Modelling unreliable and untrustworthy agent behaviour

Marek Sergot

Department of Computing, Imperial College London
180 Queen's Gate, London SW7 2BZ, UK
mjs@doc.ic.ac.uk

Summary. It cannot always be assumed that agents will behave as they are supposed to behave. Agents may fail to comply with system norms deliberately, in open agent systems or other competitive settings, or unintentionally, in unreliable environments because of factors beyond their control. In addition to analysing system properties that hold if specifications/norms are followed correctly, it is also necessary to predict, test, and verify the properties that hold if system norms are violated, and to test the effectiveness of introducing proposed control, enforcement, and recovery mechanisms. $\mathcal{C}+^{++}$ is an extended form of the action language $\mathcal{C}+$ of Giunchiglia, Lee, Lifschitz, McCain, and Turner, designed for representing norms of behaviour and institutional aspects of (human or computer) societies. We present the permission component of $\mathcal{C}+^{++}$ and then illustrate on a simple example how it can be used in conjunction with standard model checkers for the temporal logic CTL to verify system properties in the case where agents may fail to comply with system norms.

1 Introduction

It is a common assumption in many multi-agent systems that agents will behave as they are intended to behave. Even in systems such as IMPACT [1], where the language of 'obligation' and 'permission' is employed in the specification of agent behaviour, there is an explicit, built-in assumption that agents always fulfill their obligations and never perform actions that are prohibited. For systems constructed by a single designer and operating on a stable and reliable platform, this is a perfectly reasonable assumption.

There are at least two main circumstances in which the assumption must be abandoned. In *open agent societies*, where agents are programmed by different parties, where there is no direct access to an agent's internal state, and where agents do not necessarily share a common goal, it cannot be assumed that all agents will behave according to the system norms that govern their behaviour. Agents must be assumed to be untrustworthy because they act on behalf of parties with competing interests, and so may fail, or even choose not to, conform to the society's norms in order to achieve their individual goals.

It is then usual to impose sanctions to discourage norm violating behaviour and to provide some form of reparation when it does occur. The second circumstance is where agents may fail to behave as intended because of factors beyond their control. This is likely to become commonplace as multi-agent systems are increasingly deployed on dynamic distributed environments. Agents in such circumstances are unreliable, but not because they deliberately seek to gain unfair advantage over others. Imposition of sanctions to discourage norm violating behaviour is pointless, though there is a point to specifying reparation and recovery norms. There is a third, less common, circumstance, where deliberate violations may be allowed in order to deal with exceptional or unanticipated situations. An example of discretionary violation of access control policies in computer security is discussed in [2].

In all these cases it is meaningful to speak of obligations and permissions, and to describe agent behaviour as governed by norms, which may be violated, accidentally or on purpose. In addition to analysing system properties that hold if specifications/norms are followed correctly, it is also necessary to predict, test, and verify the properties that hold if these norms are violated, and to test the effectiveness of introducing proposed control, enforcement, and recovery mechanisms.

In previous work [3, 4, 5] we presented a framework for specifying open agent societies in terms of permissions, obligations, and other more complex normative relations. Norms are represented in various action formalisms in order to provide an executable specification of the agent society. This work, however, did not address verification of system properties, except in a limited sense. In another strand of work [6, 7], we have addressed verification of system properties but that was in the specific context of reasoning about knowledge in distributed and multi-agent systems. We showed that by adding a simple deontic component to the formalism of 'interpreted systems' [8] it is possible to determine formally which of a system's critical properties are compromised when agents fail to behave according to prescribed communication protocols, and then to determine formally the effectiveness of introducing additional controller agents whose role is to enforce compliance.

In this paper we conduct a similar exercise, but focussing now on agent behaviours generally rather than on communication and epistemic properties specifically. We present the main elements of a formalism $\mathcal{C}+^{++}$ which we have been developing for representing norms of behaviour and institutional aspects of (human or computer) societies [9]; we then present a simple example to sketch how it can be used in modelling unreliable/untrustworthy agent behaviour.

