Normative Multiagent Systems: A *Dynamic* Generalization

Xiaowei Huang[1,2], **Ji Ruan**[3], **Qingliang Chen**[1], **Kaile Su**[1,4]
[1]Department of Computer Science, Jinan University, China
[2]Department of Computer Science, University of Oxford, United Kingdom
[3]School of Engineering, Computer and Mathematical Sciences,
Auckland University of Technology, New Zealand
[4]Institute for Integrated and Intelligent Systems, Griffith University, Brisbane, Australia

Abstract

Social norms are powerful formalism in coordinating autonomous agents' behaviour to achieve certain objectives. In this paper, we propose a dynamic normative system to enable the reasoning of the changes of norms under different circumstances, which cannot be done in the existing static normative systems. We study two important problems (norm synthesis and norm recognition) related to the autonomy of the entire system and the agents, and characterise the computational complexities of solving these problems.

1 Introduction

Multiagent systems have been used to model and analyse distributed and heterogeneous systems, with agents being suitable for modelling software processes and physical resources. Roughly speaking, autonomy means that the system by itself, or the agents in the system, can decide for themselves what to do and when to do it [Fisher *et al.*, 2013]. To facilitate autonomous behaviours, agents are provided with capabilities, e.g., to gather information by making observations (via e.g., sensors) and communicating with each other (via e.g., wireless network), to affect the environment and other agents by taking actions, etc. Moreover, systems and agents may have specific objectives to pursue. In this paper, we study autonomy issues related to social norms [Shoham and Tennenholtz, 1992], which are powerful formalism for the coordination of agents, by restricting their behaviour to prevent destructive interactions from taking place, or to facilitate positive interactions [van der Hoek *et al.*, 2007; Ågotnes and Wooldridge, 2010].

Existing normative systems [Wooldridge and van der Hoek, 2005; Ågotnes *et al.*, 2007; Christelis and Rovatsos, 2009; Ågotnes and Wooldridge, 2010; Morales *et al.*, 2013] impose restriction rules on the multiagent systems to disallow agents' actions based on the evaluation of current system state. An implicit assumption behind this setting is that the normative systems do not have (normative) states to describe different social norms under different circumstances. That is, they are *static* normative systems. Specifically, if an action is disallowed on some system state then it will remain disallowed when the same system state occurs again. However, more realistically, social norms may be subject to changes. For example, a human society has different social norms in peacetime and wartime, and an autonomous multiagent system may have different social norms when exposed to different levels of cyber-attacks. This motivates us to propose a new definition of normative systems (in Section 3), to enable the representation of norms under multiple states (hence, *dynamic* normative systems). With a running example, we show that a dynamic normative system can be a necessity if a multiagent system wants to implement certain objectives.

We focus on two related autonomy issues[1]. The first is norm synthesis, which is to determine the existence of a normative system for the achievement of objectives. The success of this problem suggests the autonomy of the multiagent system with respect to the objectives, i.e., if all agents in the system choose to conform to the normative system[2], the objectives can be achieved. For static normative systems, norm synthesis problem is shown to be NP-complete [Shoham and Tennenholtz, 1995]. For our new, and more general, definition of normative systems, we show that it is EXPTIME-complete. This encouraging decidable result shows that the maximum number of normative states can be bounded.

The second is norm recognition, which can be seen as a successive step after deploying an autonomous multiagent systems (e.g., by norm synthesis). For deployed systems such as [Chalupsky *et al.*, 2001], it can be essential to allow new agents to join anytime. If so, it is generally expected that the new agent is able to recognise the current social norms after playing in the system for a while. Under this general description, we consider two subproblems related to the autonomy of the system and the new agent, respectively. The first one, whose complexity is in PTIME, tests whether the system, under the normative system, can be autonomous in ensuring that the new agent can eventually recognise the norms, no matter how it plays. If such a level of autonomy is unachievable, we may consider the second subproblem, whose success suggests that if the new agent is autonomous (in moving in a smart way) then it can eventually recognise the norms. We show that the second subproblem is PSPACE-complete.

[1]In this paper, we consider decision problems of these autonomy issues. The algorithms for the upper bounds in Theorem 1, 2, and 3 can be adapted to implement their related autonomy.

[2]The synthesised normative system is a common knowledge [Fagin *et al.*, 1995] to the agents.

2 Partial Observation Multiagent Systems

A multiagent system consists of a set of agents running in an environment [Fagin *et al.*, 1995]. At each time, every agent takes a local action independently, and the environment updates its state according to agents' joint action. We assume that agents have only partial observations over the system states, because in most real-world systems, agents either do not have the capability of observing all the information (e.g., an autonomous car on the road can only observe those cars in the surrounding area by its sensors or cameras, etc) or are not supposed to observe private information of other agents (e.g., a car cannot observe the destinations of other cars, etc).

Let Agt be a finite set of agents and $Prop$ be a finite set of atomic propositions. A finite multiagent system is a tuple $M = (S, \{Act_i\}_{i\in Agt}, \{L_i\}_{i\in Agt}, \{O_i\}_{i\in Agt}, I, T, \pi)$, where S is a finite set of environment states, Act_i is a finite set of local actions of agent $i \in Agt$ such that $Act = Act_1 \times ... \times Act_n$ is a set of joint actions, $L_i : S \to \mathcal{P}(Act_i) \setminus \{\emptyset\}$ provides on every state a nonempty set of local actions that are available to agent i, $I \subseteq S$ is a nonempty set of initial states, $T \subseteq S \times Act \times S$ is a transition relation such that for all $s \in S$ and $a \in Act$ there exists a state s' such that $(s, a, s') \in T$, $O_i : S \to O$ is an observation function for each agent $i \in Agt$ such that O is a set of possible observations, and $\pi : S \to \mathcal{P}(Prop)$ is an interpretation of the atomic propositions $Prop$ at the states. We require that for all states $s_1, s_2 \in S$ and $i \in Agt$, $O_i(s_1) = O_i(s_2)$ implies $L_i(s_1) = L_i(s_2)$, i.e., an agent can distinguish two states with different sets of next available actions.

