Social Laws in Alternating Time: Effectiveness, Feasibility, and Synthesis

Wiebe van der Hoek Mark Roberts Michael Wooldridge
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, UK
{wiebe,mark,mjw}@csc.liv.ac.uk

Abstract

Since it was first proposed by Moses, Shoham, and Tennenholtz, the *social laws* paradigm has proved to be one of the most compelling approaches to the offline coordination of multiagent systems. In this paper, we make four key contributions to the theory and practice of social laws in multiagent systems. First, we show that the *Alternating-time Temporal Logic* of Alur, Henzinger, and Kupferman provides an elegant and powerful framework within which to express and understand social laws for multiagent systems. Second, we show that the *effectivess*, *feasibility*, and *synthesis* problems for social laws may naturally be framed as ATL model checking problems, and that as a consequence, existing ATL model checkers may be applied to these problems. Finally, we show that the complexity of the feasibility problem in our framework is no more complex in the general case than that of the corresponding problem in the Shoham-Tennenholtz framework (it is NP-complete). Finally, we show how our basic framework can easily be extended to permit social laws in which constraints on the legality or otherwise of some action may be explicitly required. We illustrate the concepts and techniques developed by means of a running example.

1 Introduction

One of the defining problems in multiagent systems research is that of *coordination* – managing the interdependencies between the actions of multiple interacting agents [7, 18]. Techniques to coordinate activity in multiagent systems may be broadly categorised by whether they are "online" or "offline". Online techniques aim to equip agents with the ability to dynamically coordinate their activities, for example by explicitly reasoning about coordination at run-time. In contrast, offline techniques aim at developing a coordination regime at design-time, and hardwiring this regime into a system for use at run-time. There are arguments in favour of both approaches: the former is potentially more flexible, and may be more robust against unanticipated events, while the latter approach benefits from offline reasoning about coordination, thereby reducing the run-time decision-making burden on agents [18].

One of the most successful approaches to "offline" coordination is the *social laws* paradigm, introduced largely through the work of Shoham, Tennenholtz, and Moses [15, 12, 16, 17]. A social law can be understood as a set of rules imposed upon a multiagent system with the goal of ensuring that some desirable global (coordination) behaviour will result. Social laws work by *constraining* the behaviour of the agents in the system – by *forbidding* agents from performing certain actions in certain circumstances. In this sense, they can be understood as *norms* [5]. Shoham et al investigated a number of issues surrounding the development and use of social laws, including the computational complexity of their synthesis, and the possibility of the development of social laws or conventions by the agents within the system themselves.

In this paper, we make four key contributionms to the literature on social laws. First, we demonstrate that *Alternating-time Temporal Logic* (ATL) – the temporal logic of cooperation developed by Alur, Henzinger, and Kupferman [1] – provides a rich and natural technical framework within which to investigate social laws and their properties. Second, we show that the *effectiveness*, *feasibility*, and *synthesis* problems for social laws in the ATL framework can be posed as ATL *model checking* problems [4], and that existing model checkers for ATL can hence be directly applied to these problems. Third, we show that, despite its apparent expressive power, the complexity of the feasibility problem in our framework is no more complex than the corresponding problem in the Shoham-Tennenholtz framework: it is NP-complete. We also identify cases where the problem becomes tractable. Finally, we show how our basic framework can easily be extended to permit social laws in which constraints on the legality or otherwise of some action may be explicitly required: for example, we show how the feasibility and synthesis problems for *dictatorship* social laws can be formulated.

We begin by introducing Action-based Alternating Transition Systems, (AATSs), the semantic structures that we employ throughout the remainder of the paper; we then introduce ATL itself, and give some properties of the logic that will be used throughout the remainder of the paper. In section 4, we introduce and formally define our social laws framework with respect to AATSs and ATL, present the feasibility and synthesis problems, and show how these can be understood as ATL model checking problems. A discussion of related work is presented in section 5, and some conclusions are presented in 6.

2 Action-based Alternating Transition Systems

The semantic structures underpinning ATL are known as *Alternating Transition Systems* (ATSs), and several essentially equivalent variations of these structures have been described in the ATL literature. As the notions of *action* and *action pre-condition* play such a key role in our framework, we find it convenient to work with another version of ATSs, in which actions and pre-conditions are first-class citizens. We refer to these structures as *Action-based Alternating Transition Systems* (AATSs), and emphasise that they are equivalent to "conventional" ATSs. We first assume that the systems of interest to us may be in any of a finite set Q of possible *states*, with some $q_0 \in Q$ designated as the *initial state*. Systems are populated by a set Ag of *agents*; a *coalition* of agents is simply a set $G \subseteq Ag$, and the set of all agents is known as the *grand coalition*. Each agent $i \in Ag$ is associated with a set Ac_i of possible actions, and we assume that these sets of actions are pairwise disjoint (i.e., actions are unique to agents). We denote the set of actions associated with a coalition $G \subseteq Ag$ by Ac_G, so $Ac_G = \bigcup_{i \in G} Ac_i$. A *joint action* for a coalition G is a tuple $\langle \alpha_1, \ldots, \alpha_k \rangle$, where $\alpha_i \in Ac_i$, for each $i \in G$. We denote the set of all joint actions for coalition G by J_G, so $J_G = \prod_{i \in G} Ac_i$. Given an element j

of J_G and agent $i \in G$, we denote i's component of j by j_i.

An *Action-based Alternating Transition System* – hereafter referred to simply as an AATS – is an $(n+7)$-tuple

$$S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi \rangle$$

where:

- Q is a finite, non-empty set of *states*;
- $q_0 \in Q$ is the *initial state*;
- $Ag = \{1, \ldots, n\}$ is a finite, non-empty set of *agents*;
- Ac_i is a finite, non-empty set of *actions*, for each $i \in Ag$, where $Ac_i \cap Ac_j = \emptyset$ for all $i \neq j \in Ag$;
- $\rho : Ac_{Ag} \to 2^Q$ is an *action precondition function*, which for each action $\alpha \in Ac_{Ag}$ defines the set of states $\rho(\alpha)$ from which α may be executed;
- $\tau : Q \times J_{Ag} \to Q$ is a partial *system transition function*, which defines the state $\tau(q, j)$ that would result by the performance of j from state q – note that, as this function is partial, not all joint actions are possible in all states (cf. the pre-condition function above);
- Φ is a finite, non-empty set of *atomic propositions*; and
- $\pi : Q \to 2^\Phi$ is an interpretation function, which gives the set of primitive propositions satisfied in each state: if $p \in \pi(q)$, then this means that the propositional variable p is satisfied (equivalently, true) in state q.

We require that AATSs satisfy the following two coherence constraints:

1. *Non-triviality [12].* Agents always have at least one legal action:

$$\forall q \in Q, \forall i \in Ag, \exists \alpha \in Ac_i \text{ s.t. } q \in \rho(\alpha)$$

2. *Consistency*. The ρ and τ functions agree on actions that may be performed:

$$\forall q, \forall j \in J_{Ag}, (q, j) \in \text{dom}\, \tau \text{ iff } \forall i \in Ag, q \in \rho(j_i)$$

We denote the set of sequences over Q by Q^*, and the set of non-empty sequences over Q by Q^+.