$\mathcal{C}+^{++}$ is an extended form of the action language $\mathcal{C}+$ of Giunchiglia, Lee, Lifschitz, McCain, and Turner [10], a formalism for specifying and reasoning about the effects of actions and the persistence ('inertia') of facts over time. An 'action description' in $\mathcal{C}+$ is a set of $\mathcal{C}+$ rules which define a transition system of a certain kind. Implementations supporting a range of querying and planning tasks are available, notably in the form of the 'Causal Calculator'

CCalc. Our extended version $\mathcal{C}+^{++}$ provides two main extensions. The first is a means of expressing 'counts as' relations between actions, also referred to as 'conventional generation' of actions. This feature will not be discussed in this paper. The second extension is a way of specifying the permitted (acceptable, legal) states of a transition system and its permitted (acceptable, legal) transitions. This will be the focus of attention in this paper.

A main attraction of the $\mathcal{C}+$ formalism compared to other action languages in the AI literature is that it has an explicit semantics in terms of transition systems and also a semantics in terms of a nonmonotonic formalism ('causal theories', summarised below) which provides a route to implementation via translations to executable logic programs. The emphasis in this paper is on the transition system semantics. Transition systems provide a bridge between AI formalisms and standard methods in other areas of computer science. We exploit this link by applying standard temporal logic model checkers to verify system properties of transition systems defined using the language $\mathcal{C}+^{++}$.

Two points of clarification: (1) We do not distinguish in this paper between deliberate and unintentional norm violation, and (2) we are modelling agent behaviour from an external "bird's eye" perspective. We do not discuss an agent's internal state or its internal reasoning mechanisms.

2 The language $\mathcal{C}+$

The language $\mathcal{C}$ was introduced by Giunchiglia and Lifschitz [11]. It applies the ideas of 'causal theories' to reasoning about the effects of actions and the persistence ('inertia') of facts ('fluents'), building on earlier suggestions by McCain and Turner. $\mathcal{C}+$ extends $\mathcal{C}$ by allowing multi-valued fluents as well as Boolean fluents and generalises the form of rules in various ways. The definitive presentation of $\mathcal{C}+$, and its relationship to 'causal theories', is [10]. An implementation supporting a range of querying and planning tasks is available in the form of the Causal Calculator (CCalc)[1].

We present here a concise, and necessarily rather dense, summary of the language. Some features (notably 'statically determined fluents') are omitted for simplicity. There are also some minor syntactic and terminological differences from the version presented in [10], and we give particular emphasis to the transition system semantics.

Syntax and semantics We begin with σ, a multi-valued, propositional signature, which is partitioned into a (non-empty) set σ^{f} of *fluent constants* and a (non-empty) set σ^{a} of *action constants*. For each constant $c \in \sigma$ there is a finite, non-empty set $dom(c)$ of *values*. For simplicity, in this paper we will assume that each $dom(c)$ has at least two elements. An *atom* of the signature is an expression $c{=}v$, where $c \in \sigma$ and $v \in dom(c)$. $c{=}v$ is a *fluent atom* when $c \in \sigma^{\mathrm{f}}$ and an *action atom* when $c \in \sigma^{\mathrm{a}}$. A *Boolean* constant is one whose

[1] `http://www.cs.utexas.edu/users/tag/cc`

domain is the set of truth values $\{\mathsf{t}, \mathsf{f}\}$. When c is a Boolean constant, we often write c for $c{=}\mathsf{t}$ and $\neg c$ as a shorthand for $c{=}\mathsf{f}$. Formulas are constructed from the atoms using the usual propositional connectives. The expressions $\top$ and $\bot$ are 0-ary connectives, with the usual interpretation. A *fluent formula* is a formula whose constants all belong to σ^{f}; an *action formula* is a formula whose constants all belong to σ^{a}, except that $\top$ and $\bot$ are fluent formulas but not action formulas.

An *interpretation* of a multi-valued signature σ is a function mapping every constant c to some $v \in \mathit{dom}(c)$; an interpretation X is said to *satisfy* an atom $c{=}v$ if $X(c) = v$, and in this case we write $X \models c{=}v$. The satisfaction relation $\models$ is extended from atoms to formulas in accordance with the standard truth tables for the propositional connectives. We let the expression $\mathrm{I}(\sigma)$ stand for the set of interpretations of σ. For convenience, we adopt the convention that an interpretation X of σ is identified with the set of atoms that are satisfied by X, i.e., $X \models c{=}v$ iff $c{=}v \in X$ for any atom $c{=}v$ of σ.

