On the Logic of Normative Systems*

Thomas Ågotnes[1], Wiebe van der Hoek[2], Juan A. Rodríguez-Aguilar[3], Carles Sierra[3], Michael Wooldridge[2]

[1] Department of Computer Engineering, Bergen University College
Norway
tag@hib.no
[2] Department of Computer Science, University of Liverpool
UK
{wiebe,mjw}@csc.liv.ac.uk
[3] Artificial Intelligence Research Institute IIIA, Spanish Council for Scientific Research CSIC, Spain
{jar,carles}@iiia.csic.es

Abstract. We introduce *Normative Temporal Logic* (NTL), a logic for reasoning about normative systems. NTL is a generalisation of the well-known branching-time temporal logic CTL, in which the path quantifiers A ("on all paths...") and E ("on some path...") are replaced by the indexed deontic operators O_η and P_η, where for example $\mathsf{O}_\eta\varphi$ means "φ is obligatory in the context of normative system η". After defining the logic, we give a sound and complete axiomatisation, and discuss the logic's relationship to standard deontic logics. We present a symbolic representation language for models and normative systems, and identify four different model checking problems, corresponding to whether or not a model is represented symbolically or explicitly, and whether or not we are given an interpretation for the normative systems named in formulae to be checked. We show that the complexity of model checking varies from P-complete up to EXPTIME-hard for these variations.

1 Introduction

Normative systems, or social laws, have been widely promoted as an approach to coordinating multi-agent systems [Shoham and Tennenholtz, 1996]. Crudely, a normative system defines a set of constraints on the behaviour of agents, corresponding to obligations, which may or may not be observed by agents. A number of formalisms have been proposed for reasoning about normative behaviour in multi-agent systems, typically based on deontic logic [Meyer and Wieringa, 1993]. However the computational properties of such formalisms – in particular, their use in the practical design and synthesis of normative systems and the complexity of reasoning with them – has received little attention. In this paper, we rectify this omission. We present Normative Temporal Logic (NTL), a logic for reasoning about normative systems, which is closely related to the well-known

* The content of this paper has previously appeared in Proc. IJCAI 2007.

Dagstuhl Seminar Proceedings 07122
Normative Multi-agent Systems
http://drops.dagstuhl.de/opus/volltexte/2007/921

and widely-used branching time logic CTL [Emerson, 1990]. In NTL, the universal and existential path quantifiers of CTL are replaced by indexed deontic operators O_η and P_η, where $\mathsf{O}_\eta\varphi$ means that "φ is obligatory in the context of the normative system η", and $\mathsf{P}_\eta\varphi$ means "φ is permissible in the context of the normative system η". Here, φ is a temporal logic expression over the usual CTL temporal operators $\bigcirc$, $\Diamond$, $\Box$, and $\mathcal{U}$ (every temporal operator must be preceded by a deontic operator, cf. CTL syntax), and η denotes a normative system. In NTL, obligations and permissions are thus, first, *contextualised* to a normative system η and, second, have a *temporal* dimension. It has been argued that the latter can help avoid some of the paradoxes of classical deontic logic. NTL generalises CTL because by letting $\eta_\emptyset$ denote the empty normative system, the universal path quantifier A can be interpreted as $\mathsf{O}_{\eta_\emptyset}$; much of the technical machinery developed for reasoning with CTL can thus be adapted for NTL [Emerson, 1990; Clarke *et al.*, 2000]. NTL is in fact a descendent of the *Normative* ATL (NATL) logic introduced in [Wooldridge and van der Hoek, 2005]: however, NTL is *much* simpler (and we believe more intuitive) than NATL, and we are able to present many more technical results for the logic: we first give a sound and complete axiomatisation, and then discuss the logic's relationship to standard deontic logics. We introduce a symbolic representation language for normative systems, and investigate the complexity of model checking for NTL, showing that it varies from P-complete in the simplest case up to EXPTIME-hard in the worst. We present an example to illustrate the approach, and present some brief conclusions.

2 Normative Temporal Logic

Kripke Structures: Let $\Phi = \{p, q, \ldots\}$ be a finite set of atomic *propositional variables*. A *Kripke structure* (over Φ) is a quad $\mathcal{K} = \langle S, S^0, R, V\rangle$, where: S is a finite, non-empty set of *states*, with $S^0 \subseteq S$ ($S^0 \neq \emptyset$) being the *initial states*; $R \subseteq S \times S$ is a total binary relation on S, which we refer to as the *transition relation*[1]; and $V : S \rightarrow 2^\Phi$ labels each state with the set of propositional variables true in that state. A *path* over R is an infinite sequence of states $\pi = s_0, s_1, \ldots$ which must satisfy the property that $\forall u \in \mathbb{N}$: $(s_u, s_{u+1}) \in R$. If $u \in \mathbb{N}$, then we denote by $\pi[u]$ the component indexed by u in π (thus $\pi[0]$ denotes the first element, $\pi[1]$ the second, and so on). A path π such that $\pi[0] = s$ is an *s-path*.