Example 1 *As a running example, consider the following scenario. There are two trains, one of which (E) is Eastbound, the other of which (W) is Westbound, each occupy their own circular track. At one point, both tracks pass through a narrow tunnel – a crash will occur if both trains are in the tunnel at the same time. Unlike the original versions of this scenario, we do* not *assume that there is a "controller" agent, whose purpose is to ensure that collisions do not occur. Instead, we will be concerned with social laws that achieve this end.*

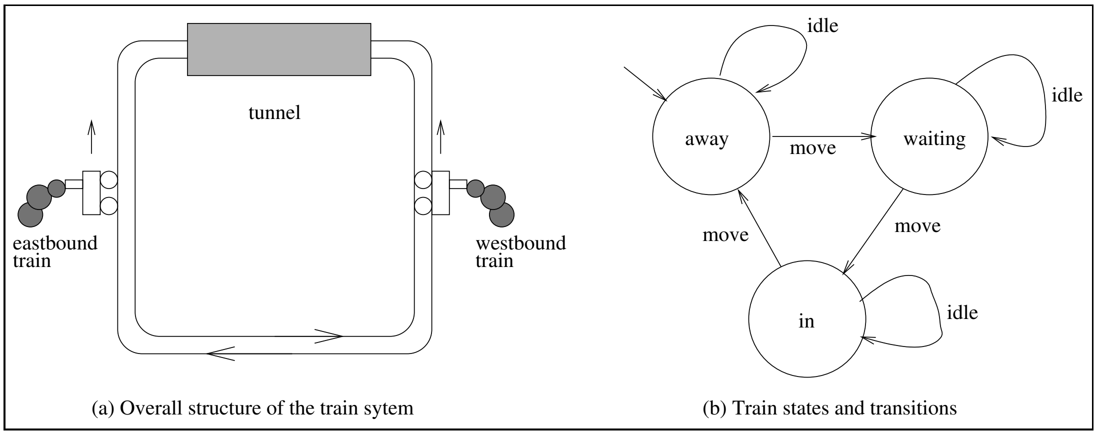

Figure 1: The train system.

We model each train $i \in Ag = \{E, W\}$ as an automaton that can be in one of three states (see Figure 1(b)): "$away_i$" (the initial state of the train); "$waiting_i$" (waiting to enter the tunnel); and "in_i" (the train is in the tunnel). Each train $i \in \{E, W\}$ has two actions available: $Ac_i = \{move_i, idle_i\}$. The $idle_i$ action is the identity, which causes no change in the train's state (i.e., it stays where it is). If a train i executes a $move_i$ action while it is $away_i$, then it goes to a $waiting_i$ state; executing a $move_i$ while $waiting_i$ causes a transition to an in_i state; and finally, executing a $move_i$ while in_i causes a transition to $away_i$ as long as the other train was not in the tunnel, while if both trains are in the tunnel, then they have crashed, and are forced to idle indefinitely. Initially, both trains are away.

The overall state of the system at any given time can be characterised in terms of the propositional variables $\{away_E, away_W, waiting_E, waiting_W, in_E, in_W\}$, where these variables have the obvious interpretation. The overall structure of the train system, and the model of trains is illustrated in Figure 1; a formal definition of the train system AATS *is given in Figure 2.*

Of course, not all combinations of propositional variables correspond to reachable system states (i.e., states that the system could possibly enter). For example, an agent i cannot be both $waiting_i$ and in_i simultaneously. There are in fact just nine reachable states of the system; see Figure 2.

Given an agent $i \in Ag$ and a state $q \in Q$, we denote the *options* available to i in q – the actions that i may perform in q – by $options(i, q)$:

$$options(i, q) = \{\alpha \mid \alpha \in Ac_i \text{ and } q \in \rho(\alpha)\}.$$

We then say that a *strategy* for an agent $i \in Ag$ is a function:

$$\sigma_i : Q \to Ac_i$$

which must satisfy the *legality* constraint that $\sigma_i(q) \in options(i, q)$ for all $q \in Q$.

A *strategy profile* for a coalition $G = \{a_1, \ldots, a_k\} \subseteq Ag$ is a tuple of strategies $\langle \sigma_1, \ldots, \sigma_k \rangle$, one for each agent $a_i \in G$. We denote by Σ_G the set of all strategy profiles for coalition $G \subseteq Ag$; if $\sigma_G \in \Sigma_G$ and $i \in G$, then we denote i's component of σ_G by σ_G^i. Given a strategy profile $\sigma_G \in \Sigma_G$ and state $q \in Q$, let $out(\sigma_G, q)$ denote the set of possible states that may result by the members of the coalition G acting as defined by their components of σ_G for one step from q:

States and Initial States:

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$ Initial state q_0

Agents, Actions, and Joint Actions:

$Ag = \{E, W\}$ $Ac_E = \{idle_E, move_E\}$ $Ac_W = \{idle_W, move_W\}$

$J_{Ag} = \{\underbrace{\langle idle_E, idle_W\rangle}_{j_0}, \underbrace{\langle idle_E, move_W\rangle}_{j_1}, \underbrace{\langle move_E, idle_W\rangle}_{j_2}, \underbrace{\langle move_E, move_W\rangle}_{j_3}\}$

Transitions/Pre-conditions:

$$\tau(q_0, j) = \begin{cases} q_0 & \text{if } j = j_0 \\ q_1 & \text{if } j = j_1 \\ q_3 & \text{if } j = j_2 \\ q_5 & \text{if } j = j_3 \end{cases} \qquad \tau(q_4, j) = \begin{cases} q_4 & \text{if } j = j_0 \\ q_7 & \text{if } j = j_1 \\ q_0 & \text{if } j = j_2 \\ q_1 & \text{if } j = j_3 \end{cases}$$

$$\tau(q_1, j) = \begin{cases} q_1 & \text{if } j = j_0 \\ q_2 & \text{if } j = j_1 \\ q_5 & \text{if } j = j_2 \\ q_6 & \text{if } j = j_3 \end{cases} \qquad \tau(q_5, j) = \begin{cases} q_5 & \text{if } j = j_0 \\ q_6 & \text{if } j = j_1 \\ q_7 & \text{if } j = j_2 \\ q_8 & \text{if } j = j_3 \end{cases}$$

$$\tau(q_2, j) = \begin{cases} q_2 & \text{if } j = j_0 \\ q_0 & \text{if } j = j_1 \\ q_6 & \text{if } j = j_2 \\ q_3 & \text{if } j = j_3 \end{cases} \qquad \tau(q_6, j) = \begin{cases} q_6 & \text{if } j = j_0 \\ q_3 & \text{if } j = j_1 \\ q_8 & \text{if } j = j_2 \\ q_4 & \text{if } j = j_3 \end{cases}$$

$$\tau(q_3, j) = \begin{cases} q_3 & \text{if } j = j_0 \\ q_5 & \text{if } j = j_1 \\ q_4 & \text{if } j = j_2 \\ q_7 & \text{if } j = j_3 \end{cases} \qquad \tau(q_7, j) = \begin{cases} q_7 & \text{if } j = j_0 \\ q_8 & \text{if } j = j_1 \\ q_1 & \text{if } j = j_2 \\ q_2 & \text{if } j = j_3 \end{cases}$$

$$\tau(q_8, j_0) = q_8$$

Propositional Variables:

$\Phi = \{away_E, away_W, waiting_E, waiting_W, in_E, in_W\}$

Interpretation Function:

$$\begin{aligned} \pi(q_0) &= \{away_E, away_W\} \\ \pi(q_1) &= \{away_E, waiting_W\} \\ \pi(q_2) &= \{away_E, in_W\} \\ \pi(q_3) &= \{waiting_E, away_W\} \end{aligned} \qquad \begin{aligned} \pi(q_4) &= \{in_E, away_W\} \\ \pi(q_5) &= \{waiting_E, waiting_W\} \\ \pi(q_6) &= \{waiting_E, in_W\} \\ \pi(q_7) &= \{in_E, waiting_W\} \\ \pi(q_8) &= \{in_E, in_W\} \end{aligned}$$

Figure 2: The AATS for the trains scenario.

$$out(\sigma_G, q) = \{q' \mid \tau(q,j) = q' \text{ where } (q,j) \in \text{dom } \tau \text{ and } \sigma^i_G(q) = j_i \text{ for } i \in G\}$$

Notice that, for any grand coalition strategy profile σ_{Ag} and state q, the set $out(\sigma_{Ag}, q)$ will be singleton.

A *computation* is an infinite sequence of states $\lambda = q_0, q_1, \ldots$. A computation $\lambda \in Q^+$ starting in state q is referred to as a *q-computation*; if $u \in \mathbb{N}$, then we denote by $\lambda[u]$ the component indexed by u in λ (thus $\lambda[0]$ denotes the first element, $\lambda[1]$ the second, and so on).

Given a strategy profile σ_G for some coalition G, and a state $q \in Q$, we define $comp(\sigma_G, q)$ to be the set of possible outcomes that may occur if every agent $a_i \in G$ follows the corresponding strategy σ_i, starting when the system is in state $q \in Q$. That is, the set $comp(\sigma_G, q)$ will contain all possible q-computations that the coalition G can "enforce" by cooperating and following the strategies in σ_G.

$$comp(\sigma_G, q) = \{\lambda \mid \lambda[0] = q \text{ and } \forall u \in \mathbb{N} : \lambda[u+1] \in out(\sigma_G, \lambda[u])\}.$$

Again, note that for any state $q \in Q$ and any grand coalition strategy σ_{Ag}, the set $comp(\sigma_{Ag}, q)$ will be a singleton, consisting of exactly one infinite computation.

3 ATL

Alternating-time Temporal Logic (ATL), can be understood as a generalisation of the well-known branching time temporal logic CTL [8], in which path quantifiers are replaced by *cooperation modalities*. A cooperation modality $\langle\langle G \rangle\rangle\varphi$, where G is a coalition, expresses the fact that the coalition G can cooperate to ensure that φ; more precisely, that there exists a strategy profile for G such that by following this strategy profile, G can ensure φ. Thus, for example, the system requirement "agents 1 and 2 can cooperate to ensure that the system never enters a fail state" may be captured by the ATL formula $\langle\langle 1, 2 \rangle\rangle \Box \neg \mathit{fail}$. The "$\Box$" temporal operator means "now and forever more": additional temporal connectives in ATL are "$\Diamond$" ("either now or at some point in the future"), "$\mathcal{U}$" ("until"), and "$\bigcirc$" ("in the next state").

Formally, the set of ATL formulae, formed with respect to a set of agents Ag, and a set of primitive propositions Φ, is given by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle G \rangle\rangle \bigcirc \varphi \mid \langle\langle G \rangle\rangle \Box \varphi \mid \langle\langle G \rangle\rangle \varphi \,\mathcal{U}\, \varphi$$

where $p \in \Phi$ is a propositional variable and $G \subseteq Ag$ is a set of agents. We identify two fragments of ATL: the $\bigcirc$-fragment is the class of ATL-formulae containing only the $\bigcirc$ temporal operator; *propositional logic* formulae are those containing no cooperation modalities.

We can now give the rules defining the satisfaction relation "$\models$" for ATL, which holds between pairs of the form S, q (where S is an AATS and q is a state in S), and formulae of ATL:

$S, q \models p$ iff $p \in \pi(q)$ (where $p \in \Phi$);

$S, q \models \neg\varphi$ iff $S, q \not\models \varphi$;

$S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$;

$S, q \models \langle\langle G \rangle\rangle \bigcirc \varphi$ iff $\exists \sigma_G \in \Sigma_G$, such that $\forall \lambda \in comp(\sigma_G, q)$, we have $S, \lambda[1] \models \varphi$;

$S, q \models \langle\langle G\rangle\rangle \Box\varphi$ iff $\exists\sigma_G \in \Sigma_G$, such that $\forall\lambda \in comp(\sigma_G, q)$, we have $S, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$;

$S, q \models \langle\langle G\rangle\rangle\varphi\,\mathcal{U}\,\psi$ iff $\exists\sigma_G \in \Sigma_G$, such that $\forall\lambda \in comp(\sigma_G, q)$, there exists some $u \in \mathbb{N}$ such that $S, \lambda[u] \models \psi$, and for all $0 \leq v < u$, we have $S, \lambda[v] \models \varphi$.

The remaining classical logic connectives ("$\wedge$", "$\rightarrow$", "$\leftrightarrow$") are assumed to be defined as abbreviations in terms of $\neg, \vee$, in the conventional manner. For readability, we omit set brackets in cooperation modalities, for example writing $\langle\langle 1\rangle\rangle$ instead of $\langle\langle\{1\}\rangle\rangle$.

Two cooperation modalities play a special role in the remainder of the paper, and are worth singling out special attention. The cooperation modality $\langle\langle\rangle\rangle$ ("the emptyset of agents can cooperate to...") asserts that its argument is true on all computations, and thus acts like CTL's universal path quantifier "A". Similarly, the cooperation modality $\langle\langle Ag\rangle\rangle$ asserts that its argument is satisfied on at least one computation, and thus acts like the CTL path quantifier "E".

The *model checking problem* for ATL is the problem of determining, for any given ATL formula φ, AATS S, and state q in S, whether or not $S, q \models \varphi$. If S is an AATS and φ is a formula then we say that φ is *initially satisfied* in S if $S, q_0 \models \varphi$; we indicate this by writing $S \models \varphi$. A formula φ is *satisfiable* if there is some AATS S and state q in S such that $S, q \models \varphi$. The *satisfiability problem* for ATL is the problem of determining, for any given ATL formula, whether this formula is satisfiable or not.