Example 1 *We consider a business system with two sets of autonomous agents: the producer agents $P = \{p_1, ..., p_n\}$, and consumer agents $C = \{c_1, ..., c_m\}$. Let $Agt = P \cup C$. Each producer agent $p_j \in P$ produces a specific kind of goods with limited quantity each time. There can be more than one agents producing the same goods. We use $g_j \in G$ to denote the kind of goods that are produced by agent p_j, and $b_j \in \mathbb{N}$ to denote the number of goods that can be produced at a time. Every consumer agent $c_i \in C$ has a designated job which needs a set of goods to complete. It is possible that more than one goods of a kind are needed. We use r_i to denote the multiset of goods that are required by agent c_i.*

We use $rr_i \subseteq r_i$ to denote the multiset of remaining goods to be collected for c_i, $d_i \in G' = G \cup \{\bot\}$ to represent c_i's current demand, and $t_i \in P' = P \cup \{\bot\}$ to represent the producer agent from whom c_i is currently requesting goods. Every interaction of agents occurs in two consecutive rounds, and we use $k \in \{1, 2\}$ to denote the current round number.

Because g_j, b_j, r_i do not change their values in a system execution, we assume that they are fixed inputs of the system. The multiagent system M has the state space as

$$S = \{1, 2\} \times \Pi_{i\in\{1,...,m\}}\{(rr_i, d_i, t_i) \mid rr_i \subseteq r_i, d_i \in G', t_i \in P'\}$$

where the first component $\{1, 2\}$ is for the round number. The initial states are $I = \{1\} \times \Pi_{i\in\{1,...,m\}}\{(\emptyset, \bot, \bot)\}$.

The consumer agent c_i has a set of actions $Act_{c_i} = \{a_\bot\} \cup \{a_{p_j} \mid p_j \in P\}$. Intuitively, $a_\bot$ means that an agent does nothing, and the action a_{p_j} means that agent c_i sends a request to producer p_j for its goods. The producer agent p_j has a set of actions $Act_{p_j} = \{a_\bot\} \cup \{a_B \mid B \subseteq C, |B| \leq b_j\}$. Intuitively, the action a_B for B a subset of agents represents that agent p_j satisfies the requests from agents in B.

We use pseudocode to describe the transition relation. In the first round, i.e., $k = 1$, it can be described as follows.

R1a. all consumer agents c_i do the following sequential steps:

- *(a) if $rr_i = \emptyset$ then we let $rr_i = r_i$. Intuitively, this represents that agent c_i's job is repeated.*
- *(b) if $d_i = \bot$ then do the following: let $d_i \in rr_i$, choose an agent p_j such that $d_i = g_j$, and let $t_i = p_j$. Intuitively, if there is no current demand, then a new demand $d_i \in rr_i$ is generated, and c_i sends a request to a producer agent p_j who is producing goods d_i.*

R1b. all producer agents p_j execute action $a_\bot$, and let $k = 2$.

In the second round, i.e., $k = 2$, it can be described as follows.

R2a. all producer agents p_j do the following sequential steps:

- *(a) select a maximal subset B of agents such that $B \subseteq \{c_i \mid t_i = p_j\}$ and $|B| \leq b_j$. Intuitively, from the existing requests, the producer agent p_j selects a set of them according to its production capability.*
- *(b) for all agents c_i in B, let $rr_i = rr_i \setminus \{g_j\}$ and $d_i = t_i = \bot$. Intuitively, if a demand d_i is satisfied, then it is removed from rr_i and we let $d_i = t_i = \bot$.*

R2b. all consumer agents execute action $a_\bot$, and let $k = 1$.

We use "var = val", for var a variable and val one of its values, to denote an atomic proposition. Then the labelling function π can be defined naturally over the states. The observation O_i will be discussed in Section 6.

We provide a simple instantiation of the system[3]*. Let $n = 2$, $G = \{g_1, g_2\}$, $b_1 = b_2 = 1$ (two agents produce goods one at each time), $m = 3$, $r_1 = \{g_1\}$, $r_2 = \{g_2\}$ and $r_3 = \{g_1, g_2\}$ (three consumers with the required goods). From the initial state $s_0 = (1, (\emptyset, \bot, \bot), (\emptyset, \bot, \bot), (\emptyset, \bot, \bot))$, we may have the following two states such that $(s_0, (a_\bot, a_\bot, a_{p_1}, a_{p_2}, a_{p_1}), s'_2) \in T$ and $(s'_2, (a_{\{c_1\}}, a_{\{c_2\}}, a_\bot, a_\bot, a_\bot), s'_1) \in T$:*

$$\begin{cases} s'_2 = (2, (\{g_1\}, g_1, p_1), (\{g_2\}, g_2, p_2), (\{g_1, g_2\}, g_1, p_1)), \text{and} \\ s'_1 = (1, (\{\}, \bot, \bot), (\{\}, \bot, \bot), (\{g_1, g_2\}, g_1, p_1)) \end{cases}$$

3 Dynamic Normative Systems

The following is our new definition of normative systems.