Normative Systems: In this paper, a normative system is *a set of constraints on the behaviour of agents in a system*. More precisely, a normative system defines, for every possible system transition, whether or not that transition is considered to be legal or not. Different normative systems may differ on whether or not a transition is legal. Formally, a normative system η (w.r.t. a Kripke structure $\mathcal{K} = \langle S, S^0, R, V\rangle$) is simply a subset of R, such that $R \setminus \eta$ is a total relation. The requirement that $R \setminus \eta$ is total is a *reasonableness* constraint: it prevents normative systems which lead to states with no successor. Let $N(R) = \{\eta \mid (\eta \subseteq R)\ \&\ (R \setminus \eta \text{ is total})\}$ be the set of normative systems over R. The intended

[1] A relation $R \subseteq S \times S$ is total iff $\forall s\, \exists s' : (s, s') \in R$.

interpretation of a normative system η is that $(s, s') \in \eta$ means transition (s, s') is forbidden in the context of η; hence $R \setminus \eta$ denotes the *legal* transitions of η. Since it is assumed η is reasonable, we are guaranteed that a legal outward transition exists for every state. If π is a path over R and η is a normative system over R, then π is η-*conformant* if $\forall u \in \mathbb{N}$, $(\pi[u], \pi[u+1]) \notin \eta$. Let $\mathcal{C}_\eta(s)$ be the set of η-conformant s-paths (w.r.t. some R).

Since normative systems are just *sets* (of disallowed transitions), we can *compare* them, to determine, for example, whether one is *more liberal* (less restrictive) than another: if $\eta \subset \eta'$, then η places fewer constraints on a system than η', hence η is more liberal. Notice that, assuming an *explicit* representation of normative systems, (i.e., representing a normative system η directly as a subset of R), checking such properties can be done in polynomial time. We can also operate on them with the standard set theoretic operations of union, intersection, etc. Taking the union of two normative systems η_1 and η_2 may yield (depending on whether $R \setminus (\eta_1 \cup \eta_2)$ is total) a normative system that is *more restrictive* (less liberal) than either of its parent systems, while taking the *intersection* of two normative systems yields a normative system which is *less restrictive* (more liberal). Care must be taken when operating on normative systems in this way to ensure the resulting system is reasonable.

Syntax of NTL: The language of NTL is a generalisation of CTL: the only issue that may cause confusion is that, within this language, we refer explicitly to normative systems, which are *semantic* objects. We will therefore assume a stock of syntactic elements Σ_η which will denote normative systems. To avoid a proliferation of notation, we will use the symbol η both as a syntactic element for normative systems in the language, and the same symbol to denote the corresponding semantic object. An *interpretation* for symbols Σ_η with respect to a transition relation R is a function $I : \Sigma_\eta \to N(R)$. When R is a transition relation of Kripke structure $\mathcal{K}$ we say that I is an interpretation over $\mathcal{K}$. We will assume that the symbol $\eta_\emptyset$ always denotes the *emptyset* normative system, i.e., the normative system which forbids *no* transitions. Note that this normative system will be reasonable for *any* Kripke structure. Thus, we require that for all I: $I(\eta_\emptyset) = \emptyset$. The syntax of NTL is defined by the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{P}_\eta \bigcirc \varphi \mid \mathsf{P}_\eta(\varphi \,\mathcal{U}\, \varphi) \mid \mathsf{O}_\eta \bigcirc \varphi \mid \mathsf{O}_\eta(\varphi \,\mathcal{U}\, \varphi)$$

where $p \in \Phi$ and $\eta \in \Sigma_\eta$. Sometimes we call α occurring in an expression $\mathsf{O}_\eta\alpha$ or $\mathsf{P}_\eta\alpha$ a *temporal formula* (although such an α is not a well-formed formula).

Semantic Rules: The semantics of NTL are given with respect to the satisfaction relation "$\models$". $\mathcal{K}, s \models_I \varphi$ holds when $\mathcal{K}$ is a Kripke structure, s is a state in $\mathcal{K}$, I an interpretation over $\mathcal{K}$, and φ a formulae of the language, as follows:

$\mathcal{K}, s \models_I \top$;
$\mathcal{K}, s \models_I p$ iff $p \in V(s)$ (where $p \in \Phi$);
$\mathcal{K}, s \models_I \neg\varphi$ iff not $\mathcal{K}, s \models_I \varphi$;
$\mathcal{K}, s \models_I \varphi \vee \psi$ iff $\mathcal{K}, s \models_I \varphi$ or $\mathcal{K}, s \models_I \psi$;
$\mathcal{K}, s \models_I \mathsf{O}_\eta \bigcirc \varphi$ iff $\forall \pi \in \mathcal{C}_{I(\eta)}(s) : \mathcal{K}, \pi[1] \models_I \varphi$;

$\mathcal{K}, s \models_I \mathsf{P}_\eta \bigcirc \varphi$ iff $\exists \pi \in \mathcal{C}_{I(\eta)}(s) : \mathcal{K}, \pi[1] \models_I \varphi$;
$\mathcal{K}, s \models_I \mathsf{O}_\eta(\varphi \,\mathcal{U}\, \psi)$ iff $\forall \pi \in \mathcal{C}_{I(\eta)}(s), \exists u \in \mathbb{N}$, s.t. $\mathcal{K}, \pi[u] \models_I \psi$ and $\forall v, (0 \leq v < u) : \mathcal{K}, \pi[v] \models_I \varphi$
$\mathcal{K}, s \models_I \mathsf{P}_\eta(\varphi \,\mathcal{U}\, \psi)$ iff $\exists \pi \in \mathcal{C}_{I(\eta)}(s), \exists u \in \mathbb{N}$, s.t. $\mathcal{K}, \pi[u] \models_I \psi$ and $\forall v, (0 \leq v < u) : \mathcal{K}, \pi[v] \models_I \varphi$

