allowed to move if it is north-heading (i.e., $\mathsf{dirNorth}(g)$ is true), and if there is another car $h$ which is located diagonally left of it (i.e., $\mathsf{topLeft}(h, g)$ is true) and heading towards east (i.e., $\mathsf{dirEast}(h)$ is true).

Following previous work [28, 24], a *normative system* is defined as a set of norms. Given an MAS $\mathcal{M}$, the system designer may apply a normative system to regulate the behaviors of agents. When a normative system $\Phi$ is applied to an MAS, each norm in $\Phi$ is applied on all agents in $\mathcal{M}$ (a.k.a. all agents are norm-abiding). We write $\mathcal{M} \oplus \Phi$ to denote the system which is the result of implementing the normative system $\Phi$ on $\mathcal{M}$ (i.e., by ruling out all actions forbidden by the normative system). Formally,

Definition 2. *Let $\mathcal{M} = \langle S, G, \{A_i\}, R, \Psi, \pi \rangle$ be an MAS and $\Phi = \{\langle \psi_k, A'_k \rangle\}$ be a normative system. The implementation of $\Phi$ over $\mathcal{M}$, written as $\mathcal{M} \oplus \Phi$, is a Kripke structure $(S', S_0, R', \Psi, L')$ such that $S' \subseteq S$ is a set of states; $S_0$ is the initial state ; $L' : S' \to 2^{\Psi}$ such that $L'(s) = \pi(s)$ for all $s \in S'$ and $R'$ satisfies the following: $(s, s') \in R$ iff $(s, \overrightarrow{a}, s') \in R$ where $\overrightarrow{a} = \Pi_{i \in G} A_i$ and for all $k$, $\overrightarrow{a}_i \notin A'_k$ for all $i$ if $s$ satisfies $\psi_k$.*

The problem of determining whether $\mathcal{M} \oplus \Phi$ satisfies the property $\omega$ can be expressed as a Computation Tree Logic (CTL) [10] satisfiability problem as follows.

$$\mathcal{M} \oplus \Phi \models \mathbf{A}\Box\omega \tag{1}$$

where **A** means "along all paths" and $\Box$ is the temporal operator meaning "now and forever more". We skip the formal definition of $\models$ as it is standard [11]. Intuitively, the above is satisfied if the system $\mathcal{M}$ always remains at a desirable state if the agents behave according to the norms. A normative system $\Phi$ (for an MAS $\mathcal{M}$) that satisfies Equation 1 is called effective. For example, in the

$$
\begin{array}{lcll}
n & ::= & \langle \psi, A' \rangle & \text{// norm} \\
\psi & ::= & P \mid \psi \wedge \psi & \text{// formula} \\
P & ::= & p^n(t_1, t_2, ..., t_n) & \text{// predicate} \\
T & ::= & G \mid \mathsf{Int} \mid \mathsf{Symbol} & \text{// term} \\
G & ::= & g_1 \mid g_2 \mid ... \mid g_n & \text{// agent} \\
A' & ::= & a_1 \mid a_2 \mid ... \mid a_{|A|} & \text{// action}
\end{array}
$$

**Figure 2: Syntax for norms**

traffic cross system, given the *non-collision* property, a normative system is effective if it never enters a state in which cars collide. Note that in this case, this is equivalent with the conventional safety property in model checking. However, in general, effectiveness may also cover any other non-safety properties.

**Minimal Effective Normative System** A normative system can be effective but useless. For instance, in the traffic cross system, a trivially effective normative system would be one which disallows car movement altogether. Thus, in this work, we are interested in effective normative systems which are also minimally constraining (hereafter *minimal*). A normative system that is effective but not minimal over-constrains the behavior of the agents by disallowing innocuous actions, and represents a non-optimal design. We remark due to the minimality requirement, liveness properties are often irrelevant since actions are enabled as long as possible.

Before defining minimal effective normative systems, we first introduce the notion of *specialization* relation to define what it means for a norm to be a specialization of another. Intuitively, if norm $n_1$ specializes norm $n_2$, $n_1$ is less restrictive than $n_2$ (i.e., $n_1$ puts less restrictions on agents than $n_2$). This is achieved by strengthening the precondition of norm $n_1$ to make its applicability more restrictive than that of norm $n_2$. We do not need to consider relaxing the action part in the norms, since we define that each norm contains only one forbidden action and thus it cannot be further relaxed. Formally, the *strict specialization* relation between any pair of norms $(n_1, n_2)$ is defined as follows.

Definition 3. *Let $n_1 = \langle \psi_1, Ac_1 \rangle$ and $n_2 = \langle \psi_2, Ac_2 \rangle$ be a pair of norms. Norm $n_1$ strictly specializes norm $n_2$ iff* (1) $\psi_1 \Rightarrow \psi_2 \wedge \neg(\psi_2 \Rightarrow \psi_1)$ *and* (2) $Ac_1 = Ac_2$.

For example, consider a pair of norms $n = \langle \psi, \{\mathsf{Move}\} \rangle$, and $n' = \langle \psi', \{\mathsf{Move}\} \rangle$ in the traffic cross system, where

$$
\begin{array}{lcl}
\psi & = & \mathsf{dirNorth}(g) \wedge \mathsf{topLeft}(h, g) \wedge \mathsf{dirEast}(h) \\
\psi' & = & \mathsf{dirNorth}(g) \wedge \mathsf{topLeft}(h, g) \wedge \mathsf{dirEast}(h) \wedge \mathsf{top}(h', g)
\end{array}
$$

We can see norm $n'$ strictly specializes $n$, since $n'$ is applicable in a more restricted set of circumstances than $n$ is (namely, $n'$ requires the presence of a third car $h'$ front of $g$). We can similarly define the *weak specialization* relation.

Definition 4. *Let $n_1 = \langle \psi_1, Ac_1 \rangle$ and $n_2 = \langle \psi_2, Ac_2 \rangle$ be a pair of norms. Norm $n_1$ weakly specializes norm $n_2$ iff the following are satisfied:* (1) $\psi_1 \Rightarrow \psi_2$, (2) $Ac_1 = Ac_2$.

If a norm $n$ strictly specializes another norm $n'$, we can equivalently say $n'$ *strictly generalizes* $n$. Similarly, if a norm $n$ weakly specializes another norm $n'$, we say $n'$ *weakly generalizes* $n$. Next we define the *specialization* relation between two normative systems based on the specialization relation between norms. We remark that we assume that there are no duplicated norms, i.e., no multiple semantically equivalent norms coexist in the same normative system. Intuitively, if the normative system $\Phi_1$ specializes another normative system $\Phi_2$, $\Phi_1$ puts less constraints on the behaviors of the agents in the system than $\Phi_2$.