Definition 1 *A dynamic normative system of a multiagent system $M = (S, \{Act_i\}_{i\in Agt}, \{L_i\}_{i\in Agt}, \{O_i\}_{i\in Agt}, I, T, \pi)$ is a tuple $N_M = (Q, \delta_n, \delta_u, q_0)$ such that Q is a set of normative states, $\delta_n : S \times Q \to \mathcal{P}(Act)$ is a function specifying, for each environment state and each normative state, a set of joint actions that are disallowed, $\delta_u : Q \times S \to Q$ is a function specifying the update of normative states according to the changes of environment states, and q_0 is the initial normative state.*

A (static) normative system in the literature can be seen as a special case of our definition where the only normative state is q_0. In such case, we have $Q = \{q_0\}$, $\delta_u(q_0, s) = q_0$ for all $s \in S$, and can therefore write the function δ_n as function

[3]The instantiation is simply to ease the understanding of the definitions in Example 1 and 2. The conclusions for the example system (i.e., Proposition 1, 2, 3, 4, 5) are based on the general definition.

$\delta : S \to \mathcal{P}(Act)$. It is required that the function δ_n (and thus δ) does not completely eliminate agents' joint actions, i.e., $\delta_n(s,q) \subset \Pi_{i\in Agt} L_i(s)$ for all $s \in S$ and $q \in Q$.

We give two dynamic normative systems.

Example 2 *Let M be the multiagent system given in Example 1. Let s_1 and s_2 range over those environmental states such that $k = 1$ and $k = 2$, respectively.*

The normative system $N^1_M = (Q^1, \delta^1_n, \delta^1_u, q^1_0)$ is such that:

- $Q^1 = \Pi_{p_j\in P}\{1,...,m\}$*, where each producer maintains a number indicating the consumer whose requirement must be satisfied in this normative state,*
- $\delta^1_n(s_1,q) = \emptyset$*, i.e., no joint actions are disallowed on s_1, and $(a_{B_1},...,a_{B_n},a_\perp,...,a_\perp) \in \delta^1_n(s_2,(y_1,...,y_n))$ if there exists $j \in \{1,...,n\}$ such that $B_j \subseteq C$ and $c_{y_j} \notin B_j$. Intuitively, for producer agent p_j, an action a_{B_j} is disallowed on the second round if B_j does not contain the consumer c_{y_j} who is needed to be satisfied in this round.*
- $\delta^1_u(q,s_2) = q$ *and* $\delta^1_u((y_1,...,y_n),s_1) = (y'_1,...,y'_n)$ *such that* $y'_j = (y_j \mod m) + 1$ *for* $j \in \{1,...,n\}$*; intuitively, the normative state increments by 1 and loops forever.*
- $q^1_0 = (1,...,n)$*, i.e., producer agents p_j start from c_j.*

For the instantiation in Example 1, we have that

- $Q^1 = \{1,2,3\}\times\{1,2,3\}$, $q^1_0 = (1,2)$,
- $(a_{B_1},a_{B_2},a_\perp,a_\perp,a_\perp) \in \delta^1_n(s'_2,(1,2))$ *if either* $B_1 \in \{\emptyset,\{c_2\},\{c_3\},\{c_2,c_3\}\}$ *or* $B_2 \in \{\emptyset,\{c_1\},\{c_3\},\{c_1,c_3\}\}$,
- $\delta^1_u((1,2),s_1) = (2,3)$, $\delta^1_u((2,3),s_1) = (3,1)$.

We define another normative system $N^2_M = (Q^2,\delta^2_n,\delta^2_u,q^2_0)$ by extending the number maintained by each producer into a first-in-first-out queue so that the ordering between consumers who have sent the requests matters. That is, we have $Q^2 = \Pi_{p_j\in P}(\{\epsilon\} \cup \{i_1...i_k \mid k \in \{1,...,m\}, i_x \in C$ for $1 \le x \le k\})$ where the symbol ϵ denotes an empty queue, and $q^2_0 = \Pi_{p_j\in P}\{\epsilon\}$ which means that producers start from empty queues. The functions δ^2_n and δ^2_u can be adapted from N^1_M, and details are omitted here due to space limit.

The following captures the result of applying a normative system on a multiagent system, which is essentially a product of these two systems.

Definition 2 *Let M be a multiagent system and N_M a normative system on M, the result of applying N_M on M is a Kripke structure $K(N_M) = (S^\dagger, I^\dagger, T^\dagger, \pi^\dagger)$ such that*

- $S^\dagger = S \times Q$ *is a set of states,*
- $I^\dagger = I \times \{q_0\}$ *is a set of initial states,*
- $T^\dagger \subseteq S^\dagger \times S^\dagger$ *is such that, for any two states (s_1,q_1) and (s_2,q_2), we have $((s_1,q_1),(s_2,q_2)) \in T^\dagger$ if and only if, (1) there exists an action $a \in Act$ such that $(s_1,a,s_2) \in T$ and $a \notin \delta_n(s_1,q_1)$, and (2) $q_2 = \delta_u(q_1,s_2)$. Intuitively, the first condition specifies the enabling condition to transit from state s_1 to state s_2 by taking a joint action a which is allowed in the normative state q_1. The second condition specifies that the transition relation needs to be consistent with the changes of normative states.*
- $\pi^\dagger : S^\dagger \to \mathcal{P}(Prop)$ *is such that* $\pi^\dagger((s,q)) = \pi(s)$.