The remaining classical logic connectives ("$\wedge$", "$\rightarrow$", "$\leftrightarrow$") are assumed to be defined as abbreviations in terms of $\neg, \vee$, in the conventional manner. We write $\mathcal{K} \models_I \varphi$ if $\mathcal{K}, s_0 \models_I \varphi$ for all $s_0 \in S^0$, $\mathcal{K} \models \varphi$ if $\mathcal{K} \models_I \varphi$ for all I, and $\models \varphi$ if $\mathcal{K} \models \varphi$ for all $\mathcal{K}$. The remaining CTL temporal operators are defined:

$$\mathsf{O}_\eta \Diamond \varphi \equiv \mathsf{O}_\eta(\top \,\mathcal{U}\, \varphi) \quad \mathsf{P}_\eta \Diamond \varphi \equiv \mathsf{P}_\eta(\top \,\mathcal{U}\, \varphi)$$
$$\mathsf{O}_\eta \Box \varphi \equiv \neg \mathsf{P}_\eta \Diamond \neg \varphi \quad \mathsf{P}_\eta \Box \varphi \equiv \neg \mathsf{O}_\eta \Diamond \neg \varphi$$

Recalling that $\eta_\emptyset$ denotes the empty normative system, we obtain the conventional path quantifiers of CTL as follows: $\mathsf{A}\alpha \equiv \mathsf{O}_{\eta_\emptyset}\alpha$, $\mathsf{E}\alpha \equiv \mathsf{P}_{\eta_\emptyset}\alpha$.

Properties and Axiomatisation: The following Proposition makes precise the expected property that *a less liberal system has more obligations (and less permissions) than a more liberal system.*

Proposition 1. *Let $\mathcal{K}$ be a Kripke structure, I be an interpretation over $\mathcal{K}$, and $\eta_1, \eta_2 \in \Sigma_\eta$: If $I(\eta_1) \subseteq I(\eta_2)$ then $\mathcal{K} \models_I \mathsf{O}_{\eta_1}\varphi \rightarrow \mathsf{O}_{\eta_2}\varphi$ and $\mathcal{K} \models_I \mathsf{P}_{\eta_2}\varphi \rightarrow \mathsf{P}_{\eta_1}\varphi$.*

We now present a sound and complete axiomatisation for NTL and some of its variants. First, let NTL$^-$ be NTL without the empty normative system $\eta_\emptyset$. Formally, NTL$^-$ is defined exactly as NTL, except for the requirement that Σ_η contains the $\eta_\emptyset$ symbol and the corresponding restriction on interpretations. An axiom system for NTL$^-$, denoted $\vdash^-$, is defined by axioms and rules (Ax1)–(R2) in Figure 1. NTL$^-$ can be seen as a *multi-dimensional* variant of CTL, where there are several indexed versions of each path quantifier.

Going on to NTL, we add axioms (Obl) and (Perm) (Figure 1); the corresponding inference system is denoted $\vdash$. We then, have the following chain of implications in NTL (the second element in the chain is a variant of the deontic axiom discussed below). If something is naturally, or physically inevitable, then it is obligatory in any normative system; if something is an obligation within a given normative system η, then it is permissible in η; and if something is permissible in a given normative system, then it is naturally (physically) possible:

$$\vdash (\mathsf{A}\varphi \rightarrow \mathsf{O}_\eta\varphi) \qquad \vdash (\mathsf{O}_\eta\varphi \rightarrow \mathsf{P}_\eta\varphi) \qquad \vdash (\mathsf{P}_\eta\varphi \rightarrow \mathsf{E}\varphi)$$

Finally, let NTL$^+$ be the extension of NTL obtained by extending the logical language with propositions on the form $\eta \equiv \eta'$ and $\eta \sqsubset \eta'$ ($\sqsubseteq$ can then be defined), interpreted in the obvious way (e.g., $\mathcal{K}, s \models_I \eta \sqsubset \eta'$ iff $I(\eta) \subset I(\eta')$). An axiom system for NTL$^+$, denoted $\vdash^+$, is obtained from $\vdash^-$ by adding the schemes (Obl+) and (Perm+) (Figure 1).