Definition 5. *Let $\Phi_1$ and $\Phi_2$ be two normative systems. $\Phi_1$ strictly specializes $\Phi_2$ iff the following two conditions are satisfied: (1) for each norm $n_i^1 \in \Phi_1$, there exists a corresponding norm $n_j^2 \in \Phi_2$ such that norm $n_i^1$ (weakly) specializes norm $n_j^2$ (2) strict specialization holds for at least one norm in $\Phi_1$ or for at least one norm $n_k^2 \in \Phi_2$, no norm in $\Phi_1$ specializes $n_k^2$.*

We remark that the specialization relation is transitive, i.e., if $\Phi_1$ specializes $\Phi_2$ and $\Phi_2$ specializes $\Phi_3$, $\Phi_1$ also specializes $\Phi_3$. We can also define *generalization* relation between normative systems as the inverse of specialization. If $\Phi_1$ strictly specializes $\Phi_2$, we say that $\Phi_2$ is a *strict generalization* of $\Phi_1$ (or we say $\Phi_2$ strictly generalizes $\Phi_1$). Finally, we use the *strict specialization* relation between pairs of normative systems to define what it means for a normative system to be *minimal effective*.

Definition 6. *Let $\mathcal{M}$ be an MAS; $\Phi$ be a normative system; $\omega$ be a property. $\Phi$ is minimal effective with respect to property $\omega$ iff* (1) $\mathcal{M} \oplus \Phi \models \mathrm{A}\Box\omega$ *and* (2) *there does not exist any $\Phi'$ such that $\mathcal{M} \oplus \Phi' \models \mathrm{A}\Box\omega$ and $\Phi'$ strictly specializes $\Phi$.*

Intuitively, $\Phi$ is minimal effective if and only if it is effective and there does not exist any strictly specialized normative system $\Phi'$ which is effective. Our problem in this work is defined as follows: given an MAS $\mathcal{M}$ and a property $\Phi$, how do we systematically synthesize a normative system which is both effective and minimal? We present the details of the minimal effective normative systems for this example in Section 5.1. Note that the minimality property is defined with respect to the way that the set of predicates are defined and also there may be multiple minimal effective normative systems in general for a given property. In such cases, our framework can compute all of them.

## 4. OUR FRAMEWORK

In this section, we present the details of our framework for designing minimal effective normative systems using lightweight formal methods. The particular lightweight formal method we employ in this work is the Alloy modeling language and its associated Alloy Analyzer [18]. However, it is important to note that our synthesis framework itself does not prescribe the use of a particular language or tool. Any other modeling tool (e.g., NuSMV model checker [9]) could be substituted for Alloy so long as: (1) its language is expressive enough for specifying first-order predicates, (2) it allows an exhaustive analysis of the system against a property, and (3) it produces a concrete counterexample, which is used for synthesizing and refining norms.

Alloy and Alloy Analyzer are designed based on the principle of lightweight formal methods. That is, if a counterexample is present in a complicated system, it is likely that some corresponding counterexample exists in a scaled-down version of the system and thus it is often important and sufficient to focus on finding (and fixing) counterexamples within a relative small bound. *We remark this assumption is echoed in our setting.* MAS often contain a large or even arbitrary number of agents which are often similar or even identical. A set of social norms which would work with a large number of agents must work for a small number of them and furthermore, as we demonstrate in our case studies later, a stable set of social norms work for a small number of agents often apply to a large number of agents as well. The underlying reason is that agents in normative systems are inherently symmetric.

### 4.1 Norm Synthesis Process

We now outline a process for automatically synthesizing minimal effective normative systems. Given an MAS, without loss of generality, we assume that each agent can observe the states of $m$ agents (except itself) within its neighborhood. For instance, in the traffic cross system, $m$ is set to be 3. The number of possible normative systems is exponential in the number of all possible norms, which is exponential in the number of agents constituting the preconditions. To efficiently search this huge space for a minimal effective normative system, we propose an approach in which we first synthesize normative systems involving only the most generalized norms for a minimal number of agents, and then incrementally refine them by taking more agents into consideration. We believe that this strategy is justified as we are deliberately searching for simple norms (e.g., an agent would need to observe only a small number of his/her neighbors) first, which in practice often works effectively.

The workflow of the norm synthesis framework is shown in Figure 3. The synthesis process is divided into two phases: (1) generating an initial candidate set of normative systems, and (2) refining them until one of them qualifies as minimal effective. In the following, we present the details of the two phases.

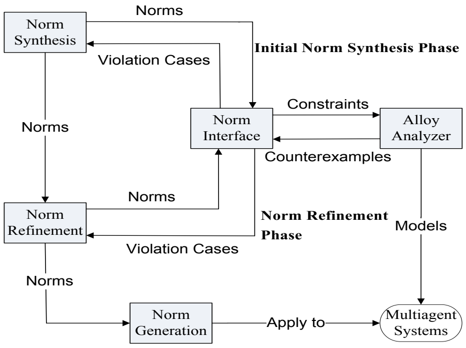


**Figure 3: Norm Synthesis Framework**

**Initial Norm Synthesis** The top three components in Figure 3 are responsible for synthesizing an initial set of candidate normative systems: *Norm Synthesis*, *Norm Interface* and the *Alloy Analyzer*. The designer starts by modeling the behaviors of an MAS and a desired property $\omega$ in the Alloy Analyzer[1]. The MAS model is parameterized with three inputs: a desired property ($\omega$), the number of agents ($k$) in the MAS, and a normative system to be enforced ($\Phi$). Given these inputs, the analyzer automatically checks whether the MAS satisfies $\omega$; if the MAS violates $\omega$, then the analyzer returns a counterexample that shows the sequence of states and actions that leads to the violation.

We start the norm synthesis for MAS involving two agents only ($k = 2$). Since agents are assumed to be identical, and thus there is no need to distinguish the agents. Since $\Phi$ is initially empty, the analyzer will generate a counterexample $c$ that shows how $\omega$ is violated (unless $\omega$ is trivially true without norms). The norm interface translates and passes this counterexample into the *SynthInitNorms* function (Algorithm 1). Note that only the first generated counterexample is considered if there are multiple counterexamples.

The algorithm works as follows. We begin by extracting from counterexample $c$ the set of norms that would have prevented $c$ (line 1). The *ExtractNS* function is constructed as follows. The precondition is simply the current local view of an agent involved in the counterexample $c$, and its action to be forbidden is the action that this agent chooses right before the occurrence of $c$. Note that it might return multiple norms since multiple agents are involved in the counterexample. The current normative system $\Phi$ is expanded to include each possible $\Phi_c \in \Phi$ (line 3), and is analyzed for its effectiveness by re-running the analyzer (*Verify* on line 6). If the analyzer fails to find a counterexample, $\Phi$ must be effective for all possible scenarios in 2-agent system, and thus is included in $\Omega$ to pass onto the refinement phase (line 8). If $\Phi$ leads to another counterexample, $c'$, then we repeat the process (line 10). When the algorithm completes, the resulting $\Omega$ contains the set of all feasible normative systems that always guarantee property $\omega$ for the MAS with two agents (satisfying Formula 1). In the traffic cross system, suppose the first collision counterexample returned by Alloy Analyzer is that car $g$ heads North while car $h$ is heading East and located at the TopLeft position of car $g$, this counterexample can be mapped to either of the following two norms:

- $\langle \psi_1, \{Move\} \rangle$, where $\psi_1 = \mathsf{dirNorth}(g) \wedge \mathsf{topLeft}(h, g) \wedge \mathsf{dirEast}(h)$
- $\langle \psi_2, \{Move\} \rangle$, where $\psi_2 = \mathsf{dirEast}(g) \wedge \mathsf{topRight}(h, g) \wedge \mathsf{dirNorth}(h)$

After that, we enforce either of them on all agents' behaviors and rerun Alloy to generate possible counterexamples separately. This procedure is repeated until no new counterexample is identified. The set of normative systems returned after the initial norm synthesis step are passed to the next phase: norm refinement phase.