Example 3 *For the instantiation, in the structure $K(N^1_M)$, we have $((s_0,(1,2)),(s'_2,(1,2))),((s'_2,(1,2)),(s'_1,(2,3))) \in T^\dagger$. The latter is because $(s'_2,(a_{\{c_1\}},a_{\{c_2\}},a_\perp,a_\perp,a_\perp),s'_1) \in T$, $\{c_1\} \notin \{\emptyset,\{c_2\},\{c_3\},\{c_2,c_3\}\}$, and $\{c_2\} \notin \{\emptyset,\{c_1\},\{c_3\},\{c_1,c_3\}\}$.*

On the other hand, for $a = (a_{\{c_1\}},a_{\{c_3\}},a_\perp,a_\perp,a_\perp)$ and $s''_1 = (1,(\{\},\perp,\perp),(\{g_2\},g_2,p_2),(\{g_1\},\perp,\perp))$, we have $(s'_2,a,s''_1) \in T$ but $((s'_2,(1,2)),(s''_1,(2,3))) \notin T^\dagger$. This is because, for p_2, it is required to make c_2 as its current priority according to the normative state, and cannot choose to satisfy c_3 instead.

We remark that, the normative system, as many current formalisms, imposes hard constraints on the agents' behaviour. As stated in e.g., [Boella *et al.*, 2006], social norms may be soft constraints that agents can choose to comply with or not. To accommodate soft social norms, we can redefine the function δ_n as $\delta_n : S \times Q \times Act \to U$ to assign each joint action a cost utility for every agent, on each environment state and normative state. With this definition, norms become soft constraints: agents can choose to take destructive actions, but are encouraged to avoid them due to their high costs. The objective language to be introduced in the next section also needs to be upgraded accordingly to express properties related to the utilities. We leave such an extension as a future work.

4 Objective Language

To specify agents' and the system's objectives, we use temporal logic CTL [Clarke *et al.*, 1999] whose syntax is as follows.

$$\phi ::= p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid EX\phi \mid E(\phi_1\, U\, \phi_2) \mid EG\phi$$

where $p \in Prop$. Intuitively, formula $EX\phi$ expresses that ϕ holds at some next state, $E(\phi_1\, U\, \phi_2)$ expresses that on some path from current state, ϕ_1 holds until ϕ_2 becomes true, and $EG\phi$ expresses that on some path from current state, ϕ always holds. Other operators can be obtained as usual, e.g., $EF\phi \equiv E(True\, U\, \phi)$, $AG\phi \equiv \neg E(True\, U\, \neg\phi)$, $AF\phi = \neg EG\neg\phi$ etc.

A path in a Kripke structure $K(N_M)$ is a sequence $s_0 s_1 ...$ of states such that $(s_i, s_{i+1}) \in T^\dagger$ for all $i \ge 0$. The semantics of the language is given by a relation $K(N_M), s \models \phi$ for $s \in S^\dagger$, which is defined inductively as follows [Clarke *et al.*, 1999]:

1. $K(N_M), s \models p$ if $p \in \pi^\dagger(s)$,
2. $K(N_M), s \models \neg\phi$ if not $K(N_M), s \models \phi$,
3. $K(N_M), s \models \phi_1 \vee \phi_2$ if $K(N_M), s \models \phi_1$ or $K(N_M), s \models \phi_2$,
4. $K(N_M), s \models EX\phi$ if there exists a state $s' \in S^\dagger$ such that $(s,s') \in T^\dagger$ and $K(N_M), s' \models \phi$,
5. $K(N_M), s \models E(\phi_1\, U\, \phi_2)$ if there exists a path $s_0 s_1 \ldots$ and a number $n \ge 0$ such that $s_0 = s$, $K(N_M), s_k \models \phi_1$ for $0 \le k \le n-1$ and $K(N_M), s_n \models \phi_2$,
6. $K(N_M), s \models EG\phi$ if there exists a path $s_0 s_1 \ldots$ such that $s_0 = s$ and $K(N_M), s_k \models \phi$ for all $k \ge 0$.

The verification problem, denoted as $K(N_M) \models \phi$, is, given a multiagent system M, its associated normative system N_M, and an objective formula ϕ, to decide whether $K(N_M), s \models \phi$ for all $s \in I^\dagger$. The norm synthesis problem is, given a system M and an objective formula ϕ, to decide the existence of a normative system N_M such that $K(N_M) \models \phi$. The norm recognition problem will be defined in Section 6. For the measurement of the complexity, we take the standard assumption that

the sizes of the multiagent system and the normative system are measured with the number of states, and the size of the objective formula is measured with the number of operators.

Example 4 *For the system in Example 1, interesting objectives expressed in CTL may include*

$$\phi_1 \equiv \bigwedge_{i\in C}\bigwedge_{j\in P} AG\,(t_i = p_j \Rightarrow EF\, d_i = \bot)$$

which says that it is always the case that if there is a request from a consumer c_i to a producer p_j (i.e., $t_i = p_j$), then the request is possible to be satisfied eventually (i.e., $d_i = \bot$), and

$$\phi_2 \equiv \bigwedge_{i\in C}\bigwedge_{j\in P} AG\,(t_i = p_j \Rightarrow AF\, d_i = \bot)$$

which says that it is always the case that if there is a request from a consumer c_i to a producer p_j, then on all the paths the request will eventually be satisfied. Both ϕ_1 and ϕ_2 are liveness objectives that are important for an ecosystem to guarantee that no agent can be starving forever. The objective ϕ_2 is stronger than ϕ_1, and their usefulness is application-dependent. The following proposition shows that static normative systems are insufficient to guarantee the satisfiability of the objectives in this ecosystem.

Proposition 1 *There exists an instance of a multiagent system M such that, for all static normative systems N_M, we have that $K(N_M) \not\models \phi_1 \wedge \phi_2$.*

The proof idea is based on the following simple case. Assume that there are one producer p_1, such that $b_1 = 1$, and two consumers c_1 and c_2, such that $r_1 = r_2 = \{g_1\}$. There only exist the following three static normative systems which have different restrictions on an environment state $s_2 = (2, (\{g_1\}, g_1, p_1), (\{g_1\}, g_1, p_1))$: (Recall that q_0 is the only normative state in static normative systems.)