(Ax1) All validities of propositional logic
(Ax2) $\mathsf{P}_\eta \Diamond \varphi \leftrightarrow \mathsf{P}_\eta(\top \mathcal{U} \varphi)$
(Ax2b) $\mathsf{O}_\eta \Box \varphi \leftrightarrow \neg \mathsf{P}_\eta \Diamond \neg \varphi$
(Ax3) $\mathsf{O}_\eta \Diamond \varphi \leftrightarrow \mathsf{O}_\eta(\top \mathcal{U} \varphi)$
(Ax3b) $\mathsf{P}_\eta \Box \varphi \leftrightarrow \neg \mathsf{O}_\eta \Diamond \neg \varphi$
(Ax4) $\mathsf{P}_\eta \bigcirc (\varphi \vee \psi) \leftrightarrow (\mathsf{P}_\eta \bigcirc \varphi \vee \mathsf{P}_\eta \bigcirc \psi)$
(Ax5) $\mathsf{O}_\eta \bigcirc \varphi \leftrightarrow \neg \mathsf{P}_\eta \bigcirc \neg \varphi$
(Ax6) $\mathsf{P}_\eta(\varphi \mathcal{U} \psi) \leftrightarrow (\psi \vee (\varphi \wedge \mathsf{P}_\eta \bigcirc \mathsf{P}_\eta(\varphi \mathcal{U} \psi)))$
(Ax7) $\mathsf{O}_\eta(\varphi \mathcal{U} \psi) \leftrightarrow (\psi \vee (\varphi \wedge \mathsf{O}_\eta \bigcirc \mathsf{O}_\eta(\varphi \mathcal{U} \psi)))$
(Ax8) $\mathsf{P}_\eta \bigcirc \top \wedge \mathsf{O}_\eta \bigcirc \top$
(Ax9) $\mathsf{O}_\eta \Box(\varphi \rightarrow (\neg\psi \wedge \mathsf{P}_\eta \bigcirc \varphi)) \rightarrow (\varphi \rightarrow \neg \mathsf{O}_\eta(\gamma \mathcal{U} \psi))$
(Ax9b) $\mathsf{O}_\eta \Box(\varphi \rightarrow (\neg\psi \wedge \mathsf{P}_\eta \bigcirc \varphi)) \rightarrow (\varphi \rightarrow \neg \mathsf{O}_\eta \Diamond \psi)$
(Ax10) $\mathsf{O}_\eta \Box(\varphi \rightarrow (\neg\psi \wedge (\gamma \rightarrow \mathsf{O}_\eta \bigcirc \varphi))) \rightarrow (\varphi \rightarrow \neg \mathsf{P}_\eta(\gamma \mathcal{U} \psi))$
(Ax10b) $\mathsf{O}_\eta \Box(\varphi \rightarrow (\neg\psi \wedge \mathsf{O}_\eta \bigcirc \varphi)) \rightarrow (\varphi \rightarrow \neg \mathsf{P}_\eta \Diamond \psi)$
(Ax11) $\mathsf{O}_\eta \Box(\varphi \rightarrow \psi) \rightarrow (\mathsf{P}_\eta \bigcirc \varphi \rightarrow \mathsf{P}_\eta \bigcirc \psi)$
(R1) If $\vdash \varphi$ then $\vdash \mathsf{O}_\eta \Box \varphi$ (generalization)
(R2) If $\vdash \varphi$ and $\vdash \varphi \rightarrow \psi$ then $\vdash \psi$ (modus ponens)
(Obl) $\mathsf{O}_{\eta_\emptyset} \alpha \rightarrow \mathsf{O}_\eta \alpha$
(Perm) $\mathsf{P}_\eta \alpha \rightarrow \mathsf{P}_{\eta_\emptyset} \alpha$
(Obl+) $\eta \sqsubseteq \eta' \rightarrow (\mathsf{O}_\eta \alpha \rightarrow \mathsf{O}_{\eta'} \alpha)$
(Perm+) $\eta \sqsubseteq \eta' \rightarrow (\mathsf{P}_{\eta'} \alpha \rightarrow \mathsf{P}_\eta \alpha)$

Fig. 1. The three systems NTL$^-$ ((Ax1)–(R2), derived from an axiomatisation of CTL); NTL ((Ax1)–(R2),(Obl),(Perm)); NTL$^+$ ((Ax1)–(R2),(Obl+),(Perm+)). α stands for a temporal formula.

Theorem 1 (Soundness and Completeness). *The inference mechanism $\vdash^-$ is sound and complete with respect to validity of* NTL$^-$ *formulas, i.e., for every formula φ in the language of* NTL$^-$*, we have $\models \varphi$ iff $\vdash^- \varphi$. The same holds for $\vdash$ with respect to formulas from* NTL *and $\vdash^+$ with respect to* NTL$^+$*.*

Proof. All three cases are proven by adjusting the technique presented in [Emerson, 1990]. For the NTL$^-$ *case, the tableau-based construction of [Emerson, 1990] immediately carries through: we will encounter, for every generated state, successors of different dimensions. For the case of* NTL*, which includes the symbol $\eta_\emptyset$, we have to add clauses corresponding to (Obl) and (Perm) to the construction of the closure $cl(\varphi)$ of a formula φ: if $\mathsf{O}_{\eta_\emptyset} \alpha$ (respectively, $\mathsf{P}_\eta \alpha$) is in $cl(\varphi)$ then also $\mathsf{O}_\eta \alpha$ (respectively, $\mathsf{P}_{\eta_\emptyset} \alpha$) should be in $cl(\varphi)$. In the case of* NTL$^+$*, we have to close off $cl(\varphi)$ under the implications of axioms (Obl+) and (Perm+).*