**Algorithm 1** SynthInitNorms($\omega$, $\Phi$, $\Omega$, $c$)

```
1: $\Omega_c = ExtractNS(c)$                       // all norms that may prevent $c$
2: for $\Phi_c \in \Omega_c$ do
3:     if $\Phi_c$ not in $\Phi$ then
4:         $\Phi = \Phi \cup \Phi_c$
5:     end if
6:     $c' = Verify(\omega, 2, \Phi)$               // analyze $\Phi$ against $\omega$ for 2 agents
7:     if $c' = \emptyset$ then
8:         $\Omega = \Omega \cup \{\Phi\}$          // no counterexample; $\omega$ satisfied
9:     else
10:        $\Omega = SynthInitNorms(\omega, \Phi, \Omega, c')$
11:    end if
12: end for
13: return $\Omega$
```

**Norm Refinement** The normative systems in $\Omega$ are composed of the most general and effective norms for the 2-agent MAS, i.e., the precondition of each norm in the normative systems in $\Omega$ are defined as a conjunction of atomic predicates that describes the relationship between two agents at most. The normative systems in $\Omega$ may over-constrain the behaviors of the agents. The *norm refinement* phase explores all possible specializations of each normative system $\Phi \in \Omega$, in a scope in which $k$ has been increased by one, discarding ineffective ones. When $k$ reaches $m + 1$, the algorithm terminates and returns a set of minimal effective normative systems for the MAS with $m + 1$ agents.

The space of specializations is exponential in the number of atomic predicates used to describe the local state of an agent. In practice, however, many of those specializations are ineffective, and can be pruned away using a top-down searching approach. The key intuition is as follows. Conceptually, we can order all possible normative systems in a tree structure where the root node represents the initial normative system, and its $i$-th level children represent the specializations of its $i$-th norm with respect to their father nodes. With the other norms unchanged, if a normative system obtained by specializing the $i$-th norm $n_i$ is ineffective, then any possible further specialization of the rest of norms (the nodes within the

[1] The description of MAS modeling in Alloy is omitted since it is not the focus of this paper. Refer to [19] for details.

subtree) will also be ineffective. Thus, all of its descendants can be pruned out from the search space, and continue onto the next specialization of $n_i$.

**Algorithm 2** RefineNorms($\omega$, $\Omega$)

```
1:  $k = 3$                                   // start with 3-agent MAS
2:  repeat
3:      $\Omega' = \emptyset$
4:      for $\Phi \in \Omega$ do
5:          $\Omega^*$ = SpecializeNS($\omega$, $\Phi$, k, 1)            // specialize $\Phi$
6:          $\Omega'$ = $\Omega' \cup (\Omega^* - \{\Phi\})$
7:      end for
8:      $k = k + 1$
9:      if $\Omega' \neq \emptyset$ then
10:         $\Omega = \Omega'$                   // replace $\Omega$ with specialized NSes
11:     end if
12: until k > m+1 or $\Omega$ is unchanged
13: return $\Omega$
```

**Algorithm 3** SpecializeNS($\omega$, $\Phi$, $k$, $i$)

```
1:  $\Omega = \emptyset$
2:  for $n \in \Phi$ do
3:      for $n' \in SpecializeNorm(n)$ do
4:          $\Phi' = (\Phi - n) \cup n'$        // replace $n$ with specialized norm
5:          $c$ = Verify($\omega$, $k$, $\Phi'$)          // check whether $\Phi'$ is effective
6:          if $i = |\Phi|$ then
7:              if c = $\emptyset$ then $\Omega = \Omega \cup \{\Phi'\}$
8:          else
9:              if $c \neq \emptyset$ then continue
10:             $\Omega = \Omega \cup SpecializeNS(\omega, \Phi', k, i + 1)$
11:         end if
12:     end for
13: end for
14: return $\Omega$
```

The refinement algorithm is shown in Algorithm 2. *RefineNorms* takes two arguments: a desired property ($\omega$) and a set of candidate normative systems from the initial phase ($\Omega$). For each normative system $\Phi$ in $\Omega$, function *SpecializeNS* computes all minimal effective specializations of $\Phi$ for the $k$-agent MAS (line 5). The existing normative systems in $\Omega$ are then replaced with their specialized counterparts from the results of *SpecializeNS* (line 10). The process is repeated until $k$ exceeds the size of the local agent view ($m$), or $\Omega$ cannot be specialized further.

Given property $\omega$, normative system $\Phi$, the number of agents $k$, and the current specialization depth $i$, *SpecializeNS* searches for the most specialized versions of $\Phi$ that also satisfy $\omega$ (Algorithm 3). First, a candidate specialization $\Phi'$ is constructed by replacing $n \in \Phi$ with one of $n$'s specializations, $n'$ (lines 3 and 4). The function *SpecializeNorm(n)* (line 3) returns the set of possible specialized norms of norm $n$ in the current specialization context following Definition 3. Then, $\Phi'$ is verified against $\omega$ using the Alloy Analyzer (line 5); if $\Phi'$ is effective and cannot be further specialized, then it is stored in $\Omega$ as a minimal effective normative system. In theory, *SpecializeNS* may explore every possible combination of all specializations of norms in $\Phi$, which is exponential in $|\Phi|$. However, if the Alloy Analyzer returns a counterexample $c$ for a normative system $\Phi'$, every specialization of $\Phi'$ (the subtree of the node corresponding to $\Phi'$) must also be ineffective, and is omitted from the search (line 9).

*Example* An example of specializing a normative system NS is shown in Figure 4. In Figure 4, the normative system contains only one norm NS, which has three possible specialized norms. Each of its specialized norms can also be further specialized. When we search for specialized effective normative system, we check the specialized norms NS-sub2 and NS-sub3 and found that both of them are ineffective. Then we can significantly reduce the searching space by pruning away the subtrees of both norms. We only need to further search the subtree of norm NS-sub1 which is effective, and finally obtain the most specialized one NS-sub1-sub2. If we consider the traffic cross system example, as shown in Section 5.1 later, we found that any specialized normative system of a normative system from the initial norm synthesis stage turned out to be ineffective, thus there is no need to further specialize any more.

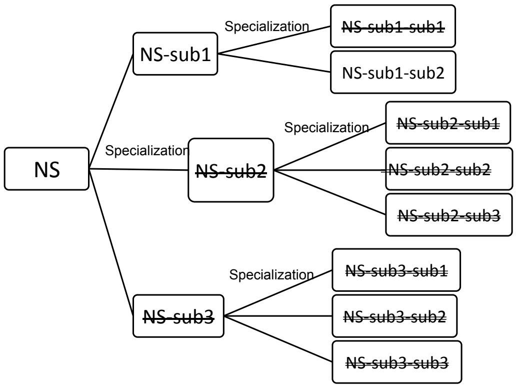


**Figure 4: Specializing Normative System Example**

## 4.2 Theoretic Analysis

We first describe two properties which will be useful for proving the main theorems of our synthesis framework. First, if enforcing a normative system ensures that a property is *always* satisfied by an MAS , then implementing any normative system which *generalizes* the existing normative system instead also ensures that the same property always holds in the MAS.