Every action description D of $\mathcal{C}+$ defines a labelled transition system $\langle S, \mathbf{A}, R\rangle$ where

- S is a (non-empty) set of *states*, each of which is an interpretation of the fluent constants σ^{f} of D; $S \subseteq \mathrm{I}(\sigma^{\mathrm{f}})$;
- $\mathbf{A}$ is a set of *transition labels*, sometimes referred to as *action labels* or *events*; $\mathbf{A}$ is the set of interpretations of the action constants σ^{a}, $\mathbf{A} = \mathrm{I}(\sigma^{\mathrm{a}})$;
- R is a set of transitions, $R \subseteq S \times \mathbf{A} \times S$.

For example: suppose there are three agents, a, b, and c which can move in direction E, W, N, or S, or remain idle. Suppose (for the sake of an example) that they can also whistle as they move. Let the action signature consist of action constants $\mathit{move}(a)$, $\mathit{move}(b)$, $\mathit{move}(c)$ with domains $\{E, W, N, S, \mathit{idle}\}$, and Boolean action constants $\mathit{whistle}(a)$, $\mathit{whistle}(b)$, $\mathit{whistle}(c)$. Then one possible interpretation of the action signature, and therefore one possible transition label, is $\{\mathit{move}(a){=}E, \mathit{move}(b){=}N, \mathit{move}(c){=}\mathit{idle}, \mathit{whistle}(a), \neg\mathit{whistle}(b), \mathit{whistle}(c)\}$.

Because every transition label ϵ is an interpretation of the action signature σ^{a}, action formulas α can be evaluated on the transition labels. We sometimes say that a transition (s, ϵ, s') is a transition of type α when $\epsilon \models \alpha$.

An action description D in $\mathcal{C}+$ is a set of *causal laws*, which are expressions of the following three forms. A *static law* is an expression:

$$F \ \mathsf{if}\ G \tag{1}$$

where F and G are fluent formulas. Static laws express constraints on states. A *fluent dynamic law* is an expression:

$$F \ \mathsf{if}\ G \ \mathsf{after}\ \psi \tag{2}$$

where F and G are fluent formulas and ψ is any formula of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$. Informally, (2) states that fluent formula F is satisfied by the resulting state

s' of any transition (s, ϵ, s') with $s \cup \epsilon \models \psi$, as long as fluent formula G is also satisfied by s'. Some examples follow. An *action dynamic law* is an expression:

$$\alpha \text{ if } \psi \tag{3}$$

where α is an action formula and ψ is any formula of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$. Action dynamic laws are used to express, among other things, that any transition of type α must also be of type α' (α' if α), or that any transition from a state satisfying fluent formula G must be of type β (β if G). Examples will be provided in later sections.

The $\mathcal{C}+$ language provides various abbreviations for common forms of causal laws. We will employ the following in this paper.

α causes F if G expresses that fluent formula F is satisfied by any state following the occurrence of a transition of type α from a state satisfying fluent formula G. It is shorthand for the dynamic law F if $\top$ after $G \wedge \alpha$. α causes F is shorthand for F if $\top$ after α.

nonexecutable α if G expresses that there is no transition of type α from a state satisfying fluent formula G. It is shorthand for the fluent dynamic law $\bot$ if $\top$ after $G \wedge \alpha$, or α causes $\bot$ if G.

inertial f states that values of the fluent constant f persist by default ('inertia') from one state to the next. It is shorthand for the collection of fluent dynamic laws $f{=}v$ if $f{=}v$ after $f{=}v$ for every $v \in dom(f)$.

Of most interest are *definite* action descriptions, which are action descriptions in which the head of every law (static, fluent dynamic, or action dynamic) is either an atom or the symbol $\bot$, and in which no atom is the head of infinitely many laws of D. We will restrict attention to definite action descriptions in this paper.

Now for the semantics. (See [9] for further details.)