Going beyond NTL$^+$, we can impose further structure on Σ_η and its interpretations. For example, we can add unions and intersections of normative systems by requiring Σ_η to include symbols $\eta \sqcup \eta'$, $\eta \sqcap \eta'$ whenever it includes η and η', and require interpretations to interpret $\sqcup$ as set union and $\sqcap$ as set intersection. As discussed above, we must then further restrict interpretations such that $R \setminus (I(\eta_1) \cup I(\eta_2))$ always is total. This would give us a kind of calculus of normative systems. Let $\mathcal{K}$ be a Kripke structure and I be an interpretation with

the mentioned properties:

$$\mathcal{K} \models_I \mathsf{P}_{\eta \sqcup \eta'} \varphi \rightarrow \mathsf{P}_\eta \varphi \qquad \mathcal{K} \models_I \mathsf{P}_\eta \varphi \rightarrow \mathsf{P}_{\eta \sqcap \eta'} \varphi$$
$$\mathcal{K} \models_I \mathsf{O}_\eta \varphi \rightarrow \mathsf{O}_{\eta \sqcup \eta'} \varphi \qquad \mathcal{K} \models_I \mathsf{O}_{\eta \sqcap \eta'} \varphi \rightarrow \mathsf{O}_\eta \varphi$$

(all of which follow from Proposition 1). Having such a calculus allows one to reason about the composition of normative systems.

Relationship with Deontic Logic: The two main differences between the language of NTL and the language of conventional deontic logic (henceforth "deontic logic") are, first, *contextual* deontic operators allowing a formula to refer to several different normative systems, and, second, *temporal* operators. *All* deontic expressions in NTL refer to time: $\mathsf{P}_\eta \bigcirc \varphi$ ("it is permissible in the context of η that φ is true at the next time point"); $\mathsf{O}_\eta \Box \varphi$ ("it is obligatory in the context of η that φ always will be true"); etc. Deontic logic contains no notion of time. In order to compare our temporal deontic statements with those of deontic logic we must take the temporal dimension to be implicit in the latter. Two of the perhaps most natural ways of doing that is to take "obligatory" ($\mathsf{O}\varphi$) to mean "*always* obligatory" ($\mathsf{O}_\eta \Box \varphi$), or "obligatory at the *next point in time*" ($\mathsf{O}_\eta \bigcirc \varphi$), respectively, and similarly for permission. In either case, all the principles of *Standard Deontic Logic* (SDL) hold also for NDL, viz., $\mathsf{O}(\varphi \rightarrow \psi) \rightarrow (\mathsf{O}\varphi \rightarrow \mathsf{O}\psi)$ (K); $\neg \mathsf{O} \bot$ (D); and from φ infer $\mathsf{O}\varphi$ (N). The two mentioned temporal interpretations of the (crucial) deontic axiom D are (both NTL validities):

$$\neg \mathsf{O}_\eta \Box \bot \text{ and } \neg \mathsf{O}_\eta \bigcirc \bot$$

With these translations, all of the most commonly discussed so-called paradoxes of deontic logic also holds in NTL. However, it has been argued (cf., e.g., [Meyer and Wieringa, 1993]) that one of the causes behind some of the instances of the paradoxes (particularly those involving contrary-to-duty obligations) is that the language of conventional deontic logic is too weak, and that by incorporating temporal operators some instances of the paradoxes can be avoided.

3 Symbolic Representations

In practice, explicit state representations of Kripke structures are rarely if ever used when reasoning about systems, because of the *state explosion problem*: given a system with n Boolean variables, the system will typically have 2^n states. Instead, practical reasoning tools provide *succinct*, *symbolic* representation languages for defining Kripke structures. We present such a language for defining models, and also introduce an associated symbolic language for defining normative systems.

A Symbolic Language for Models: We present the SIMPLE REACTIVE MODULES LANGUAGE (SRML), a "stripped down" version of Alur and Henzinger's REACTIVE

MODULES LANGUAGE (RML) [Alur and Henzinger, 1999], which was introduced in [Hoek *et al.*, 2006]. SRML represents the core of RML, with some "syntactic sugar" removed to simplify the presentation and semantics. The basic idea is to present a Kripke structure $\mathcal{K}$ by means of a number of symbolically represented agents, where the choices available to every agent are defined by a number of rules, defining which actions are available to the agent in every state; a transition (s, s') in $\mathcal{K}$ corresponds to a *tuple of actions, one for each agent in the system*. Here is an example of an agent definition in SRML (agents are referred to as "modules" in (S)RML):

```
module toggle controls x
  init
  ℓ1 : ⊤ ⇝ x′ := ⊤
  ℓ2 : ⊤ ⇝ x′ := ⊥
  update
  ℓ3 : x ⇝ x′ := ⊥
  ℓ4 : (¬x) ⇝ x′ := ⊤
```