LEMMA 1. *Let $\mathcal{M}$ be an MAS with an arbitrary number of agents; $\omega$ be a property; and $\Phi$ be a normative system such that $\mathcal{M} \oplus \Phi \models A\Box\omega$. We have $\mathcal{M} \oplus \Phi' \models A\Box\omega$ for any normative system $\Phi'$ that generalizes $\Phi$.* □

The above lemma can be easily proved based on a property of ATL which has been proved in [15], and can be expressed intuitively as follows: "*Implementing behavioral constraints on a multiagent system guarantees to preserve the universal properties of the system.*" We omit the proof of this property and simply use the result here. The overall formula $A\Box\omega$ is one kind of universal property [15], and also based on definition of *generalization* relation (Section 3), enforcing any generalized normative system is equivalent with implementing additional behavioral constraints on agents. Therefore, this lemma holds.

Another useful property is that if a normative system ensures the satisfaction of a property in a k-agent MAS, then the same property can also be satisfied in any MAS with fewer agents by enforcing the same normative system. Given an MAS $\mathcal{M}$, we write $\mathcal{M}_k$ to denote the MAS with exactly $k$-agents.

LEMMA 2. *Let $\mathcal{M}$ be an MAS; $\omega$ be a property; and $\Phi$ be a normative system such that $\mathcal{M}_k \oplus \Phi \models A\Box\omega$. $\mathcal{M}_n \oplus \Phi \models A\Box\omega$ for any $n \leq k$.* □

We sketch the proof of this lemma using contradiction as follows. Let us assume that the normative system $\Phi$ is not effective for a multiagent system $\mathcal{M}'$ with less than $k$ agents. It means that the system must be able to evolve to certain state $s$ which violates property $\omega$ under $\Phi$. We can construct a $k$-agent system by letting one agent $i$ always be outside the neighborhood of the rest of agents and not choose any action each round. Thus this additional agent $i$ does not affect the state transition of the rest of agents each round, and the dynamics of state transitions for the rest of agents is actually the same for the system with less than $k$ agents. Therefore, the agents must also be able to reach a state that violates property $\omega$. This leads to a contradiction.

Now we are ready to establish the correctness of our approach, i.e., every normative system synthesized from the framework is minimal effective in always satisfying property $\omega$ for any MAS with $m + 1$ agents (Formula 1).

THEOREM 1. *Let $\mathcal{M}$ be an MAS and $\omega$ be a property. If any agent $\mathcal{M}$ can only observe the states of $m$ neighboring agents. Any normative system $\Phi$ in the final set $\Omega$ synthesized from the framework is minimal effective with respects to $\omega$ for $\mathcal{M}_{m+1}$.*

The proof of the theorem is available in Appendix. Although the above theorem guarantees that $\Phi \in \Omega$ is minimal effective for the MAS with $m + 1$ agents, it may not be for a larger number of agents. In contrast, the objective of designing normative systems is often to have a system which is minimal effective for many or arbitrary number of agents. Though our framework fulfills the objective directly, as we show in our case studies, since the normative systems synthesized using our framework often could be proved to be minima effective for an arbitrary number of agents, with a manual induction proof. The following theorem further reduces the effort required for manual proving by showing that as long as we prove that $\Phi$ is effective for the MAS with more agents (i.e., $\mathcal{M}_k$ where $k > m + 1$), $\Phi$ is guaranteed to be minimal as well.

THEOREM 2. *Let $\mathcal{M}$ be an MAS. Let $\phi \in \Omega$ be a minimal effective normative system for $\mathcal{M}_{m+1}$ with respects to property $\omega$ synthesized using our approach. If $\Phi$ is effective with respects to property $\omega$ for $\mathcal{M}_k$ where $k > m + 1$, it is minimal effective with respects to property $\omega$ for $\mathcal{M}_k$.*

PROOF. We prove this theorem by contradiction. Let us assume that there exists another normative system $\Phi_1$ which specializes $\Phi$ and is minimal effective for the $k$-agent system ($k > m+1$). Based on Lemma 2, we know that $\Phi_1$ is also effective for the $(m + 1)$-agent system, which also specializes $\Phi$. This contradicts with the fact that $\Phi$ is minimal effective for the $(m + 1)$-agent system. □

# 5. EVALUATION

In this section, we evaluate our synthesis framework with multiple MASs in order to answer the following two research questions.

- RQ1: How efficient is it to use our framework to synthesize a set of minimal effective normative systems?
- RQ2: Is our underlying assumption justified, i.e., are the normative systems we synthesized based on a small number of agents generalizes to an arbitrary number of agents?

## 5.1 Traffic Junction Model

In the following, we apply our synthesis framework to a practical and well-studied multi-agent coordination problem, i.e., a traffic junction network [24, 25, 20]. We present minimal effective normative systems synthesized from our framework for ensuring a *non-collision* property, and compare our results to the state-of-the-art simulation-based synthesis approach. Note that this property is also applicable to other similar domains such as prey-predator problem [32] or other multi-agent coordination problems [26].

**Table 1: The Norms Generated from Our Framework**

| Norm | Precondition | Actions |
|---|---|---|
| $n_1$ | dirNorth$(g)$ ∧ topLeft$(h, g)$ ∧ dirEast$(h)$ | {Move} |
| $n_2$ | dirNorth$(g)$ ∧ top$(h, g)$ | {Move} |
| $n_3$ | dirSouth$(g)$ ∧ topLeft$(h, g)$ ∧ dirEast$(h)$ | {Move} |
| $n_4$ | dirSouth$(g)$ ∧ top$(h, g)$ | {Move} |
| $n_5$ | dirWest$(g)$ ∧ topLeft$(h, g)$ ∧ dirEast$(h)$ | {Move} |
| $n_6$ | dirWest$(g)$ ∧ top$(h, g)$ | {Move} |
| $n_7$ | dirEast$(g)$ ∧ topLeft$(h, g)$ ∧ dirEast$(h)$ | {Move} |
| $n_8$ | dirEast$(g)$ ∧ top$(h, g)$ | {Move} |

Our representation of the traffic network is based on the model by Morales *et al.* [25]. We have already introduced this model in Section 3 and the traffic network is shown in Figure 1(a). Assuming the absence of a central controller that co-ordinates the movement of the cars within the junction (i.e., no traffic lights), our goal is to automatically synthesize minimal effective norms for the following safety property: (non-collision) no two cars should ever occupy the single cell on the traffic network anytime.

*Predicates:* in each local view, for the sake of modeling convenience in Alloy, we define 4 predicates (e.g., dirEast()) describing the direction of a car in each cell (i.e., $4^4$ possibilities), along with 3 predicates (e.g., topLeft) that describe a relationship between any two cars in their local views (i.e., $2^3$ possibilities). Thus, we can estimate there are $4^4 \times 2^3 = 2048$ possible norms, leading to potentially $2^{2048}$ normative systems.