Let $\mathrm{T}_{static}(s)$ stand for the heads of all static laws in D whose bodies are satisfied by s; let $E(s, \epsilon, s')$ stand for the heads of all fluent dynamic laws in D whose bodies are satisfied by the transition (s, ϵ, s'); and let $A(\epsilon, s)$ stand for the heads of all action dynamic laws whose bodies are satisfied by the transition (s, ϵ, s').

$$\begin{aligned}
\mathrm{T}_{static}(s) &=_{\mathrm{def}} \{F \mid F \text{ if } G \text{ is in } D,\ s \models G\} \\
E(s, \epsilon, s') &=_{\mathrm{def}} \{F \mid F \text{ if } G \text{ after } \psi \text{ is in } D,\ s' \models G,\ s \cup \epsilon \models \psi\} \\
A(\epsilon, s) &=_{\mathrm{def}} \{\alpha \mid \alpha \text{ if } \psi \text{ is in } D,\ s \cup \epsilon \models \psi\}
\end{aligned}$$

Let D be a definite action description and σ^{f} its fluent signature. A set s of fluent atoms is a state of D iff it satisfies the static laws of D, that is, iff

- $s \models \mathrm{T}_{static}(s)$ (i.e., $\mathrm{T}_{static}(s) \subseteq s$)

(s, ϵ, s') is a transition of D iff s and s' are interpretations of the fluent signature σ^{f} and ϵ is an interpretation of the action signature σ^{a} such that:

- $s \models \mathrm{T}_{static}(s) \quad (\mathrm{T}_{static}(s) \subseteq s;\ s \text{ is a state of } D)$
- $s' = \mathrm{T}_{static}(s') \cup E(s, \epsilon, s')$
- $\epsilon \models A(\epsilon, s) \quad (A(\epsilon, s) \subseteq \epsilon)$

One can see from the definition that s' is a state of D when (s, ϵ, s') is a transition of D.

Paths Finally, when $\langle S, \mathbf{A}, R\rangle$ is a labelled transition system, a *path* of length m is a sequence $s_0\, \epsilon_0\, s_1\, \cdots\, s_{m-1}\, \epsilon_{m-1}\, s_m \quad (m \geq 0)$ such that $(s_{i-1}, \epsilon_{i-1}, s_i) \in R$ for $i \in 1..m$. We will also be interested in infinite (ω length) paths.

Causal theories The language $\mathcal{C}+$ can be regarded as a higher-level notation for defining particular classes of theories in the non-monotonic formalism of 'causal theories', and indeed this is how it is presented in [10]. For present purposes the important points are these: for every (definite) action description D and non-negative integer m there is a natural translation from D to a causal theory Γ^D_m which encodes the paths of length m in the transition system defined by D; moreoever, for every definite causal theory Γ^D_m there is a formula $comp(\Gamma^D_m)$ of (classical) propositional logic whose (classical) models are in 1-1 correspondence with the paths of length m in the transition system defined by D. Thus, one method of computation for $\mathcal{C}+$ action descriptions is to construct the formula $comp(\Gamma^D_m)$ from the action description D and then employ a (standard, classical) satisfaction solver to determine the models of $comp(\Gamma^D_m)$. This is the method employed in the 'Causal Calculator' CCalc.

We summarise the main steps for completeness; the reader may wish to skip the details on first reading. A full account is given in [10].

A causal theory of signature σ is a set of expressions ('causal rules') of the form

$$F \Leftarrow G$$

where F and G are formulas of signature σ. F is the head of the rule and G is the body. A rule $F \Leftarrow G$ is to be read as saying that F is 'caused' if G is true, or (perhaps better), that there is an explanation for the truth of F if G is true.

Let Γ be a causal theory and let X be an interpretation of its signature. The *reduct* Γ^X is the set of all rules of Γ whose bodies are satified by the interpretation X: $\Gamma^X =_{\mathrm{def}} \{F \mid F \Leftarrow G \text{ is a rule in } \Gamma \text{ and } X \models G\}$. X is a *model* of Γ iff X is the unique model (in the sense of multi-valued signatures) of Γ^X.

A causal theory Γ is *definite* if the head of every rule of Γ is an atom or $\bot$, and no atom is the head of infinitely many rules of Γ. Every *definite* causal theory Γ can be translated into a formula $comp(\Gamma)$ of (classical) propositional logic via the process of 'literal completion': for each atom $c{=}v$ construct the formula $c{=}v \leftrightarrow G_1 \vee \cdots \vee G_n$ where $G_1, \ldots, G_n$ $(n \geq 0)$ are the bodies of the rules of Γ with head $c{=}v$; $comp(\Gamma)$ is the conjunction of all such formulas together with formulas $\neg F$ for each rule of the form $\bot \Leftarrow F$ in Γ. The models

of a definite causal theory Γ are precisely the (classical) models of its literal completion, $comp(\Gamma)$.