3.1 Some Properties of ATL

We begin by proving some properties of ATL that we use in subsequent proofs. For any $S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi\rangle$, let the relation R_{Ag} between states in Q be defined as: $q_1 R_{Ag} q_2$ iff for some $j \in Ac_{Ag}, \tau(q_1, j) = q_2$. In other words, $q_1 R_{Ag} q_2$ is true if the grand coalition can enforce a transition from q_1 to q_2. Now, let $S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi\rangle$ and $S' = \langle Q', q'_0, Ag, Ac'_1, \ldots, Ac'_n, \rho', \tau', \Phi, \pi'\rangle$ be two AATSs. We say that S' is a subsystem of S, (notation $S' \sqsubseteq S$), or that S is a supersystem of S', (notation $S \sqsupseteq S'$), if there exists a relation $\Re \subseteq Q \times Q'$ such that:

1. $q_0 \Re q'_0$
2. $\forall q \in Q,\ q' \in Q' : q \Re q' \Rightarrow (\pi(q) = \pi(q'))$
3. $\forall q_1 \in Q,\ q'_1, q'_2 \in Q' : ((q_1 \Re q'_1 \ \&\ q'_1 R'_{Ag} q'_2) \Rightarrow (\exists q_2 \in Q :\ q_1 R_{Ag} q_2 \ \&\ q_2 \Re q'_2))$

A special case is obtained when $S' = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho', \tau', \Phi, \pi\rangle$, in which case we write $S' \ll S$. The relation $\Re$ is essentially half of a *bisimulation* between two Kripke models [3, p.64]: the final clause represents the "backward clause" for such relations, with the "forward clause" not being present in the conditions on $\Re$. We are interested in which formulas are preserved when going from S to S', where $S' \sqsubseteq S$. To this end, we define a *universal* and an *existential* sublanguage of ATL, denoted $\mathcal{L}^u$ and $\mathcal{L}^e$, respectively. These languages are defined by the following grammars:

$$\mathcal{L}^u \quad \upsilon ::= p \mid \neg p \mid \upsilon \wedge \upsilon \mid \upsilon \vee \upsilon \mid \langle\langle\rangle\rangle\bigcirc\upsilon \mid \langle\langle\rangle\rangle\Diamond\upsilon \mid \langle\langle\rangle\rangle\Box\upsilon \mid \langle\langle\rangle\rangle\upsilon\,\mathcal{U}\,\upsilon$$

$$\mathcal{L}^e \quad \epsilon ::= p \mid \neg p \mid \epsilon \wedge \epsilon \mid \epsilon \vee \epsilon \mid \langle\langle Ag\rangle\rangle\bigcirc\epsilon \mid \langle\langle Ag\rangle\rangle\Diamond\epsilon \mid \langle\langle Ag\rangle\rangle\Box\epsilon \mid \langle\langle Ag\rangle\rangle\epsilon\,\mathcal{U}\,\epsilon$$

where $p \in \Phi$.

Lemma 1 Suppose we have $S \sqsubseteq S'$ and $\upsilon \in \mathcal{L}^u$, $\epsilon \in \mathcal{L}^e$. Then:

1. $\forall q \in Q,\ q' \in Q'$ with $q\Re q'$: $S, q \models \upsilon \Rightarrow S', q' \models \upsilon$.

2. $\forall q \in Q,\ q' \in Q'$ with $q\Re q'$: $S', q' \models \epsilon \Rightarrow S, q \models \epsilon$.

Proof: We only prove the first item; the second follows from it and the observation that every $\epsilon \in \mathcal{L}^e$ is equivalent to the negation of some $\upsilon \in \mathcal{L}^u$. The proof is by structural induction. For the case υ is p or $\neg p$ where $p \in \Phi$, the claim follows immediately from the second condition in the definition of subsystems. The cases of conjunction and disjunction follow directly, so assume that υ equals $\langle\langle\rangle\rangle\Diamond\upsilon'$, and that the claim is proven for υ'. Assume that $S, q \models \langle\langle\rangle\rangle\Diamond\upsilon'$, and $q\Re q'$. The former implies that for all strategy profiles $\sigma \in \Sigma_{Ag}$, for the unique $\lambda_\sigma \in comp(\sigma, q)$, we have that for some i_σ, $S, \lambda[i_\sigma] \models \upsilon'$. In order to derive a contradiction, suppose that $S', q' \not\models \langle\langle\rangle\rangle\Diamond\upsilon'$. It would mean $S', q' \models \langle\langle Ag\rangle\rangle\Box\neg\upsilon'$, i.e., for some strategy profile $\sigma' \in \Sigma_{Ag}$, and the computation $\lambda' \in comp(\sigma', q')$ and all $i \in \mathbb{N}$, we have $S', \lambda'[i] \models \neg\upsilon'$. Obviously, we have $q' = \lambda'[0]R_{Ag}\lambda'[1]\ldots\lambda'[i]\ldots$. By the backward condition on $\Re$, we hence also find $q_0 = q, q_1, \ldots q_i, \ldots$ with $q_i R_{Ag} q_{i+1}$ and $q_i\Re\lambda'[i]$. By the inductive assumption, we have $S, q_i \models \neg\upsilon'$ for all i. But then, the agents Ag have a strategy to always ensure $\neg\upsilon'$ from q: they just choose the actions that induce this path. This then means that $S, q \models \langle\langle Ag\rangle\rangle\Box\neg\upsilon'$, which contradicts the fact that $S, q \models \langle\langle\rangle\rangle\Diamond\upsilon'$. The remaining cases are similar. $\square$

It is not possible within the scope of this paper to examine other properties of ATL in detail; instead, we simply note some results that are pertinent to the goals of this paper.

Theorem 1 ([1, 14, 6]) *The satisfiability problem for* ATL *is* EXPTIME*-complete [6], while the satisfiaibility problem for the* $\bigcirc$*-fragment of* ATL *is* PSPACE*-complete [14]. The model checking problem for "full"* ATL *can be solved in time* $O(|Q| \cdot |\varphi|)$*, where* $|Q|$ *is the number of states, and* $|\varphi|$ *is the size of the formula to be checked [1].*

4 Social Laws

We now introduce the formal framework of social laws, which we build upon throughout the remainder of the paper. Intuitively, a social law consists of two parts:

- An *objective*, which represents what the society aims to achieve by the introduction, or adoption of the law. In human societies, for example, the objective of a law might be to eliminate alcohol-related road accidents.

- A *behavioural constraint*, which corresponds to the requirements that a law places on the members of a society. A behavioural constraint corresponding to the objective of eliminating alcohol-related road accidents might be to forbid the action of driving a car after drinking alcohol.