**Experiment Results** Our framework generated a minimal effective normative system $\Omega$ for the non-collision property with $m = 3$ and $k = 5$. The overall synthesis process invoked the Alloy Analyzer (i.e., *Verify* from the synthesis algorithm) 108 times to check the effectiveness of candidate normative systems, which pruned away lots of candidate normative systems. The reason is that any specialization when $k > 2$ (i.e., any norm with a precondition that is expressed over more than 2 agents) turned out to be ineffective, thus there is no need to further specialize any more. We ran our experiments on a Windows 7 machine with a 2.4GHZ CPU and 4GB RAM, and it took approximately 4 minutes to finish the synthesis.

The generated normative system $\Omega$ contains 8 norms. Many of these norms are symmetric, and cover the equivalent scenario under different orientations of the car. For example, $\Omega$ requires a car $g$ to stop when it is directly behind another car $h$; $\Omega$ covers this scenario using four separate norms corresponding to norm $n_2$, $n_4$, $n_6$ and $n_8$ in Table 1 respectively, depending on the direction of $g$. The rest of four norms in Table 1 cover another equivalent scenario where car $g$ is not allowed to move and yield to another car $h$ that is approaching $g$ from the left (i.e., *priority to the left*).

We compare the results of our experiment ($\Phi_A$) to the normative system ($\Phi_B$) produced for the same non-collision problem using the state-of-the-art simulation-based synthesis framework in [25]. The authors of [25] used a different set of atomic predicates to model the traffic junction, however, the two models are equivalent in terms of the dynamics of agent behaviors (i.e., the model size is equivalent). Thus, we first translate their normative system into our representation. Their normative system contains 16 norms, as listed in Table 2. The norms can be divided into four sets of symmetric norms ($n_1 - n_4$, $n_5 - n_8$,$n_9 - n_{12}$, $n_{13} - n_{16}$) that differ only in the direction of car $g$.

First we can observe that $\Phi_A$ and $\Phi_B$ share one common set of norms; namely, *priority to the left* norms ($n_1$ and $n'_1$). However,

**Table 2: The Normative System Generated in [11]**

| Norm | Precondition | Actions |
|---|---|---|
| $n'_1$ | $\mathsf{dirNorth}(g) \wedge \mathsf{topLeft}(h,g) \wedge \mathsf{dirEast}(h)$ | {Move} |
| $n'_2$ | $\mathsf{dirNorth}(g) \wedge \mathsf{top}(h_1,g) \wedge \mathsf{dirWest}(h_1) \wedge \mathsf{topLeft}(h_2,g) \wedge \mathsf{dirWest}(h_2)$ | {Move} |
| $n'_3$ | $\mathsf{dirNorth}(g) \wedge \mathsf{top}(h_1,g) \wedge \mathsf{dirEast}(h_1) \wedge \mathsf{topRight}(h_2,g) \wedge \mathsf{dirEast}(h_2)$ | {Move} |
| $n'_4$ | $\mathsf{dirNorth}(g) \wedge \mathsf{top}(h,g) \wedge \mathsf{dirNorth}(h)$ | {Move} |
| $n'_5$ | $\mathsf{dirSouth}(g) \wedge \mathsf{topLeft}(h,g) \wedge \mathsf{dirWest}(h)$ | {Move} |
| $n'_6$ | $\mathsf{dirSouth}(g) \wedge \mathsf{top}(h_1,g) \wedge \mathsf{dirEast}(h_1) \wedge \mathsf{topLeft}(h_2,g) \wedge \mathsf{dirEast}(h_2)$ | {Move} |
| $n'_7$ | $\mathsf{dirSouth}(g) \wedge \mathsf{top}(h_1,g) \wedge \mathsf{dirWest}(h_1) \wedge \mathsf{topRight}(h_2,g) \wedge \mathsf{dirWest}(h_2)$ | {Move} |
| $n'_8$ | $\mathsf{dirSouth}(g) \wedge \mathsf{top}(h,g) \wedge \mathsf{dirSouth}(h)$ | {Move} |
| $n'_9$ | $\mathsf{dirWest}(g) \wedge \mathsf{topLeft}(h,g) \wedge \mathsf{dirNorth}(h)$ | {Move} |
| $n'_{10}$ | $\mathsf{dirWest}(g) \wedge \mathsf{top}(h_1,g) \wedge \mathsf{dirSouth}(h_1) \wedge \mathsf{topLeft}(h_2,g) \wedge \mathsf{dirSouth}(h_2)$ | {Move} |
| $n'_{11}$ | $\mathsf{dirWest}(g) \wedge \mathsf{top}(h_1,g) \wedge \mathsf{dirNorth}(h_1) \wedge \mathsf{topRight}(h_2,g) \wedge \mathsf{dirNorth}(h_2)$ | {Move} |
| $n'_{12}$ | $\mathsf{dirWest}(g) \wedge \mathsf{top}(h,g) \wedge \mathsf{dirWest}(h)$ | {Move} |
| $n'_{13}$ | $\mathsf{dirEast}(g) \wedge \mathsf{topLeft}(h,g) \wedge \mathsf{dirSouth}(h)$ | {Move} |
| $n'_{14}$ | $\mathsf{dirEash}(g) \wedge \mathsf{top}(h_1,g) \wedge \mathsf{dirNorth}(h_1) \wedge \mathsf{topLeft}(h_2,g) \wedge \mathsf{dirNorth}(h_2)$ | {Move} |
| $n'_{15}$ | $\mathsf{dirEast}(g) \wedge \mathsf{top}(h_1,g) \wedge \mathsf{dirNorth}(h_1) \wedge \mathsf{topRight}(h_2,g) \wedge \mathsf{dirNorth}(h_2)$ | {Move} |
| $n'_{16}$ | $\mathsf{dirEast}(g) \wedge \mathsf{top}(h,g) \wedge \mathsf{dirEast}(h)$ | {Move} |

norm $n_2$ is more constraining than the combination of the norms $n'_2$, $n'_3$, and $n'_4$, since $n_2$ prevents $g$ from moving forward when it is behind another car $h$, *regardless* of the direction of $h$. Thus, $\Phi_B$ is more specialized than $\Phi_A$, which might seem more desirable than $\Phi_A$. However, $\Phi_B$ is not effective in ensuring that the MAS satisfies the non-collision property. To demonstrate this, we encoded $\Phi_B$ in our Alloy model, and checked their effectiveness. The Alloy Analyzer returned a counterexample that demonstrates a collision between two cars. Consider a state in which two cars, $g$ and $h$, are described using the following predicates:

$$\mathsf{dirNorth}(g) \wedge \mathsf{top}(g,h) \wedge \mathsf{dirEast}(h)$$

None of the norms in $\Phi_B$ is applicable to this local view of $g$, and so the MAS allows $g$ to perform any action. When $g$ decides to move north and $h$ decides to stay in the same location, the two cars end up in a collision. It is worth noting that during our refinement algorithm (Algorithm 3), $n'_2$, $n'_3$, and $n'_4$ are considered as candidate specializations of $n_2$. They are discarded after being determined to be ineffective. Thus it shows that the synthesis approach using a verification tool is complementary to simulation-based approaches and can provide strong guarantee on effectiveness or minimality.