- *N^3_M is such that $\delta^3_n(s_2, q_0) = \{a_{\{c_1\}}\}$, i.e., c_1 is not satisfied.*
- *N^4_M is such that $\delta^4_n(s_2, q_0) = \{a_{\{c_2\}}\}$, i.e., c_2 is not satisfied.*
- *N^5_M is such that $\delta^5_n(s_2, q_0) = \emptyset$, i.e., no restriction.*

We can see that $K(N^h_M) \not\models \phi_1$ for $h \in \{3, 4\}$ and $K(N^5_M) \not\models \phi_2$. The former is because one of the agents is constantly excluded from being satisfied. For the latter, there exists an infinite path $s_0(s_2 s_1)^\infty$ such that $s_0 = (1, (\emptyset, \bot, \bot), (\emptyset, \bot, \bot))$ is an initial state, s_2 is given as above, and $s_1 = (1, (\emptyset, \bot, \bot), (\{g_1\}, g_1, p_1))$ is the state on which consumer c^1_i's requirement is satisfied. On this path, the requirement from c^2_i is never satisfied. On the other hand, *for the dynamic normative systems in Example 2, all the consumers' requests can be satisfied, so we have the following conclusion.*

Proposition 2 *Given a system M and a normative system N^1_M or N^2_M, we have that $K(N^h_M) \models \phi_1 \wedge \phi_2$ for $h \in \{1, 2\}$.*

The above example suggests that, to achieve some objectives, we need *dynamic* normative systems to represent the changes of social norms under different circumstances. Then, another question may follow about the maximum number of normative states. The dynamic system could be uninteresting if the number of states can be infinite. Fortunately, in the next section, we show with the complexity result that, for objectives expressed with CTL formulas, in the worst case, an exponential number of normative states are needed.

5 The Complexity of Norm Synthesis

We have the following result for norm synthesis.

Theorem 1 *The norm synthesis problem is EXPTIME-complete, with respect to the sizes of the system and the objective formula.* [4]

Proof idea for the upper bound: EXPTIME Membership
We use an automata-theoretic approach. Given a system M, we construct a Büchi tree automaton $A_M = (\Sigma, D, Q, \delta, q_0, Q)$, by a variant of the approach in [Kupferman and Vardi, 1996], such that $Q = S \times \{\top, \vdash, \bot\}$ and $\delta : Q \times \Sigma \times D \to 2^{Q^*}$ a transition function. Given a CTL formula ϕ and a set $D \subset \mathbb{N}$ with a maximal element k, there exists a Büchi tree automaton $A_{D,\neg\phi}$ that accepts exactly all the tree models of $\neg\phi$ with branching degrees in D. By [Vardi and Wolper, 1986], the size of $A_{D,\neg\phi}$ is $O(2^{k\cdot|\phi|})$. Then we show that the norm synthesis problem is equivalent to an unsuccessful result from checking the emptiness of the product automaton $A_M \times A_{D,\neg\phi}$. The checking of emptiness of Büchi tree automaton can be done in quadratic time, so the norm synthesis can be done in exponential time.

Proof idea for the lower bound: EXPTIME Hardness
We reduce from the problem of a linearly bounded alternating Turing machine (LBATM) accepting an empty input tape, which is known to be EXPTIME-complete [Chandra *et al.*, 1980]. An alternating Turing machine AT is a tuple $(Q, \Gamma, \tau, q_0, g)$ where Q is a finite set of states, Γ is a finite set of tape symbols including a blank symbol $\bot$, $\tau : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{-1, +1\})$ is a transition function, $q_0 \in Q$ is an initial state, $g : Q \to \{\forall, \exists, accept, reject\}$ specifies the type of each state. An LBATM is an ATM which uses m tape cells for a Turing machine description of size m.

We construct a multiagent system $M(AT)$ of a single agent $\exists$, such that the state space S consists of three sets $(Q \times L)$, $(Q \times L \times \Gamma)$, and $(Q \times L \times L \times \Gamma)$. Each transition $(r, c, d) \in \tau(q, b)$ of AT is simulated by three consecutive transitions in $M(AT)$, moving from states in one set to states in the next set. The basic idea of the transition relation T is to use agent $\exists$ to move on $\exists$ states and treat movements on $\forall$ states as nondeterminism. Then the function δ_n of normative systems can be imposed to restrict the behaviour of the agent $\exists$.

For the objectives, we use formula $\phi \wedge AFacc$, where ϕ is a CTL formula expressing the correct behaviour of the transition relation T with respect to τ, and $AFacc$ expresses that all paths are accepting. Then the norm synthesis problem on $M(AT)$ can be shown to be equivalent to the acceptance of the automata AT. Therefore, it is EXPTIME hard.

6 Agent Recognition of Social Norms

For a multiagent system to be autonomous without human intervention, it is important that it can maintain its functionality when new agents join or old agents leave. For a new agent to join and function well, it is essential that it is capable of recognising the social norms that are currently active. As stated in the previous sections, the agent has only partial observation over the system state, and is not supposed to observe the social norms. On the other hand, it is also unrealistic to

[4] Refer to http://arxiv.org/abs/1604.05086 for full proofs.

assume that the agent does not know anything about the social norms of the system it is about to join. Agent is designed to have a set of prescribed capabilities and is usually supposed to work within some specific scenarios. Therefore, the actual situation can be that, the agent knows in prior that there are a set of possible normative systems, one of which is currently applied on the multiagent system. We remark that, assuming a set of normative systems does not weaken the generality of the setting, because Theorem 1 implies that there are a finite number of possible normative systems (subject to a bisimulation relation between Kripke structures). This situation naturally leads to the following two new problems:

- (**NC$_1$**) to determine whether the agent can always recognise which normative system is currently applied; and
- (**NC$_2$**) to determine whether the agent can find a way to recognise which normative system is currently applied.