We represent an objective for a social law as a formula of ATL, the intuition being that a social law is *effective* if it ensures that the objective is satisfied. (We give the formal definition shortly.) ATL provides a natural and powerful language for expressing the objectives of social laws, not least because such laws frequently refer to the *powers* or *rights* (or, conversely, the limits to powers) that agents have. The original social laws framework of Shoham and Tennenholtz illustrates this [15]. In this framework, each agent $i \in Ag$ was associated with a set $F_i \subseteq Q$ of *focal states*, the idea being that a successful social law would be one which ensured that if ever an agent $i \in Ag$ found the environment was in a state $q_1 \in F_i$, then i should have the power to ensure that, the environment eventually entered state $q_2 \in F_i$. This objective can naturally be expressed within our framework: assume that, for each $q \in Q$, we have a proposition $\mathbf{q}$ that is satisfied iff the system is currently in state q. Then, given focal state sets $F_1, \ldots, F_n$, we can express the Shoham-Tennenholtz objective as follows:

$$\bigwedge_{i \in Ag} \left(\bigwedge_{\mathbf{q} \in F_i} \left[\mathbf{q} \rightarrow \bigwedge_{\mathbf{q}' \in F_i} \langle\langle i \rangle\rangle \Diamond \mathbf{q}' \right] \right) \tag{1}$$

The ATL framework allows us to express much richer objectives, however.

Example 2 *Recall the trains scenario introduced earlier. The most obvious requirement for a social law is that the trains do not crash. The objective for this social law, O_1, is thus:*

$$O_1 = \neg(in_E \wedge in_W)$$

The basic system S_1 does not ensure that (O_1) is satisfied – there are initial computations of the basic system on which both trains enter the tunnel simultaneously: $S_1, q_0 \not\models \langle\langle\rangle\rangle \Box \neg(in_E \wedge in_W)$.

We model a behavioural constraint, β, as a function

$$\beta : Ac_{Ag} \rightarrow 2^Q$$

with the intended interpretation that if $q \in \beta(\alpha)$, then action α may *not* be performed when the system is in state q – that is, α is "forbidden" in state q. (As an aside, notice the duality between the pre-condition function ρ, and behavioural constraints β: if $q \in \rho(\alpha)$, then α is permitted in q, whereas if $q \in \beta(\alpha)$, then α is forbidden in q.) We will require that any behavioural constraint is "reasonable", in that it always permits an agent to have at least one action left that can be performed in any state:

$$\forall\, i \in Ag, \forall q \in Q, \exists \alpha \in options(i, q) \text{ s.t. } q \notin \beta(\alpha).$$

We can now see what it means for a behavioural constraint β to be *implemented* in an AATS S: it simply means eliminating from S all transitions that are forbidden by β. The operation of implementing a behavioural constraint is thus an *update* on AATSs, in the sense that it results in a new AATS, which potentially satisfies different formulae. We denote the AATS obtained from S by implementing β by $S \dagger \beta$. Formally, if $S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi \rangle$ is an AATS, and β is a behavioural constraint on S, then

$$S \dagger \beta = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho', \tau', \Phi, \pi \rangle,$$

where:

1. $\forall \alpha \in Ac$,

   $\rho'(\alpha) = \rho(\alpha) \setminus \beta(\alpha)$

2. $\forall q \in Q, \forall j \in J_{Ag}$,
$$\tau'(q,j) = \begin{cases} \tau(q,j) & \text{if } (q,j) \in \text{dom } \tau \text{ and } \forall i \in Ag, q \notin \beta(j_i) \\ \text{undefined} & \text{otherwise} \end{cases}$$

3. All other components of $S \dagger \beta$ are as in S.

It is natural to ask what properties the implementation operator "†" has. First, notice that AATSs are closed under the implementation of behavioural constraints. That is, if S is an AATS and β is a behavioural constraint over S, then $S \dagger \beta$ is an AATS – it satisfies the non-triviality and consistency coherence constraints given earlier. Second, notice that $(S \dagger \beta) \dagger \beta = S \dagger \beta$. Third, consider what properties of AATSs might be preserved through by the implementation of social laws. To answer this question, first recall the notion of a *subsystem* as defined in section 3.1; we have the following result.

Lemma 2 Let S be an AATS, and let β be a behavioural constraint over S. Then $S \dagger \beta$ is a subsystem of S, i.e., $S \dagger \beta \sqsubseteq S$. (In fact, $S \dagger \beta \ll S$.)

From Lemma 1, we immediately obtain the following.

Lemma 3 Suppose we have an AATS S, a behavioural constraint β over S, a state q in S, and formulae $\upsilon \in \mathcal{L}^u$, $\epsilon \in \mathcal{L}^e$. Then:

1. If $S, q \models \upsilon$ then $S \dagger \beta, q \models \upsilon$.
2. If $S \dagger \beta, q \models \epsilon$ then $S, q \models \epsilon$.

Bringing these components together, a *social law* over an AATS S is a pair:

$$(\varphi, \beta)$$

where:

- φ is an ATL formula called the *objective* of the law; and
- $\beta : Ac_{Ag} \to 2^Q$ is a behavioural constraint on S.

A social law (φ, β) is *effective* in AATS S if, after implementing β in S, we know that $\langle\langle\rangle\rangle \Box \varphi$ will be initially satisfied in S, i.e., if $S \dagger \beta \models \langle\langle\rangle\rangle \Box \varphi$.

There are three key computational questions with respect to social laws, which we shall investigate throughout the remainder of this paper:

1. *Effectiveness*. Given an AATS S and a social law (φ, β) over S, determine whether (φ, β) is effective in S.

2. *Feasibility*. Given an AATS S and a formula φ of ATL representing an objective, does there exist a behavioural constraint β such that (φ, β) is an effective social law in S.

3. *Synthesis*. Given an AATS S and a formula φ of ATL representing an objective, exhibit a behavioural constraint β such that (φ, β) is an effective social law in S if such a constraint exists, otherwise answer "no".

Our first result, with respect to the effectiveness problem, is now immediate.

Lemma 4 The effectiveness problem for social laws may be solved in time polynomial in the size of S and φ.

Proof: Generate $S' = S \dagger \beta$, and check that $S', q_0 \models \langle\langle\rangle\rangle \Box\varphi$. The first step can obviously be done in polynomial time, and from Theorem 1, so can the second step. $\square$

Example 3 *With respect to the trains example, is there a behavioural constraint β_1, such that (O_1, β_1) is an effective social law? Clearly there is. The constraint β_1 must ensure that the system never enters state q_8: from examination of the state transition function τ (see Figure 2), we can see that $\tau(q_5, j_3) = \tau(q_6, j_2) = \tau(q_7, j_1) = q_8$, and there are no other transitions leading to q_8 (apart from when the trains have already crashed, which we need not consider!) Consider the behavioural constraint β_1 as follows.*

$$\beta_1(\alpha) = \begin{cases} \emptyset & \text{if } \alpha = idle_E \\ \emptyset & \text{if } \alpha = idle_W \\ \{q_5, q_6\} & \text{if } \alpha = move_E \\ \{q_7\} & \text{if } \alpha = move_W \end{cases}$$

The constraint ensures that:

- *when both agents are waiting to enter the tunnel, the eastbound train is prevented from moving;*
- *when the westbound train is already in the tunnel and the eastbound train is waiting to enter the tunnel, then the eastbound train is prevented from moving; and*
- *when the eastbound train is already in the tunnel and the westbound train is waiting to enter the tunnel, then the westbound train is prevented from moving.*