In the following, we intuitively show that the synthesized norms can be easily generalized to an arbitrary number of agents. That is, the minimal effective normative system synthesized for the 7 × 7 road junction with 4 cars is also minimal effective for the general case of $n \times n$ road junction with $m$ cars. Based on the semantics of the normative system we synthesized in Table 1, the normative system is essentially equivalent with implementing the following general rule: an arbitrary car $c$ should stop if there is a car $c'$ right in front of $c$ or if there is another car $c'$ on its left-hand side and heading towards its right-side. It can be proved that, for any $n \times n$ road junction with $m$ cars, if there is a collision, there must be a violation of this rule. Thus the normative system in Table 1 always guarantees no collision for any general case.

## 5.2 Two-hop Coloring Problem

In the following, we apply our approach to synthesize population protocols. The population protocol model has become a very important computation paradigm for modeling distributed systems such as mobile ad hoc networks [16] and researchers are actively designing and publishing such protocols [7, 21]. Intuitively, a population protocol can be considered as a set of rules specifying the local behaviors of each agent in a population such that certain desirable property can be satisfied eventually. The challenge is that a population protocol must work with an arbitrary number of agents and an arbitrary initial configuration of the agents. For instance, in the two-hop coloring problem [6], a population protocol is required for a ring of agents so that *eventually always* each pair of neighboring agents should have different colors. For a ring network, we can see that three colors should be sufficient. Thus the problem is to design a protocol to regulate the behaviors of each agent such that the goal can be eventually and always satisfied through finite local interactions (computations) among each pair of neighboring agents. Existing work [27, 23, 12] mainly investigates how to effectively and automatically check whether a given two-hop coloring protocol can eventually and always satisfy the goal. Here we take a different perspective and consider the question of how an effective two-hop coloring protocol can be automatically synthesized.

To synthesize a two-hop coloring protocol, we observe that a protocol corresponds to the opposite of a normative system. For example, given a state $s$, if a protocol specifies that an agent should perform an action from a subset $A' \subseteq A$, then the corresponding norm should specify that this agent should not perform any action from the subset $A \backslash A'$. Based on this observation, the original question of synthesizing an effective protocol can be translated into an equivalent question of how to synthesize a minimal effective normative system such that the desirable property is satisfied. Formally we aim at synthesizing a minimal effective normative system $\Phi$ such that the following property can be satisfied: $\mathcal{M} \oplus \Phi \models \Diamond\Box\omega$, where $\omega$ denotes the targeted state that no neighboring agents have the same color. Note that this property is not a universal property, however, we show that our framework is also able to synthesize a minimal and effective normative system.

A two-hop coloring protocol specifies the allowed behavior of a pair of neighboring agents for their local interaction. Accordingly the synthesis of the corresponding normative system is the specification of the forbidden behavior of each pair of neighboring agents for their local interaction. Inspired by [6], given a population of agents in a ring network, we assume that each agent consists of two basic components: color $c[i]$ and its boolean color index $F[i][u]$ ($u$ denotes a color). The color of an agent can be considered as its action and the color index can be considered as its internal state. We represent the local state of each pair of neighboring agents $i$ and $j$ using their boolean color indexes $F[i][u]$ and $F[j][u]$, and thus there are two possible states: either the agents are synchronized (i.e., $F[i][\mathit{color}[j]] = F[j][\mathit{color}[i]]$) or they are not (i.e., $F[i][\mathit{color}[j]] \neq F[j][\mathit{color}[i]]$).

For each pair of interacting agents $i$ and $j$, without loss of generality, assume that agent $i$ is the initiator and agent $j$ is the responder. The action set $A_i$ of an initiator agent $i$ consists of the combinations of any possible operations on the previous two components, i.e., updating the color so that $\mathit{color}[i]$ is $0, 1$ or $2$; or updating the boolean color index state so that $F[i][\mathit{color}[j]]$ is $\neg F[i][\mathit{color}[j]]$ or $F[j][\mathit{color}[i]]$ or $\neg F[j][\mathit{color}[i]]$. In contrast, the action set of agent $j$ is restricted to operations on the update of the boolean color index only, i.e., setting $F[j][\mathit{color}[i]]$ to be $\neg F[j][\mathit{color}[i]]$ or $F[i][\mathit{color}[j]]$ or $\neg F[i][\mathit{color}[j]]$.

**Table 3: The Norms Generated from Our Framework for a pair of neighboring agents $i$ and $j$**

| Norm | Precondition | (forbidden) Operations |
|---|---|---|
| $n_1$ | Equal($F[i][color[j], F[j][color[i]]$) | {color[i]=0/1/2} |
| $n_2$ | Equal($F[i][color[j], F[j][color[i]]$) | {F[i][color[j]] = F[j][color[i]]/¬ F[j][color[i]]} |
| $n_3$ | Equal($F[i][color[j], F[j][color[i]]$) | {F[j][color[i]] = F[i][color[j]]/¬ F[i][color[j]]/F[i][color[j]]} |
| $n_4$ | NotEqual($F[i][color[j], F[j][color[i]]$) | {F[i][color[j]] = F[j][color[i]]/¬ F[j][color[i]]} |
| $n_5$ | NotEqual($F[i][color[j], F[j][color[i]]$) | {F[j][color[i]] = F[i][color[j]]/¬ F[i][color[j]]/¬ F[j][color[i]]} |

**Table 4: The Protocol Generated from Our Framework for a pair of neighboring agents $i$ and $j$**

| Rules | Precondition | (allowed) Operations |
|---|---|---|
| $r_1$ | Equal($F[i][color[j], F[j][color[i]]$) | {F[i][color[j] = ¬ F[i][color[j]]} |
| $r_2$ | Equal($F[i][color[j], F[j][color[i]]$) | {F[j][color[i] = ¬ F[j][color[i]]} |
| $r_3$ | NotEqual($F[i][color[j], F[j][color[i]]$) | {color[i] = 0/1/2} |
| $r_4$ | NotEqual($F[i][color[j], F[j][color[i]]$) | {F[i][color[j] = F[j][color[i]]} |

**Experimental Results** Our framework first generated a minimal effective normative system for the two-hop coloring property with a 3-node ring. The analysis involves 1028 times of checking to verify the effectiveness of candidate normative systems. We ran our experiments on the same machine and it took approximately 50 minutes to synthesize the norms. The normative system synthesized from our framework is shown in Table 3, which specifies the actions that are not allowed in different states. We then obtain the most "general" protocol for this two-hop coloring problem by "negating" the normative system. The two-hop coloring protocol synthesized is shown in Table 4. Specifically, rule 1 and 2 specify how the values of $F[i][color[j]]$ and $F[j][color[i]]$ should be updated if they are synchronized; rule 3 specifies that the value of $color[i]$ can be updated to any integer value between 0 and 2 if they are unsynchronized; rule 4 specifies how the value of $F[i][color[j]]$ should be updated if they are unsynchronized.

Next, we show that the protocol synthesized above for a 3-node ring works for a network of an arbitrary number of nodes. Formally, we would like to show that, for any $n$-node ring, the protocol in Table 4 guarantees that *eventually always* each pair of neighboring agents have different colors (hereafter the property). We sketch the proof based on mathematical induction as follows. We already know that the protocol works for a 3-node ring. For induction, we assume that it works for a $n$-node ring and proves that it works for a $(n+1)$-node ring as well. Given a $(n+1)$-node ring, for any node $u$ connecting another two neighboring nodes $v$ and $w$, we can first reduce it into a $n$-node ring by connecting nodes $v$ and $v'$ and ignoring node $u$ first. Then the reduced $n$-node ring can always satisfy the two-hop coloring property. After that, we insert node $u$ back into its original position in the ring and apply the same protocol again. Next we show that after a finite steps, the $(n+1)$-node ring satisfies the property. First, based on the protocol in Table 4, we can see that if the ring satisfies the property and all nodes are synchronized, nothing changes in the ring and the property is satisfied. Now let us consider all possible scenarios when node $u$ is inserted back into the ring.