Given an action description D in $\mathcal{C}+$, and any non-negative integer m, translation to the corresponding causal theory Γ_m^D proceeds as follows. The signature of Γ_m^D is obtained by time-stamping every fluent and action constant of D with non-negative integers between 0 and m: the (new) atom $f[i]{=}v$ represents that fluent $f{=}v$ holds at integer time i, or more precisely, that $f{=}v$ is satisfied by the state s_i of a path $s_0\,\epsilon_0\,\cdots\,\epsilon_{m-1}\,s_m$ of the transition system defined by D; the atom $a[i]{=}v$ represents that action atom $a{=}v$ is satisfied by the transition ϵ_i of such a path. In what follows, $\psi[i]$ is shorthand for the formula obtained by replacing every atom $c{=}v$ in ψ by the timestamped atom $c[i]{=}v$.

Now, for every static law F **if** G in D and every $i \in 0\,..\,m$, include in Γ_m^D a causal rule of the form

$$F[i] \Leftarrow G[i]$$

For every fluent dynamic law F **if** G **after** ψ in D and every $i \in 0\,..\,m{-}1$, include a causal rule of the form

$$F[i{+}1] \Leftarrow G[i{+}1] \wedge \psi[i]$$

And for every action dynamic law α **if** ψ in D and every $i \in 0\,..\,m{-}1$, include a causal rule of the form

$$\alpha[i] \Leftarrow \psi[i]$$

We also require the following 'exogeneity laws'. For every fluent constant f and every $v \in dom(f)$, include a causal rule:

$$f[0]{=}v \Leftarrow f[0]{=}v$$

And for every action constant a, every $v \in dom(a)$, and every $i \in 0\,..\,m{-}1$, include a causal rule:

$$a[i]{=}v \Leftarrow a[i]{=}v$$

(There are some further complications in the full $\mathcal{C}+$ language concerning 'statically determined' fluents and non-exogenous actions, which we are ignoring here for simplicity.)

It is straightforward to check [10] that the models of causal theory Γ_m^D, and hence the (classical) models of the propositional logic formula $comp(\Gamma_m^D)$, correspond 1-1 to the paths of length m of the transition system defined by the $\mathcal{C}+$ action description D. In particular, models of $comp(\Gamma_1^D)$ encode the transitions defined by D and models of $comp(\Gamma_0^D)$ the states defined by D.

Given an action description D and a non-negative integer m, the 'Causal Calculator' CCalc performs the translation to the causal theory Γ_m^D, constructs $comp(\Gamma_m^D)$, and then invokes a standard propositional satisfaction solver to find the (classical) models of $comp(\Gamma_m^D)$. So, for example, plans of length m from an initial state satisfying fluent formula F to a goal state

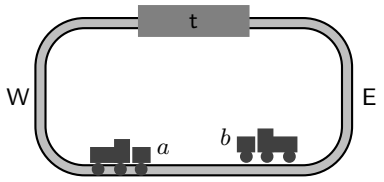
satisfying fluent formula G can be found by determining the models of the (classical) propositional formula $comp(\Gamma^D_m) \wedge F[0] \wedge G[m]$.

It must be emphasised, however, that $\mathcal{C}+$ is a language for defining labelled transition systems (of a certain kind), and is not restricted to use with CCalc. A variety of other languages can be interpreted on the transition system defined by a $\mathcal{C}+$ action description. In particular, in later sections we will look at the use of the branching time temporal logic CTL for expressing system properties to be checked on transition systems defined by $\mathcal{C}+$.

Example (trains)

The following example is used in [12, 13] to illustrate the use of alternating-time logic (ATL) for determining the effectiveness of 'social laws' designed to co-ordinate the actions of agents in multi-agent system. We will use the example for a different purpose: in this section, to illustrate use of the language $\mathcal{C}+$, and in later sections, to show how the extended form $\mathcal{C}+^{++}$can be used to analyse variants of the example in which agents may fail to obey social laws.

There are two trains, a and b, with a running clockwise round a double track, and b running anti-clockwise. There is a tunnel in which the double track becomes a single track. If the trains are both inside the tunnel at the same time they will collide. The tunnel can thus be seen as a kind of critical section, or as a resource for which the trains must compete.