Notice that this constraint is, in a sense, asymmetric, as it constrains the eastbound train rather than the westbound train: we could equally well replace the first constraint with the requirement that if both trains are waiting to enter the tunnel, then the westbound *train is prevented from moving, thus enabling the eastbound train to enter. Now, let $S_2 = S_1 \dagger \beta_1$. We claim that $S_2, q_0 \models \langle\langle\rangle\rangle \Box\neg(in_E \wedge in_W)$; this can be established by inspection. Thus:*

Lemma 5 *(O_1, β_1) is an effective social law in S_1.*

Of course, there are other, less "sensible" behavioural constraints that are effective in S_1 for O_1. Consider β_2:

$$\beta_2(\alpha) = \begin{cases} Q & \text{if } \alpha = move_E \text{ or } \alpha = move_W \\ \emptyset & \text{otherwise} \end{cases}$$

This behavioural constraint prevents both trains from moving. And yet:

Lemma 6 *(O_1, β_2) is an effective social law in S_1.*

Clearly, our original objective needs some refinement. Consider objective O_2:

$$O_2 = O_1 \wedge \bigwedge_{i\in\{E,W\}} \big(waiting_i \rightarrow \langle\langle i\rangle\rangle\Diamond(in_i \wedge O_1)\big)$$

This objective ensures that, not only do the trains never crash, but that both trains can always safely enter the tunnel if they are waiting. Neither β_1 nor β_2 is effective for O_2. Consider β_3, which works by forbidding trains from lingering in the tunnel, but is otherwise the same as β_1:

$$\beta_3(\alpha) = \begin{cases} \{q_4, q_7\} & \text{if } \alpha = idle_E \\ \{q_2, q_6\} & \text{if } \alpha = idle_W \\ \{q_5, q_6\} & \text{if } \alpha = move_E \\ \{q_7\} & \text{if } \alpha = move_W \end{cases}$$

Lemma 7 *(O_2, β_3) is an effective social law in S_1.*

However, objective O_2 may still need further refinement, as less sensible behavioural constraints can still be effective in S_1:

Lemma 8 *(O_2, β_2) is an effective social law in S_1.*

So, consider objective O_3:

$$O_3 = O_1 \ \wedge \bigwedge_{i\in\{E,W\}} \begin{pmatrix} (away_i \rightarrow \langle\langle i\rangle\rangle\Diamond waiting_i) & \wedge \\ (waiting_i \rightarrow \langle\langle i\rangle\rangle\Diamond(in_i \wedge O_1)) & \wedge \\ (in_i \rightarrow \langle\langle i\rangle\rangle\bigcirc away_i) & \end{pmatrix}$$

This objective ensures that, not only do the trains never crash, but both trains will at some point enter the $waiting_i$ state. The objective also ensures that, once in the $waiting_i$ state, the train does eventually enter the state where it has safely entered the tunnel. Once the train is in the tunnel, it will not linger there, but it's next state will be where it has safely left the tunnel and gone into the $away_i$ state.

Lemma 9 *(O_3, β_2) is not an effective social law in S_1, but (O_3, β_3) is.*

The first issue to which we address ourselves is the computational complexity of the feasibility problem. As it turns out, the feasibility problem in our framework in no harder, for objectives expressed as arbitrary atl formulae, then the corresponding problem studied by Shoham and Tennenholtz, where objectives were expressed as sets of focal states (see above); and for objectives propositional logic objectives, it is easier.

Theorem 2 *The feasibility problem for objectives expressed as arbitrary* ATL *formulae is* NP*-complete, while for objectives expressed as propositoional formulae, the feasibility problem is decidable in polynomial time.*

Proof: With respect to the first result, membership in NP may be seen by the following non-deterministic algorithm: (1) guess a behavioural constraint β; (2) verify that β is effective. Since dom $\beta = Ac_{Ag}$, step (1) can be done in (non-deterministic) polynomial time $O(|Ac_{Ag} \times Q|)$, while from Lemma 4, step (2) requires only polynomial time. To see that the problem is NP-hard, observe that the Useful Social Law problem of Shoham-Tennenholtz, (proven to be NP-complete in [16, p.612]), can be directly reduced to our feasibility problem, by encoding focal state sets as shown in equation (1). As the reduction is clearly possible in polynomial time, we are done. The second result is an immediate corollary of the following result, which shows that, for objectives expressed as formulae of propositional logic, the feasibility problem reduces directly to model checking in ATL. $\square$

Theorem 3 *Suppose φ is a propositional logic formula (representing an objective), and S is an* AATS*. Then $S \models \langle\langle Ag \rangle\rangle \Box \varphi$ iff φ is feasible in S.*

Proof:

($\rightarrow$) Assume $S \models \langle\langle Ag \rangle\rangle \Box \varphi$. By the semantics of ATL, we know that $\exists \sigma_{Ag} \in \Sigma_{Ag}$ such that $\forall \lambda \in comp(\sigma_{Ag}, q_0)$, we have $S, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$. In fact, since σ_{Ag} is a grand coalition strategy, $comp(\sigma_{Ag}, q_0)$ will be a singleton; let λ^* denote its member; so $\forall u \in \mathbb{N}$: $S, \lambda^*[u] \models \varphi$. We must show that this implies there exists a behavioural constraint β such that $S \dagger \beta \models \langle\langle\rangle\rangle \Box \varphi$; we construct β as follows:

for each $i \in Ag$,

for each $\alpha \in Ac_i$,

set $\beta(\alpha) = \{q \mid \sigma_i(q) \neq \alpha\}$.

We claim that, for any q_0-computation λ of $S \dagger \beta$, we have $S \dagger \beta, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$. To see this, observe that there will in fact be a single q_0 computation of $S \dagger \beta$, namely λ^*, and we know that $\forall u \in \mathbb{N}$, $S, \lambda^*[u] \models \varphi$. We appeal to the fact that φ is a propositional formula, and that π is the same in S and $S \dagger \beta$, and conclude that for all $u \in \mathbb{N}$: $S \dagger \beta, \lambda^*[u] \models \varphi$.

($\leftarrow$) Assume φ is feasible in S. Then there exists a behavioural constraint β such that $S \dagger \beta, q_0 \models \langle\langle\rangle\rangle \Box \varphi$. Let Λ denote the set of q_0-computations of $S \dagger \beta$. Then by the semantics of ATL, for all $\lambda \in \Lambda$ and for all $u \in \mathbb{N}$, we have $S \dagger \beta, \lambda[u] \models \varphi$. We need to show that this implies $S \models \langle\langle Ag \rangle\rangle \Box \varphi$. By the semantics of ATL, $S, q_0 \models \langle\langle Ag \rangle\rangle \Box \varphi$ iff $\exists \sigma_{Ag} \in \Sigma_{Ag}$ such that $\forall \lambda \in comp(q_0, \sigma_{Ag})$, and $\forall u \in \mathbb{N}$, we have $S, \lambda[u] \models \varphi$. We show how to construct such a σ_{Ag}. We start by constructing from β a *non-deterministic* strategy $\sigma_i^n : Q \rightarrow 2^{Ac_i}$ for each agent $i \in Ag$, as follows:

for each $i \in Ag$,

for each $q \in Q$,

set $\sigma_i^n(q) = \{\alpha \mid q \notin \beta(\alpha)\}$.