If node $u$ violates the property, and without loss of generality, let us assume $color[u] = color[w]$ where node $w$ is the other neighbor of node $v$. We consider the following three cases.

*Case 1* ($F[u][color[v]] \neq F[v][color[u]]$): let the protocol work on the pair of nodes $(u, v)$. There must be one color $c$ which can be assigned to node $u$ such that property is satisfied. Let $color[u] = c$. This operation removes the color violation from node $u$. By updating $F[u][color[v]]] = F[v][color[u]]$, nodes $u, v$ are synchronized.

*Case 2* ($F[u][color[v]] = F[v][color[u]] \neq F[w][color[v]]$): Let the protocol work on the pair of nodes $(v, w)$. Then the values of $F[w][color[v]]$ and $F[v][color[w]]$ will be flipped. Node $u$ and $v$ will be unsynchronized. Next, it is resolved as Case 1.

*Case 3* ($F[u][color[v]] = F[v][color[u]] = F[w][color[v]]$): Let the protocol work on the pair of (v,w) and (u,v) sequentially. This will cause the values of $F[v][color[w]]$ and $F[u][color[v]]$ flip once and the value of $F[v][color[u]]$ flip twice. Thus node $u$ and $v$ become unsynchronized. Next, it is resolved as Case 1.

If the original color of node $u$ satisfies the property, we only need to make sure it is synchronized with its neighbors $v$ and $w$. Suppose that they are not synchronized initially. Let the color update operation be $color[u] = color[u]$ (unchanged), and the update operation on function $F$ will synchronize the value of $F[u][color[v]]$ and $F[v][color[u]]$. Based on the previous result, we conclude the protocol works for the an arbitrary number of nodes.

## 5.3 Orienting Undirected Rings Problem

In the following, we apply our approach to synthesize a population protocol for orienting undirected rings and compare the result with the one published in [6]. Given a ring that is two-hop colored already, we are interested in reaching a state such that all nodes are well-oriented in the sense that (1) each node has exactly one predecessor and one successor and the predecessor and successor should not be the same; (2) for each pair of nodes $u$ and $v$, if $u$ is the predecessor of $v$, then $v$ must be the successor of $u$; (3) for each pair of neighboring nodes $u$ and $v$, either $u$ is the predecessor or successor of $v$. We are interested in finding a protocol such that the previous orientation property can be eventually satisfied in the ring as long as all nodes (agents) follow the protocol to update their states in a pairwise manner. Similar to the previous example, to synthesize such a protocol, we start with synthesizing the corresponding effective normative system using our framework. Formally we aim to synthesize a minimal effective normative system $\Phi$ such that the following property can be satisfied: $\mathcal{M} \oplus \Phi \models \Diamond\Box\omega$, where $\omega$ denotes the targeted state satisfying the above orientation property.

Naturally, we represent the state of a node $u$ using the following three components: its current color $color[u]$, its successor node color $succolor[u]$ and its predecessor node color $predcolor[u]$. During an interaction between a pair of nodes $(u, v)$, without loss of generality, node $u$ is assumed to be the initiator and node $v$ is the responder. By considering whether the initiator node $u$ is the successor or predecessor of responder node $v$, we distinguish the states between a pair of nodes into the following four states,

- u is the successor but not the predecessor of v: $succolor[u] = color[v]$ and $predcolor[u] \neq color[v]$
- u is the predecessor but not the successor of v: $predcolor[u] = color[v]$ and $succolor[u] \neq color[v]$

Table 5: The Protocol Generated from Our Framework for a Pair of Neighboring Agents $i$ and $j$

| Norm | Precondition | (allowed) Operations |
|---|---|---|
| $n_1$ | Equal$(succolor[u], color[v])$ and !Equal$(predcolor[u], color[v])$ | {action a1} |
| $n_2$ | !Equal$(succolor[u], color[v])$ and Equal$(predcolor[u], color[v])$ | {action a2} |
| $n_3$ | Equal$(succolor[u], color[v])$ and Equal$(predcolor[u], color[v])$ | {action a2} |
| $n_4$ | !Equal$(succolor[u], color[v])$ and !Equal$(predcolor[u], color[v])$ | {action a2} |

- u is both the predecessor and successor of v: $predcolor[u] = color[v]$ and $succolor[u] = color[v]$
- u is neither the predecessor or successor of v: $predcolor[u] \neq color[v]$ and $succolor[u] \neq color[v]$

For each state, the nodes can update either $succolor[u]$ or $predcolor[u]$ in either of the following ways.

- node $u$ is the predecessor of node $v$ (denoted as action a1): $predcolor[v] = color[u]$ and $succolor[u] = color[v]$
- node $u$ is the successor of node $v$ (denoted as action a2): $predcolor[u] = color[v]$ and $succolor[v] = color[u]$

**Experimental Results** Our framework first generated a minimal effective normative system for the orienting property under a 3-node ring. The overall synthesis process involves 124 times of checking to validate the effectiveness of candidate normative systems. We ran our experiments on the same machine and it took approximately 5 minutes to synthesize the norms. The synthesized protocol for this orienting undirected ring problem can be obtained by taking the opposite of the synthesized normative system similarly as the previous example, which is shown in Table 5. The protocol consists of four rules, each of which specifies the action to be performed under each condition (state).

In the following, we generalize the protocol to any $n$-node ring. We again prove it works $n$-node ring using mathematical induction as follows. Let us first assume that the protocol works for a $m$-node ring. Next we construct the corresponding $(m + 1)$-node ring by inserting another node $v$ into an arbitrary position. Let us denote its neighboring nodes as $u$ and $w$ and their neighbors are $u'$ and $w'$ respectively. Without loss of generality, let us assume that $color[u] = predcolor[u']$ and $color[w] = succolor[w']$.

Assume that node $u$ and $v$ interact first. Since the $(m + 1)$-node ring also satisfies the two-hop coloring property, there is only one possibility after this interaction, i.e., $predcolor[u] = color[v]$ and $succolor[v] = color[u]$. Next nodes $v$ and $w$ interact, based on the two-hop coloring property, we know that only one possibility happens as follows: $color[v] = succolor[w]$ and $predcolor[v] = color[w]$. After that all nodes have been well-oriented and the well-oriented configuration remains thereafter since always action $a2$ is selected following the protocol. Thus the theorem holds.

**Remarks:** we note that for both case studies in Section 5.2 and 5.3, the synthesized protocols are equivalent with the ones proposed by Angluin *et al.* [6]. In response to the research questions we introduced at the beginning of Section 5, we have demonstrated in the above examples that 1) we can efficiently synthesize non-trivial normative systems; 2) the synthesized results indeed can be generalized to cases with arbitrary number of agents, and are as good as those published ones. In general, manual generalization of the synthesized norms for complex problems could be challenging. However, our approach shows the potential of the lightweight formal method: it can be useful for automatically generating candidate norms which could be generalized. The manual induction proof may be automatized by leveraging theorem proof techniques. Furthermore, while it is true formal methods suffer from scalability issues, it is less a concern in our framework as in all these representative cases, we find the right norms with as few as three agents.