There are obviously many ways in which the example can be formulated. The following will suffice for present purposes. Although it may seem unnecessarily complicated, this formulation is convenient for the more elaborate versions of the example to come later.

Let fluent constants $loc\,(a)$ and $loc\,(b)$ represent the locations of trains a and b respectively. They both have possible values $\{\mathsf{W}, \mathsf{t}, \mathsf{E}\}$. For action constants, we take a and b with possible values $\{go, stay\}$. (Action constants $act(a)$ and $act(b)$ may be easier to read but we choose a and b for brevity.)

The $\mathcal{C}+$ action description representing the possible movements of the trains is as follows. We will call this action description $\mathsf{D}_{\mathrm{trains}}$.

inertial $loc\,(a)$, $loc\,(b)$

train a moves clockwise:

a=go causes $loc\,(a)$=t if $loc\,(a)$=W
a=go causes $loc\,(a)$=E if $loc\,(a)$=t
a=go causes $loc\,(a)$=W if $loc\,(a)$=E

train b moves anti-clockwise:

$b{=}go$ causes $loc\,(b){=}\mathsf{t}$ if $loc\,(b){=}\mathsf{E}$
$b{=}go$ causes $loc\,(b){=}\mathsf{W}$ if $loc\,(b){=}\mathsf{t}$
$b{=}go$ causes $loc\,(b){=}\mathsf{E}$ if $loc\,(b){=}\mathsf{W}$

collisions:

$collision$ iff $loc\,(a){=}\mathsf{t} \wedge loc\,(b){=}\mathsf{t}$ % for convenience
nonexecutable $a{=}go$ if $collision$
nonexecutable $b{=}go$ if $collision$

The Boolean fluent constant *collision* is introduced for convenience.[2] The example can be formulated perfectly well without it.

The transition system defined by $\mathsf{D}_{\mathrm{trains}}$ is shown in Fig. 1.

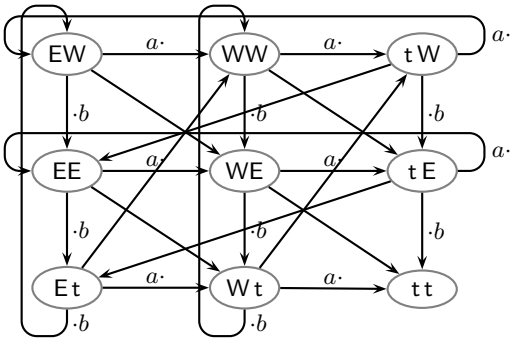


Fig. 1. The transition system for the trains example. A state label such as EW is short for $\{loc\,(a){=}\mathsf{E}, loc\,(b){=}\mathsf{W}\}$. Horizontal edges, labelled $a\cdot$, are transitions in which train a moves and b does not. Vertical edges, labelled $\cdot b$, are transitions in which train b moves and a does not. Diagonal edges, unlabelled in the diagram, are transitions in which both trains move. Reflexive edges, corresponding to transitions in which neither train moves, are omitted from the diagram to reduce clutter.

3 The language $\mathcal{C}+^{++}$

$\mathcal{C}+^{++}$ is an extended form of the language $\mathcal{C}+$ designed for representing norms of behaviour and institutional aspects of (human or computer) societies [9]. It provides two main extensions to $\mathcal{C}+$. The first is a means of expressing 'counts as' relations between actions, also referred to as 'conventional generation' of

[2] For readers familiar with $\mathcal{C}+$, a law of the form F iff G is used here as shorthand for the pair of laws F if G and default $\neg F$. default $\neg F$ is a $\mathcal{C}+$ abbreviation for the law $\neg F$ if $\neg F$.

actions. That will not be discussed further in this paper. The second extension is a way of specifying the permitted (acceptable, legal) states of a transition system and its permitted (acceptable, legal) transitions.

Syntax and semantics An action description of $\mathcal{C}+^{++}$ defines a *coloured transition system*, which is a structure of the form: $\langle S, \mathbf{A}, R, S_{\mathrm{g}}, R_{\mathrm{g}}\rangle$ where $\langle S, \mathbf{A}, R\rangle$ is a labelled transition system of the kind defined by $\mathcal{C}+$ action descriptions, and where the two new components are

- $S_{\mathrm{g}} \subseteq S$, the set of 'permitted' ('acceptable', 'ideal', 'legal') states—we call S_{g} the 'green' states of the system;
- $R_{\mathrm{g}} \subseteq R$, the set of 'permitted' ('acceptable', 'ideal', 'legal') transitions—we call R_{g} the 'green' transitions of the system.