The successful answer to the problem NC_1 implies the successful answer to the problem NC_2, but not vice versa. Intuitively, the successful answer of NC_1 implies a high-level autonomy of the system that the new agent can be eventually incorporated into the system no matter how it behaves. We assume that once learned the social norms the new agent will behave accordingly. If such an autonomy of the system cannot be achieved, the successful answer of NC_2 implies a high-level autonomy of the agent that, by moving in a smart way, it can eventually recognise the social norms.

We formalise the problems first. Let Ψ be a set of possible normative systems defined on a multiagent system, $Path(K(N))$ be the set of possible paths of the Kripke structure $K(N)$ for $N \in \Psi$. We assign every normative system in Ψ a distinct index, denoted as $ind(N)$. This index is attached to every path $\rho \in Path(K(N))$, and let $ind(\rho) = ind(N)$.

Let the new agent be x such that $x \notin Agt$ and its observation function be O_x. For any state $(s, q) \in S^\dagger$, we define a projection function $\widehat{(s,q)} = s$. So $\widehat{\rho}$ is the projection of a path ρ of a Kripke structure to the associated multiagent system. We extend O_x to the paths of Kripke structure $K(N)$ as follows: $O_x(\rho s^\dagger) = O_x(\rho) \cdot O_x(\widehat{s^\dagger})$ for $\rho \in Path(K(N))$ and $s^\dagger \in S^\dagger$. We have $O_x(\epsilon) = \epsilon$, which means when a path is empty, the observation is also empty. We also define its inverse O_x^{-1} which gives a sequence o of observations, returns a set of possible paths ρ on which agent x's observations are o, i.e.,

$$O_x^{-1}(o) = \{\rho \in Path(K(N)) \mid O_x(\rho) = o, N \in \Psi\}.$$

W.l.o.g., we assume that $N_0 \in \Psi$ is the active normative system. Let $\mathbb{N}$ be the set of natural numbers, we have

Definition 3 *NC_1 problem is the existence of a number $k \in \mathbb{N}$ such that for all paths $\rho \in Path(K(N_0))$ such that $|\rho| \geq k$, we have that $\rho' \in O_x^{-1}(O_x(\rho))$ implies that $ind(\rho') = ind(\rho)$.*

NC_2 problem is the existence of a path $\rho \in Path(K(N_0))$ such that for all $\rho' \in O_x^{-1}(O_x(\rho))$ we have $ind(\rho') = ind(\rho)$.

Intuitively, NC_1 states that as long as the path is long enough, the new agent can eventually know that the active normative system is N_0. That is, no matter how the new agent behaves, it can eventually recognise the current normative system. On the other hand, NC_2 states that such a path exists (but not necessarily for all paths). That is, to recognise the normative system, the new agent needs to move smartly.

Example 5 *For the system in Example 1, we assume a new consumer agent c_v such that $v = m + 1$. Also, we define $O_{c_v}(s) = \{c_i \mid c_i \in C, t_i(s) = t_v(s) \neq \bot\}$ for all $s \in S$. Intuitively, the agent c_v keeps track of the set of agents that are currently having the same request. Unfortunately, we have*

Proposition 3 *There exists an instance of system M such that under the set $\Psi = \{N_M^1, N_M^2\}$ of normative systems, both NC_1 and NC_2 are unsuccessful.*

This can be seen from a simple case where there are a single producer p_1 with $b_1 = 1$ and a set of consumers C such that $r_i = \{g_1\}$ for all $c_i \in C$. For the initial state, every consumer sends its request to p_1, so $\{c_i \mid t_i = p_1\} = C$. For any path ρ_1 of $K(N_M^1)$ and ρ_2 of $K(N_M^2)$, we have $\widehat{\rho_1} = \widehat{\rho_2} = s_0 s_2 s_1^1 s_2 ... s_1^m s_2 s_1^v s_2 s_1^1 ...$ where $s_0 = (1, (\emptyset, \bot, \bot), ..., (\emptyset, \bot, \bot))$, $s_2 = (2, (\{g_1\}, g_1, p_1), ..., (\{g_1\}, g_1, p_1))$, and s_1^i is different with s_2 in c_i's local state, e.g., $s_1^1 = (1, (\{\}, \bot, \bot), ..., (\{g_1\}, g_1, p_1))$. And therefore $O_{c_v}(\rho_1) = O_{c_v}(\rho_2) = \emptyset C(C \setminus \{c_1\}) C(C \setminus \{c_2\}) ...$, i.e., the agent c_v's observations are always the same[5]. That is, the new agent c_v finds that for ρ_1, the single path on $K(N_M^1)$, there are $\rho_2 \in O_{c_v}^{-1}(O_{c_v}(\rho_1))$ and $ind(\rho_1) \neq ind(\rho_2)$. Therefore, neither NC_1 nor NC_2 can be successful in such a case.

The reasons for the above result may come from either the insufficient capabilities of the agent or the designing of normative systems. We explain this in the following example.

Example 6 *First, consider that we increase the capabilities of the new agent by updating the rule R2b in Section 2.*

R2b'. the new agent c_v may cancel its current request by letting $d_v = t_v = \bot$; all other consumer agents execute action $a_\bot$; and let $k = 1$.

With this upgraded capabilities of the new agent, the NC_2 can be successful. The intuition is that, by canceling and re-requesting for at least twice, the ordering of consumer agents whose requests are satisfied can be different in two normative systems: with N_M^2, there are other agents c_i between c_m and c_v, but with N_M^1, their requests are always satisfied consecutively. Note that, by its new capabilities, c_v can always choose a producer agent which have more than 2 existing and future requests (Assuming that $n \ll m$, which is usual for a business ecosystem).