This module, named *toggle*, controls a single Boolean variable, x. The choices available to the agent are defined by the `init` and `update` rules[2]. The `init` rules define the choices available to the agent with respect to the initialisation of its variables, while the `update` rules define the agent's choices subsequently. In this example, there are two `init` rules and two `update` rules. The `init` rules define two choices for the initialisation of variable x: assign it the value $\top$ or the value $\bot$. Both of these rules can fire initially, as their conditions ($\top$) are always satisfied; in fact, only one of the available rules will ever *actually* fire, corresponding to the "choice made" by the agent on that decision round. The effect of firing a rule is to execute the assignment statements on the r.h.s. of the rule, which modify the agent's controlled variables. (The "prime" notation for variables, e.g., x', means "the value of x afterwards".) Rules are identified by *labels* (ℓ_i); these labels do not form part of the original RML language, and in fact play no part in the semantics of SRML – they are used to identify rules in normative systems, as we shall see below. We assume a distinguished label "$\Box$" for rules, which is used to identify rules that should never be made illegal by any normative system. With respect to `update` rules, the first rule says that if x has the value $\top$, then the corresponding action is to assign it the value $\bot$, while the second rule says that if x has the value $\bot$, then it can subsequently be assigned the value $\top$. In sum, the module non-deterministically chooses a value for x initially, and then on subsequent rounds toggles this value. In this example, the `init` rules are non-deterministic, while the `update` rules are deterministic. An SRML *system*, ρ, is a set of such modules, where the controlled variables of modules are mutually disjoint.

The Kripke structure $\mathcal{K}_\rho = \langle S_\rho, S^0_\rho, R_\rho, V_\rho \rangle$ corresponding to SRML system ρ is given as follows: the state set S_ρ and valuation function V_ρ corresponds to states (valuations of variables) that could be reached by ρ, with initial states S^0_ρ being states that could be generated by `init` rules; the transition relation R_ρ

[2] To be more precise, the rules are *guarded commands*.

is defined by $(s, s') \in R_\rho$ iff there exists a tuple of `update` rules, one for each module in the system, such that each rule is enabled in s and s' is obtained from executing this collection of rules on s.

A Symbolic Language for Normative Systems: We now introduce the SRML *Norm Language* (SNL) for representing normative systems, which corresponds to the SRML language for models. The general form of an SNL normative system definition is:

$$\begin{array}{l}\texttt{normative-system}\ id\\ \quad \chi_1\ \texttt{disables}\ \ell_{1_1}, \ldots, \ell_{1_k}\\ \quad \ldots\\ \quad \chi_m\ \texttt{disables}\ \ell_{m_1}, \ldots, \ell_{m_k}\end{array}$$

Here, $id \in \Sigma_\eta$ is the name of the normative system; these names will be used to refer to normative systems in formulae of NTL. The body of the normative system is defined by a set of *constraint rules*. A constraint rule

$$\chi\ \texttt{disables}\ \ell_1, \ldots, \ell_k$$

consists of a condition part χ, which is a propositional logic formula over the variables of the system, and a set of rule labels $\{\ell_1, \ldots, \ell_k\}$ (we require $\square \notin \{\ell_1, \ldots, \ell_k\}$). If χ_i is satisfied in a particular state, then *any* SRML *rule with a label that appears on the r.h.s. of the constraint rule will be illegal in that state, according to this normative system.* An SNL *interpretation* is then simply a set of SNL normative systems, each with a distinct name.

Given SNL normative systems η_1 and η_2, for some SRML system ρ, we say: η_1 is *at least as liberal* as η_2 in system ρ if for every state $s \in S_\rho$, every rule that is legal according to η_2 is legal according to η_1; and they are *equivalent* if for every state $s \in S_\rho$, the set of rules legal according to η_1 and η_2 are the same.

Theorem 2. *The problem of testing whether* SNL *normative system* η_1 *is at least as liberal as* SNL *normative system* η_2 *is* PSPACE*-complete, as is the problem of testing equivalence of such systems.*

Proof. We do the proof for checking equivalence; the liberality case is similar. For membership of PSPACE, consider the complement problem: guess a state s, check that $s \in S_\rho$, (reachability of states in RML is in PSPACE [Alur and Henzinger,]) and check that there is some rule legal in s according to η_2 is not legal in s according to η_1, or vice versa. Hence the complement problem is in NPSPACE, and so the problem is in PSPACE. For PSPACE-hardness, we reduce the problem of propositional invariant checking over (S)RML modules [Alur and Henzinger,]. Given an SRML system ρ and propositional formula φ, define normative systems η_1 and η_2 as follows (where ℓ does not occur in ρ):

$$\begin{array}{ll}\texttt{normative-system}\ \eta_1 & \texttt{normative-system}\ \eta_2\\ \quad \neg\varphi\ \texttt{disables}\ \ell & \quad \bot\ \texttt{disables}\ \ell\end{array}$$

According to η_2, ℓ is always enabled; thus η_1 will be equivalent to η_2 iff φ holds across all reachable states of the system.

4 Model Checking

Model checking is an important computational problem for any modal or temporal logic [Clarke *et al.*, 2000]. We consider two versions of the model checking problem for NTL, depending on whether the model is presented explicitly or symbolically. For each of these cases, there are two further possibilities, depending on whether we are given an interpretation I for normative systems named in formulae or not. If we are given an interpretation for the normative systems named in the formula, then NTL model checking essentially amounts to a conventional verification problem: showing that, under the given interpretation, the model and associated normative systems have certain properties. However, the *uninterpreted* model checking problem corresponds to the *synthesis* of normative systems: we ask whether *there exist* normative systems that would have the desired properties.

Explicit State Model Checking: The *interpreted explicit state model checking problem* for NTL is as follows.