## 6. CONCLUSION AND FUTURE WORK

We proposed an approach based on lightweight formal methods and tools to automatically synthesize minimal effective normative systems and protocols for multi-agent systems. Complementary to a simulation-based approach, our approach provides a theoretical guarantee on the effectiveness or minimality within the given bounds of the analysis. As future work, we plan to further automatize the framework by leveraging theorem proving techniques to help users establish the induction proof automatically.

## 7. APPENDIX

We prove Theorem 1 by induction as follows. For the sake of convenience, we denote the set of normative systems returned from *SynthInitNorms* and each round of norm refinement process (*RefineNorms*) with $k = 3, ..., m+1$ agents as $\Omega_2$ and $\Omega_k$ respectively.

Initially, when the system has only two agents ($k = 2$), from the initial norm synthesis process in Section 4.1, we know that $\Omega_2$ satisfied the following properties: 1) it contains all effective normative system for 2-agent system where each norm precondition is defined in terms of at most $k = 2$ agents. 2) each norm in $\Omega_2$ can not be further specialized given that the norm precondition is defined in terms of at most $k = 2$ agents.

When we increase the number of agents and synthesize specialized normative systems during the norm refinement process, we are actually exploring all the possible specialized normative systems by specializing the precondition of each norm in each normative system obtained from the initial norm synthesis process. Next we prove that for any case of $k$ agents, if the set of $\Omega_k$ of normative systems satisfied the above properties, then the above two properties also hold for the set $\Omega_{k+1}$. Suppose that this is not true, then there must exist another normative system $\Phi'$ in which each norm precondition is defined in terms of at most k+1 agents, and $\Phi'$ is effective for $(k + 1)$-agent system and also not in $\Omega_{k+1}$. We can always construct a corresponding generalized normative system $\Phi''$ of $\Phi'$, such that the precondition of each norm in $\Phi''$ is defined as a conjunction of atomic predicates that describes the relationship between $k$ agents at most. Based on Lemma 1, we know that $\Phi''$ is also effective for $(k + 1)$-agent system. Further, $\Phi''$ is also effective for $k$-agent system based on Lemma 2. Thus $\Phi''$ must be in $\Omega_k$, which leads to a contradiction. Therefore, there does not exist any normative system that does not belong to $\Omega_{k+1}$ and also is effective for $(k + 1)$-agent system.

Based on previous induction, finally we have the set $\Omega_{m+1}$ contains all the effective normative systems in always satisfying property $\omega$ for the $(m + 1)$-agent system. Since the normative systems in $\Omega_{m+1}$ are not comparable in terms of specialization relationship (all generalized normative systems are excluded in Algorithm 2 (Line 6)), any normative system in $\Omega_{m+1}$ is minimal effective in always satisfying property $\omega$ for $(m + 1)$-agent system.

# 8. REFERENCES

[1] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Conservative social laws. In *Proceedings of 20th European Conference on Artificial Intelligence.*, pages 49–54, 2012.

[2] T. Ågotnes and M. Wooldridge. Optimal social laws. In *Proceedings of AAMAS10*, pages 667–674, 2010.

[3] D. Alrajeh, J. Kramer, A. Russo, and S. Uchitel. Learning operational requirements from goal models. In *Proceedings of the 31st international conference on software engineering*, pages 265–275. IEEE Computer Society, 2009.

[4] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM (JACM)*, 49(5):672–713, 2002.

[5] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[6] D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(4):13, 2008.

[7] J. Aspnes and E. Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. Springer, 2009.

[8] G. Christelis and M. Rovatsos. Automated norm synthesis in an agent-based planning environment. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 161–168, 2009.

[9] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *Computer Aided Verification*, pages 359–364. Springer, 2002.

[10] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

[11] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[12] J. Clément, C. Delporte-Gallet, H. Fauconnier, and M. Sighireanu. Guidelines for the verification of population protocols. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 215–224. IEEE, 2011.

[13] N. Criado, E. Agrente, and V. Botti. Open issus for normative multi-agent systems. *AI communications*, pages 233–264, 2011.

[14] R. Degiovanni, D. Alrajeh, N. Aguirre, and S. Uchitel. Automated goal operationalisation based on interpolation and sat solving. In *Proceedings of the 36th International Conference on Software Engineering*, pages 129–139. ACM, 2014.

[15] W. V. der Hoek, M. Robert, and M. Wooldridge. Social laws in alternating time: effectiveness, feasibility and synthesis. *Synthese*, pages 1–19, 2007.

[16] S. Giordano et al. Mobile ad hoc networks. *Handbook of wireless networks and mobile computing*, pages 325–346, 2002.

[17] D. Jackson. Lightweight formal methods. In *FME 2001: Formal Methods for Increasing Software Productivity, International Symposium of Formal Methods Europe, Berlin, Germany, March 12-16, 2001, Proceedings*, page 1, 2001.

[18] D. Jackson. Alloy: a lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.*, 11(2):256–290, 2002.

[19] D. Jackson. *Software Abstractions: Logic, Language, and Anlysis*. The MIT Press, 2011.

[20] J. Koeppen and M. Lopez-Sanchez. Generating new regulations by learning from experience. In *Proceedings of 9th workshop on Coordination, Organization, Institutions and Norms in Multi-agent Systems*, pages 72–79, 2010.

[21] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 513–522. ACM, 2010.

[22] E. Letier and A. Van Lamsweerde. Deriving operational software specifications from system goals. In *Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering*, pages 119–128. ACM, 2002.

[23] Y. Liu, J. Pang, J. Sun, and J. Zhao. Verification of population ring protocols in pat. In *Theoretical Aspects of Software Engineering, 2009. TASE 2009. Third IEEE International Symposium on*, pages 81–89. IEEE, 2009.

[24] J. Morales, M. López-Sánchez, and M. Esteva. Using experience to generate new regulations. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, pages 307–312, 2011.

[25] J. Morales, M. Lopez-Sanchez, J. A. Rodriguez-Aguilar, M. Wooldridge, and W. Vasconcelos. Automated synthesis of normative systems. In *Proceedings of AAMAS13*, pages 483–490, 2013.

[26] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[27] J. Pang, Z. Luo, and Y. Deng. On automatic verification of self-stabilizing population protocols. *Frontiers of Computer Science in China*, 2(4):357–367, 2008.

[28] Y. Shoham and M. Tennenholtz. On the synthesis of useful socail laws for artificial agent societies. In *Proceedings of AAAI 1992*. ACM Press, 1992.

[29] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73:231–252, 1995.

[30] W. W. Vasconcelos, M. J. Kollingbaum, and T. J. Norman. Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 19(2):124–152, 2009.

[31] W. Visser, N. Bjørner, and N. Shankar. Software engineering and automated deduction. In *Proceedings of the on Future of Software Engineering*, pages 155–166. ACM, 2014.

[32] C. H. Yong and R. Miikkulainen. Cooperative coevolution of multi-agent systems. *University of Texas at Austin, Austin, TX*, 2001.