Proposition 4 *With the new rule R2b', the NC_2 problem is successful on system M and the set $\Psi = \{N_M^1, N_M^2\}$.*

However, the NC_1 problem is still unsuccessful, because the agent c_v may not move in such a smart way. For this, we replace N_M^1 with $N_M^6 = (Q^1, \delta_n^1, \delta_u^6, q_0^1)$ such that

- *$\delta_u^6(q, s_2) = q$ and $\delta_u^6((y_1, ..., y_n), s_1) = (y'_1, ..., y'_n)$, s.t. $y'_j = ((y_j + 1) \mod m) + 1$ for $j \in \{1, ..., n\}$. Intuitively, the normative state increments by 2 (modulo m).*

Proposition 5 *Both NC_1 and NC_2 problems are successful on system M and the set $\Psi = \{N_M^2, N_M^6\}$.*

[5]We reasonably assume that, for N_M^1, when a producer sees c_v, it will adjust its range in normative states from $\{1, ..., m\}$ to $\{1, ..., m, v\}$.

7 The Complexity of Norm Recognition

The discussion in the last section clearly shows that, the two norm recognition problems are non-trivial. It is therefore useful to study if there exist efficient algorithms that can decide them automatically. In this section, we show a somewhat surprising result that the determination of NC_1 problem can be done in PTIME, while it is PSPACE-complete for NC_2 problem. Assume that the size of the set Ψ is measured over both the number of normative systems and the number of normative states. We have the following conclusions.

Theorem 2 *The NC_1 problem can be decided in PTIME, with respect to the sizes of the system and the set Ψ.*

The proof idea is as follows. The unsuccessful answer to the NC_1 problem can be obtained by the existence of two infinite paths $\rho \in Path(K(N_0))$ and $\rho' \in O_x^{-1}(O_x(\rho))$ such that $ind(\rho') \neq ind(\rho)$. To check this, we construct a product system $K(N_0) \times K(N_1)$ for $N_1 \in \Psi \setminus \{N_0\}$. The product system synchronises the behaviour of the two Kripke structures $K(N_0)$ and $K(N_1)$ such that the observations are always the same. Then the existence of an infinite path in the product system is equivalent to the existence of ρ and ρ'. Further, the existence of an infinite path is equivalent to the existence of reachable strongly connected components. The latter can be decided in PTIME by the Tarjan's algorithm [Tarjan, 1972].

Theorem 3 *The NC_2 problem is PSPACE-complete, with respect to the sizes of the system and the set Ψ.*

Proof idea for the upper bound: PSPACE Membership

The upper bound is obtained by having a nondeterministic algorithm which takes a polynomial size of space, i.e., it is in NPSPACE=PSPACE. The algorithm starts by guessing a set of initial states of the structures $\{K(N) \mid N \in \Psi\}$ on which agent x has the same observation. It then continuously guesses the next set of states such that they are reachable in one step from some state in the current set and on which agent x has the same observation. If this guess can be done infinitely then the NC_2 problem is successful. This infinite number of guesses can be achieved with a finite number of guesses, by adapting the approach of LTL model checking.

Proof idea for the lower bound: PSPACE Hardness

It is obtained by a reduction from the problem of deciding if, for a given nondeterministic finite state automaton A over an alphabet Σ, the language $L(A)$ is equivalent to the universal language Σ^*. Let $A = (Q, q_0, \delta, F)$ be an NFA such that Q is a set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is a transition function, and $F \subseteq Q$ is a set of final states. We construct a system $M(A)$ which consists of two subsystems, one of them simulates the behaviour of A and the other simulates the behaviour of the language Σ^*. The subsystems are reachable from an initial state s_0 by two actions a_1 and a_2 respectively. On the system $M(A)$, we have two normative systems whose only difference is on the state s_0: N_0 disallows action a_1 and N_1 disallows action a_2. Therefore, the universality of the NFA A is equivalent to the unsuccessful answer to the NC_2 problem on $M(A)$ and $\Psi = \{N_0, N_1\}$.

8 Related Work

Normative multiagent systems have attracted many research interests in recent years, see e.g., [Boella *et al.*, 2006; Criado *et al.*, 2011] for comprehensive reviews of the area. Here we can only review some closely related work.

Norm synthesis for static normative systems. As stated, most current formalisms of normative systems are static. [Shoham and Tennenholtz, 1995] shows that this norm synthesis problem is NP-complete. [Christelis and Rovatsos, 2009] proposes a norm synthesis algorithm in declarative planning domains for reachability objectives, and [Morales *et al.*, 2013] considers the on-line synthesis of norms. [Bulling and Dastani, 2011] considers the norm synthesis problem by conditioning over agents' preferences, expresses as pairs of LTL formula and utility, and a normative behaviour function.

Changes of normative system. [Knobbout *et al.*, 2014] represents the norms as a set of atomic propositions and then employs a language to specify the update of norms. Although the updates are parameterised over actions, no considerations are taken to investigate, by either verification or norm synthesis, whether the normative system can be imposed to coordinate agents' behaviour to secure the objectives of the system.

Norm recognition. Norm recognition can be related to the norm learning problem, which employs various approaches, such as data mining [Savarimuthu *et al.*, 2013] and sampling and parsing [Oren and Meneguzzi, 2013; Cranefield *et al.*, 2015], for the agent to learn social norms by observing other agents' behaviour. On the other hand, our norm recognition problems are based on formal verification, aiming to decide whether the agents are designed well so that they can recognise the current normative system from a set of possible ones. We also study the complexity of them.