> Given a Kripke structure $\mathcal{K} = \langle S, S^0, R, V\rangle$, interpretation $I : \Sigma_\eta \rightarrow N(R)$ and formula φ of NTL, is it the case that $\mathcal{K} \models_I \varphi$?

The CTL model checking problem is P-complete [Schnoebelen, 2003]. The standard dynamic programming algorithm for CTL model checking may be easily adapted for interpreted explicit state NTL model checking, and has the same worst case time complexity. More interesting is the case where we are *not* given an interpretation. The *uninterpreted explicit state model checking problem* for NTL is as follows.

> Given a Kripke structure $\mathcal{K} = \langle S, S^0, R, V\rangle$ and formula φ of NTL, does there exist an interpretation $I : \Sigma_\eta \rightarrow N(R)$ such that $\mathcal{K} \models_I \varphi$?

Theorem 3. *The uninterpreted explicit state model checking problem for* NTL *is* NP*-complete.*

Proof. For membership in NP, simply guess an interpretation I and verify that $\mathcal{K} \models_I \varphi$. Since interpretations are polynomial in the size of the Kripke structure and formula, guessing can be done in (nondeterministic) polynomial time, and checking is the interpreted explicit state model checking problem. Hence the problem is in NP. For NP-hardness, we reduce SAT. Given SAT instance φ over variables $x_1, \ldots, x_k$, for each variable x_i, create two variables $t(x_i)$ and $f(x_i)$, and define a Kripke structure with $3k + 1$ states, as illustrated in Figure 2; state s_0 is the initial state, and state s_{3k} is a final state. Let φ^* denote the NTL formula obtained from φ by systematically replacing every variable x_i with $(\mathsf{P}_\eta \Diamond t(x_i))$. Define the formula to be model checked as:

$$\varphi^* \wedge \bigwedge_{1\leq i\leq k}(\mathsf{P}_\eta \Diamond (t(x_i) \vee f(x_i))) \wedge$$
$$\bigwedge_{1\leq i\leq k}(\mathsf{P}_\eta \Diamond t(x_i) \rightarrow \neg \mathsf{P}_\eta \Diamond f(x_i))(\mathsf{P}_\eta \Diamond f(x_i) \rightarrow \neg \mathsf{P}_\eta \Diamond t(x_i))$$

This formula is satisfied in the structure by some interpretation iff φ is satisfiable.

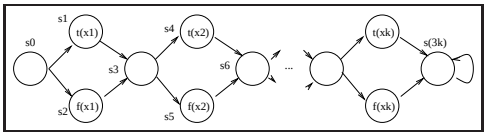

Fig. 2. Reduction for Theorem 3.

Symbolic Model Checking: As we noted above, explicit state model checking problems are perhaps of limited interest, since such representations are exponentially large in the number of propositional variables. Thus we now consider the SRML *model checking problem for* NTL. Again, we have two versions, depending on whether we are given an interpretation or not.

Theorem 4. *The interpreted* SRML *model checking problem for* NTL *is* PSPACE*-complete.*

Proof. PSPACE-hardness is by a reduction from the problem of propositional invariant verification for SRML [Alur and Henzinger,][3]. Given a propositional formula φ and an (S)RML system ρ, let $I = \{\eta_\emptyset\}$, and simply check whether $\mathcal{K}_\rho \models_I \mathsf{O}_{\eta_\emptyset} \Box\varphi$. Membership of PSPACE is by adapting the CTL symbolic model checking algorithm of [Cheng, 1995].

Theorem 5. *The uninterpreted* SRML *model checking problem for* NTL *is* EXPTIME*-hard.*

Proof. By reduction from the problem of determining whether a given player has a winning strategy in the two-player game PEEK-G_4 [Stockmeyer and Chandra, 1979, p.158]. An instance of PEEK-G_4 is a quad $\langle X_1, X_2, X_3, \varphi\rangle$ where: X_1 and X_2 are disjoint, finite sets of Boolean variables – variables X_1 are under the control of agent 1, and X_2 are under the control of agent 2; $X_3 \subseteq (X_1 \cup X_2)$ are the variables true in the initial state of the game; and φ is a propositional formula over $X_1 \cup X_2$, representing the winning condition. The agents take try to make φ true, by taking it in turns to alter the value of at most one of their variables. The decision problem is to determine whether agent 2 has a winning strategy in a given game. The idea of the proof is to define an SRML system that such that the runs of the system correspond to plays of the given game instance, and then to define an NTL formula to be model checked, which names a normative system η, such that the transitions legal according to η correspond to a winning strategy for player 2. The construction of the SRML system follows that of the EXPTIME-completeness proof of ATL model checking in [Hoek *et al.*, 2006], with the difference that player 2's update rules are given labels (so that they may be disabled). The formula to model check then defines three properties: (*i*) if it is agent 2's turn, then according to η at most one of its possible moves is legal; (*ii*) all of agent 1's moves are legal according to η (i.e, agent 2 must win against

[3] In fact, the result of [Alur and Henzinger, 1999] is stated for RML, but the proof only makes use of features from SRML.

all of these); and (*iii*) the legal paths according to η must represent wins for agent 2.