Now, say a (conventional deterministic) strategy σ_i is *consistent* with σ_i^n if $\sigma_i(q) \in \sigma_i^n(q)$, for all $q \in Q$. Now, let $\sigma_{Ag} = \langle \sigma_1, \ldots, \sigma_n \rangle$ be any grand coalition strategy profile such that each σ_i is consistent with the corresponding non-deterministic strategy σ_i^n. We claim that, thus defined, $\forall \lambda \in comp(\sigma_{Ag}, q_0)$ we have $S, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$. To see this, observe that by construction we have $comp(\sigma_{Ag}, q_0) \subseteq \Lambda$, as defined above. We know that $\forall \lambda \in \Lambda$, and for all $u \in \mathbb{N}$, we have $S \dagger \beta, \lambda[u] \models \varphi$. Since φ is propositional, its valuation depends only upon the state in which it is interpreted. Since π is unchanged in S and $S \dagger \beta$, it must be that $\forall \lambda \in \Lambda$, and for all $u \in \mathbb{N}$: $S, \lambda[u] \models \varphi$.

$\square$

Notice that, as a direct corollary to this result, the synthesis problem for propositional logic objectives may also be solved in polynomial time: for if the answer to the model checking problem is "yes", then the witness to this will be strategy for the grand coalition from which, as the proof of the theorem illustrates, we can extract a behavioural constraint implementing the objective.

This previous Theorem illustrates a close relationship between the feasibility problem and model checking, which begs the question as to what extent the result can be extended for objectives beyond propositional logic. We have the following.

Theorem 4 *Suppose $\upsilon \in \mathcal{L}^u$ is a universal* ATL *formula (representing an objective), and S is an* AATS*. Then $S \models \langle\langle Ag \rangle\rangle \Box \upsilon$ implies υ is feasible in S.*

Proof: (Sketch.) The basic structure of the proof is as Theorem 3, but as υ is no longer a propositional formula, but a universal ATL formula, we make use of Lemma 3. ▫

4.1 Objectives with Explicit Action Constraints

We now consider objectives for laws *that explicitly refer to the legality or otherwise of actions actions*. The following example illustrates the idea.

Example 4 *If α is an action, then let the proposition $\ell(\alpha)$ mean "action α is legal". Consider the following three objectives in the context of the trains system, as discussed earlier.*

$$\begin{array}{lcl} O_5 & = & \neg(in_E \wedge in_W) \wedge \ell(move_E) \wedge \ell(move_W) \\ O_6 & = & \neg(in_E \wedge in_W) \wedge \ell(move_E) \\ O_7 & = & \neg(in_E \wedge in_W) \wedge \ell(move_W) \end{array}$$

Objective O_5 requires that not only do the trains never crash, but that both trains are always able to move. *Any behavioural constraint for this objective must not prevent the trains from moving if they choose to do so. Objective O_6 is similar, but only requires that the Eastbound train is always able to move; and O_7 only requires that the Westbound train is always able to move.*

In our basic social laws framework, we have no way of expressing or reasoning about social laws with such objectives, We now show how the basic framework can be extended to include such constraints. In particular, we show how this can be done in the context of the MOCHA model checking system [2]. MOCHA takes as input a specification of an AATS, expressed in the REACTIVE MODULES language, and a formula of ATL, and is capable of either checking whether this formula is true in the AATS, or else giving a counter example. The actual syntax of the REACTIVE MODULES language used in MOCHA is rather complex, and so we adopt the following simplified syntax in the interests of easy comprehension. A REACTIVE MODULES agent (they are called "atoms" in the REACTIVE MODULES literature) has the following structure:

$$\begin{array}{l} \texttt{agent}\ \mathit{name}\ \texttt{reads}\ \mathit{in}\ \texttt{writes}\ \mathit{out} \\ \quad P_1 \mapsto \alpha_1; \\ \quad P_2 \mapsto \alpha_2; \\ \quad \cdots \\ \quad P_k \mapsto \alpha_k. \end{array} \tag{2}$$

where *name* is the name of the agent, $in \subseteq \Phi$ is a list of boolean variables that the agent observes, $out \subseteq \Phi$ is a list of boolean variables that it controls, and provides to the rest of the system, and the $P_j \mapsto \alpha_j$ structures are *guarded commands*, where P_j is a predicate over the variables the agent observes and controls (i.e., a boolean expression over variables $in \cup out$), and $\alpha_j \in Ac_{name}$ is an action. Action's can be understood as functions that take as input the variables visible to the agent ($in \cup out$), and produce as output an assignment for the variables controlled by the agent (*out*). The idea is that at each time step, each agent generates the set of rules whose pre-conditions are satisfied by the current state of the system. One of these rules is non-deterministically chosen for execution – this involves simply executing the corresponding action, which produces an assignment for the variables under its control (*out*), which is the value they take in the next state of the system. Of course, a guarded command may have $\top$ as a pre-condition, in which case it is always enabled for execution.

It should be clear how a collection of such agents maps to an AATS, as defined in section 2. The most important point is that the pre-conditions to guarded commands correspond to the pre-condition function ρ: given a state of the system q and guarded command $P \mapsto \alpha$, then $q \in \rho(\alpha)$ iff $q \models P$.

We now show how an AATS, expressed in such a framework, can be extended to permit objectives referring to the legality or otherwise of actions. The idea is to define a transformation on AATSs:

$$\cdot^{\circ} : \text{AATS} \rightarrow \text{AATS}$$

such that the transformed system includes the $\ell(\alpha)$ propositions as before, but otherwise has the same properties.

Given an AATS $S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi \rangle$, defined using the REACTIVE MODULES language as above, the system S° is created as follows.

First, for each action $\alpha \in Ac$, we create:

- A new propositional variable $\ell(\alpha)$, with the intended interpretation that $\ell(\alpha)$ will be true in a state if the action α is legal according to the behavioural constraint that we are looking for.

- A new agent, which controls the variable $\ell(\alpha)$, as follows.

$$\begin{array}{l} \texttt{agent}\ \alpha\textit{-controller}\ \texttt{reads}\ \Phi\ \texttt{writes}\ \ell(\alpha) \\ \qquad \top \mapsto \ell(\alpha) := \top; \\ \qquad \top \mapsto \ell(\alpha) := \bot. \end{array}$$

  Thus, in every possible state, the agent α-*controller* has two actions available: make $\ell(\alpha)$ true, or make $\ell(\alpha)$ false. Note that because the condition on the guard of each of these actions is $\top$, the α-*controller* agent can *always* perform these actions. We will denote by *controllers* the set of all controller agents introduced in this way.

We then take each of the original agents i in the system, and transform it as follows.

- We replace each guarded command $P \mapsto \alpha$ for agent i with a rule as follows.

$$\ell(\alpha)' \wedge P \mapsto \alpha$$

Notice that the first conjunct of the guard is "primed"; this notation in REACTIVE MODULES means the value that $\ell(\alpha)$ has been assigned *in the current round*, i.e., the value of this variable after it has been assigned by the α-*controller* agent. Thus agent i may perform the action α iff the original guard condition P is true in the current state, *and the new agent controlling $\ell(\alpha)$ has assigned this variable the value $\top$ in the current state*. In this way, the α-*controller* agent can determine whether or not α is performed.