Application of social norms Social norms are to regulate the behaviour of the stakeholders in a system, including sociotechnical system [Chopra and Singh, 2016] which has both humans and computers. They are used to represent the commitments (by e.g., business contracts, etc) between humans and organisations. The dynamic norms of this paper can be useful to model more realistic scenarios in which commitments may be changed with the environmental changes.

9 Conclusions

In the paper, we first present a novel definition of normative systems, by arguing with an example that it can be a necessity to have multiple normative states. We study the complexity of two autonomy issues related to normative systems. The decidability (precisely, EXPTIME-complete) of norm synthesis is an encouraging result, suggesting that the maximum number of normative states is bounded for CTL objectives. For the two norm recognition subproblems, one of them is, surprisingly, in PTIME and the other is PSPACE-complete. Because the first one suggests a better level of autonomy, to see if an agent can recognise the social norms, we can deploy a PTIME algorithm first. If it fails, we may apply a PSPACE algorithm to check the weaker autonomy.

Acknowledgement The authors would like to thank the reviewers for their constructive comments which are help-

ful in improving the paper. The authors are supported by ERC Advanced Grant VERIWARE, EPSRC Mobile Autonomy Programme Grant EP/M019918/1, National Natural Science Foundation of China Grant 61572234 and 61472369, Fundamental Research Funds for the Central Universities of China Grant 21615441, and ARC Grant DP150101618.

References

[Ågotnes and Wooldridge, 2010] Thomas Ågotnes and Michael Wooldridge. Optimal social laws. In *AAMAS 2010*, pages 667–674, 2010.

[Ågotnes *et al.*, 2007] Thomas Ågotnes, Wiebe van der Hoek, and Michael Wooldridge. Normative system games. In *AAMAS 2007*, 2007.

[Boella *et al.*, 2006] Guido Boella, Leendert van der Torre, and Harko Verhagen. Introduction to normative multiagent systems. *Computational and Mathematical Organization Theory*, 12(2-3):71–79, 2006.

[Bulling and Dastani, 2011] Nils Bulling and Mehdi Dastani. Verifying normative behaviour via normative mechanism design. In *IJCAI 2011*, pages 103–108, 2011.

[Chalupsky *et al.*, 2001] Hans Chalupsky, Yolanda Gil, Craig A. Knoblock, Kristina Lerman, Jean Oh, David V. Pynadath, Thomas A. Russ, and Milind Tambe. Electric elves: Applying agent technology to support human organizations. In *IAAI 2001*, pages 51–58, 2001.

[Chandra *et al.*, 1980] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1980.

[Chopra and Singh, 2016] Amit K. Chopra and Munindar P. Singh. From social machines to social protocols: Software engineering foundations for sociotechnical systems. In *WWW 2016*, pages 903–914, 2016.

[Christelis and Rovatsos, 2009] George Christelis and Michael Rovatsos. Automated norm synthesis in an agent-based planning environment. In *AAMAS 2009*, pages 161–168, 2009.

[Clarke *et al.*, 1999] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.

[Cranefield *et al.*, 2015] Stephen Cranefield, Tony Savarimuthu, Felipe Meneguzzi, and Nir Oren. A bayesian approach to norm identification (extended abstract). In *AAMAS 2015*, pages 1743–1744, 2015.

[Criado *et al.*, 2011] N. Criado, E. Argente, and V. Botti. Open issues for normative multi-agent systems. *AI Communications*, 2011.

[Fagin *et al.*, 1995] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[Fisher *et al.*, 2013] Michael Fisher, Louise Dennis, and Matt Webster. Verifying autonomous systems. *Communications of the ACM*, 56(9):84–93, 2013.

[Knobbout *et al.*, 2014] Max Knobbout, Mehdi Dastani, and John-Jules Ch. Meyer. Reasoning about dynamic normative systems. In *JELIA 2014*, pages 628–636, 2014.

[Kupferman and Vardi, 1996] Orna Kupferman and Moshe Y. Vardi. Module checking. In *8th International Conference on Computer Aided Verification (CAV1996)*, pages 75–86, 1996.

[Morales *et al.*, 2013] Javier Morales, Maite Lopez-Sanchez, Juan A. Rodriguez-Aguilar, Michael Wooldridge, and Wamberto Vasconcelos. Automated synthesis of normative systems. In *AAMAS 2013*, pages 483–490, 2013.

[Oren and Meneguzzi, 2013] Nir Oren and Felipe Meneguzzi. Norm identification through plan recognition. In *COIN 2013@AAMAS*, 2013.

[Savarimuthu *et al.*, 2013] B. T. R. Savarimuthu, S. Cranefield, M. A. Purvis, and M. K. Purvis. Identifying prohibition norms in agent societies. *Artificial intelligence and law,*, 21(1):1–46, 2013.

[Shoham and Tennenholtz, 1992] Yoav Shoham and Moshe Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *AAAI 1992*, pages 276–281, 1992.

[Shoham and Tennenholtz, 1995] Yoav Shoham and Moshe Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.

[Tarjan, 1972] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[van der Hoek *et al.*, 2007] Wiebe van der Hoek, Mark Roberts, and Michael Wooldridge. Social laws in alternating time: effectiveness, feasibility, and synthesis. *Synthese*, 156(1):1–19, 2007.

[Vardi and Wolper, 1986] Moshe Y. Vardi and Pierre Wolper. Automata-theoretic techniques for modal logics of programs. *J. Comput. Syst. Sci.*, 32(2):183–221, 1986.

[Wooldridge and van der Hoek, 2005] Michael Wooldridge and Wiebe van der Hoek. On obligations and normative ability: Towards a logical analysis of the social contract. *Journal of Applied Logic*, 3:396–420, 2005.