5 Example: Traffic Norms

Consider a circular road, with two parallel lanes. Vehicles circulate on the two lanes clockwise. We consider two types of vehicles: cars, and ambulances. The road is discretised in a finite number of positions, each one represented as an instance of a proposition $at(\mathit{lane\text{-}number}, \mathit{lane\text{-}position}, \mathit{vehicle\text{-}id})$. Thus $at(2, 5, \mathit{car}23)$ means agent $\mathit{car}23$ is on lane 2 at position 5 (lane 1 is the outer lane, lane 2 is the inner lane). We also refer to lane 1 as the left lane and to lane 2 as the right lane. At each time step, cars can either remain still or change their position by one unit, moving either straight or changing lane. Ambulances can remain still or change their position by one or two units, either straight or changing lanes at will. We are interested in normative systems that prevent crashes, and that permit ambulances take priority over private cars. So consider the following normative systems:

- η_1: Ambulances have priority over all other vehicles (i.e., cars stop to allow ambulances overtake them);
- η_2: Cars cannot use the rightmost (priority) lane;
- η_3: Vehicles have "right" priority (i.e., left lane has to give way to any car running in parallel on the right lane).

We modelled this scenario using an RML-based model checking system for ATL [Alur *et al.*, 2002]. Each vehicle is modelled as a module containing the rules that determine their physically legal movements, and global traffic control is modelled as a set of norms that constrain the application of certain rules. For example, here is the (somewhat simplified) definition of a car (we abuse notation to facilitate comprehension: for example addition and subtraction here are modulo-n operations, where n is the number of road positions, and the $at(\ldots)$ predicates are implemented as propositions):

```
module car-23 controls at(l,p,car-23)
init
  [] // initialise ...
update
 car-23-straight:
  at(l,p,car-23) &  not(at(l,p+1,car-1)) &
  ... & not(at(l,p+1,vehicle-n)) ->
   at(l,p+1,car-23)' := T, at(l,p,car-23)' := F;
 car-23-right:
  at(1,p,car-23) &  not(at(2,p+1,car-1)) &
  ... & not(at(2,p+1,vehicle-n)) ->
   at(2,p+1,car-23)' := T, at(1,p,car-23)' := F;
 car-23-left:
  at(2,p,car-23) & not(at(1,p+1,car-1)) &
```

```
  ... & not(at(1,p+1,vehicle-n)) ->
   at(1,p+1,car-23)' := T, at(2,p,car-23)' := F;
 car-23-still:
  T -> skip;
```

We can then define the norms described above using SNL; (again, we abuse notation somewhat in the interests of brevity; variables must be expanded out for each car and position, in the obvious way):

```
normative-system N1
 at(1,p,car-i) and at(1,p-1,amb-j) disables
  car-i-straight, car-i-left, car-i-right;

normative-system N2
 at(2,p,car-i) disables
  car-i-straight, car-i-still;
   at(1,p,car-i) disables car-i-right;

normative-system N3
    at(1,p,car-i) and at(2,p,car-j) disables
       car-i-straight, car-i-right;
```

Using a model checker, we can then evaluate properties of the system; e.g., if there is only one ambulance then we have $\mathsf{O}_{\eta_1 \cup \eta_2 \cup \eta_3} \Box \neg crash$.

6 Conclusions & Acknowledgments

Several issues present themselves for future work: tight bounds for complexity of uninterpreted symbolic model checking, the complexity of satisfiability, and a full implementation of a model checker encompassing the variations discussed in section 4 are the most obvious.

We gratefully acknowledge support from the Spanish research project Web(I)-2, TIC2003-08763-c02-00), the EU funded project OpenKnowledge, FP6-027253), the UK EPSRC project "Virtual Organisations for E-Science" and the Research Council of Norway (project 166525/V30).

References

Alur and Henzinger, . R. Alur and T. A. Henzinger. *Computer aided verification.* In press.

Alur and Henzinger, 1999. R. Alur and T. A. Henzinger. Reactive modules. *Form. Meth. Sys. Des.*, 15(11), 1999.

Alur *et al.*, 2002. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *JACM*, 49(5):672–713, 2002.

Cheng, 1995. A. Cheng. Complexity results for model checking. Tech. Rep. RS-95-18, Uni. Aarhus, 1995.

Clarke *et al.*, 2000. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press 2000.

Emerson, 1990. E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Hand. of Theor. Comp. Sci. Vol. B*, pages 996–1072. Elsevier, 1990.

Hoek *et al.*, 2006. W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. In *Proc. AAMAS-2006*, 2006.

Meyer and Wieringa, 1993. J.-J. Ch. Meyer and R. J. Wieringa, eds. *Deontic Logic in Comp. Sci.*. Wiley, 1993.

Schnoebelen, 2003. P. Schnoebelen. The complexity of temporal logic model checking. In *Advances in Modal Logic Vol 4*, 2003.

Shoham and Tennenholtz, 1996. Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. In *Computational Theories of Interaction and Agency*. MIT Press, 1996.

Stockmeyer and Chandra, 1979. L. J. Stockmeyer and A. K. Chandra. Provably difficult combinatorial games. *SIAM Jnl of Comp.*, 8(2):151–174, 1979.

Wooldridge and van der Hoek, 2005. M. Wooldridge and W. van der Hoek. On obligations and normative ability. *Jnl Appl. Logic*, 4(3-4):396–420, 2005.