- Next, if *in* is the set of variables that i `reads`, then we replace *in* by $in \cup \{\ell(\alpha) \mid \alpha \in Ac_i\}$. In other words, the agent i `reads` those ℓ-variables that determine whether its actions are legal.

  Notice that agent i is not dependent on the ℓ-variables of any other agent's actions, and hence any strategy computed for i by MOCHA will not be dependent on the ℓ-variables of other agents.

Now, we can prove:

Theorem 5 *Suppose φ is a propositional logic formula (representing an objective), and S is an* AATS. *Then*

$$S^\circ \models \langle\langle controllers \rangle\rangle \Box \left(\varphi \wedge \left[\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha) \right] \right)$$

iff φ is feasible in S.

Proof: (Sketch) The structure of the proof is as Theorem 3: if $S^\circ, q_0 \models \langle\langle controllers \rangle\rangle \Box \varphi$, then the controller agents *controllers* have a strategy for enabling/disabling actions such that if this strategy is followed, the law is implemented. From this strategy, we can extract a behavioural contraint that implements the law. The additional condition ensures that every agent has at least one action available in every state, and so the strategy controlling the $\ell(\alpha)$ variables really does map to a behavioural constraint, as defined earlier. Conversely, if there is a behavioural constraint β such that (φ, β) is an effective social law in S°, then we can extract a strategy for the *controllers* such that this strategy ensures φ is always true. □

Notice that the obvious analogue of Theorem 4 also holds, although space restrictions prevent more discussion.

Example 5 *In the context of the train example:*

Lemma 10 *Objective O_5 is not feasible in S_1, while both objectives O_6 and O_7 are feasible.*

Using this framework, we can also investigate more general properties of social laws. Consider behavioural constraints that act as *dictatorships* (cf. [13, p.17]). A behavioural constraint is a dictatorship if, once it is implemented, it never presents an agent with more than one possible action at any given time; that is, if an agent has no choice about what action it may perform. Formally, a behavioural constraint β with respect to an AATS S is a dictatorship if

$$\forall i \in Ag, \forall q \in Q, |options(i, q)| = 1$$

where $options(i, q)$ is evaluated in $S \dagger \beta$.

Example 6 *The following is a dictatorship behavioural constraint for the trains scenario:*

$$\beta_4(\alpha) = \begin{cases} \{q_0, q_4, q_6\} & \textit{if } \alpha = \textit{idle}_E \\ \{q_0, q_5, q_6\} & \textit{if } \alpha = \textit{idle}_W \\ \{q_5\} & \textit{if } \alpha = \textit{move}_E \\ \{q_4\} & \textit{if } \alpha = \textit{move}_W \end{cases}$$

Theorem 6 *Suppose φ is a propositional logic formula (representing an objective), and S is an* AATS. *Then φ is feasible in S by a dictatorship behavioural constraint iff*

$$S^\circ \models \langle\langle \textit{controllers} \rangle\rangle \Box \left(\varphi \wedge \bigwedge_{i \in Ag} \left[\bigvee_{\alpha \in Ac_i} \ell(\alpha) \right] \wedge \left[\bigwedge_{\alpha_1 \in Ac_i} \bigwedge_{\substack{\alpha_2 \in Ac_i, \\ \alpha_1 \neq \alpha_2}} \neg(\ell(\alpha_1) \wedge \ell(\alpha_2)) \right] \right).$$

Proof: (Sketch.) As before, except that we have an additional constraint, which ensures that no more than one action from an agent's action set is enabled at any given time. □

Example 7 *Returning to the trains scenario for one final time:*

Lemma 11 (O_2, β_4) *and* (O_3, β_4) *are effective social laws in* S_1.

5 Related Work

The closest approach to ours in the literature is the original framework of Moses, Shoham, and Tennenholtz. Shoham and Tennenholtz were the first to precisely articulate the notion of social laws for multiagent systems, and set up a basic formal framework within which computational questions about social laws could be formulated [15, 16, 17]. The particular application domain was that of traffic laws for robotic agents. The basic framework was extended by Fitoussi and Tennenholtz, to consider *simple* social laws – essentially, social laws that could not be any simpler without failing [9]. In related work, Moses and Tennenholtz developed an deontic epistemic logic for representing properties of multiagent systems with normative structures in place [12]. Although semantically similar to ATL (and ATEL [11]), their logic was quite different to ATL in terms of the syntactic constructs it provided, and the emphasis was primarily on deriving axioms capturing static aspects of artificial social systems and social laws.

6 Conclusions

In this paper, we have demonstrated how the Alternating-time Temporal Logic of Alur, Henzinger, and Kupferman provides a natural and powerful framework within which to express and reason about social laws. Following a formulation of social laws within ATL, we demonstrated how the effectiveness, feasibility, and synthesis problems could be understood as model checking problems for ATL, and also demonstrated how our basic framework could naturally be extended to include social laws that require explicit constraints on actions.

There are many possible routes for future investigation. One obvious question is the extent to which other notions such as knowledge can be incorporated into the framework. For example, one could imagine a social law with the objective that certain facts about agents must remain private to those agents; epistemic extensions to ATL seem to provide a natural language for representing such objectives [11, 10].

References

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

[2] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. Mocha: Modularity in model checking. In *CAV 1998: Tenth International Conference on Computer-aided Verification, (LNCS Volume 1427)*, pages 521–525. Springer-Verlag: Berlin, Germany, 1998.

[3] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press: Cambridge, England, 2001.

[4] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press: Cambridge, MA, 2000.

[5] F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7:69–79, 1999.

[6] G. van Drimmelen. Satisfiability in alternating-time temporal logic. In *Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 208–217, Ottawa, Canada, 2003.

[7] E. H. Durfee. Distributed problem solving and planning. In G. Weiß, editor, *Multiagent Systems*, pages 121–164. The MIT Press: Cambridge, MA, 1999.

[8] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[9] D. Fitoussi and M. Tennenholtz. Choosing social laws for multi-agent systems: Minimality and simplicity. *Artificial Intelligence*, 119(1-2):61–101, 2000.

[10] W. van der Hoek and M. Wooldridge. Model checking cooperation, knowledge, and time — a case study. *Research in Economics*, 57(3), September 2003.

[11] W. van der Hoek and M. Wooldridge. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 2003.

[12] Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and AI*, 14(6):533–562, 1995.

[13] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[14] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[15] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, San Diego, CA, 1992.

[16] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. In P. E. Agre and S. J. Rosenschein, editors, *Computational Theories of Interaction and Agency*, pages 597–618. The MIT Press: Cambridge, MA, 1996.

[17] Y. Shoham and M. Tennenholtz. On the emergence of social conventions: Modelling, analysis, and simulations. *Artificial Intelligence*, 94(1-2):139–166, July 1997.

[18] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, 2002.