We refer to the complements $S - S_{\mathrm{g}}$ and $R - R_{\mathrm{g}}$ as the 'red states' and 'red transitions', respectively. Semantical devices which partition states (and here, transitions) into two categories are familiar in the field of deontic logic where are known to yield rather simplistic logics; full discussion of their adequacy is outside the scope of this paper. It is also possible to consider a more elaborate structure, of *partially coloured* transition systems in which states and transitions can be green, red, or uncoloured, but we shall not present that version here.

A coloured transition system $\langle S, \mathbf{A}, R, S_{\mathrm{g}}, R_{\mathrm{g}}\rangle$ must further satisfy the following constraint:

- if $(s, \epsilon, s') \in R_{\mathrm{g}}$ and $s \in S_{\mathrm{g}}$ then $s' \in S_{\mathrm{g}}$.

We refer to this as the *green-green-green* constraint. The idea is that occurrence of a permitted (green) transition in a permitted (green) state must always lead to a permitted (green) state. All other possible combinations of green/red states and green/red transitions are allowed. In particular, and *contra* the assumptions underpinning John-Jules Meyer's construction of 'dynamic deontic logic' [14], a non-permitted (red) transition can result in a permitted (green) state. Similarly, it is easy to devise examples in which a permitted (green) transition can lead to a non-permitted (red) state. Some illustrations will arise in the examples to be considered later. The only combination that cannot occur is the one eliminated by the 'green-green-green' constraint: a permitted (green) transition from a permitted (green) state cannot lead to a non-permitted (red) state.

The language $\mathcal{C}+^{++}$ extends the language $\mathcal{C}+$ with two new forms of rules. A *state permission law* is an expression of the form

$$\textsf{not-permitted } F \tag{4}$$

where F is a fluent formula. An *action permission law* is an expression of the form

$$\textsf{not-permitted } \alpha \textsf{ if } \psi \tag{5}$$

where α is an action formula and ψ is any formula of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$. not-permitted α is an abbreviation for not-permitted α if $\top$. It is also convenient to allow two variants of rule forms (4) and (5), allowing oblig F as an abbreviation for not-permitted $\neg F$ and oblig α as an abbreviation for not-permitted $\neg\alpha$.

Informally, in the transition system defined by an action description D, a state s is red whenever $s \models F$ for any state permission law not-permitted F. All other states are green by default. A transition (s, ϵ, s') is red whenever $s \cup \epsilon \models \psi$ and $\epsilon \models \alpha$ for any action permission law not-permitted α if F after ψ. All other transitions are green, *subject to* the 'green-green-green' constraint which may impose further conditions on the possible colouring of a given transition.

Let D be an action description of $\mathcal{C}+^{++}$. D_{basic} refers to the subset of laws of D that are also laws of $\mathcal{C}+$. The transition system defined by D has the states S and transitions R that are defined by its $\mathcal{C}+$ component, D_{basic}, and green states S_{g} and green transitions R_{g} given by $S_{\mathrm{g}} =_{\mathrm{def}} S - S_{\mathrm{red}}$, $R_{\mathrm{g}} =_{\mathrm{def}} R - R_{\mathrm{red}}$ where

$$S_{\mathrm{red}} =_{\mathrm{def}} \{s \mid s \models F \text{ for some law not-permitted } F \text{ in } D\}$$

$$R_{\mathrm{red}} =_{\mathrm{def}} \{(s, \epsilon, s') \mid s \cup \epsilon \models \psi,\ \epsilon \models \alpha \text{ for some law not-permitted } \alpha \text{ if } \psi \text{ in } D\} \cup \{(s, \epsilon, s') \mid s \in S_{\mathrm{g}} \text{ and } s' \notin S_{\mathrm{g}}\}$$

The second component of the R_{red} definition ensures that the 'green-green-green' constraint is satisfied.

Example Consider the trains example of section 2. A collision is undesirable, unacceptable, not permitted ('red'). Construct an action description D_1 of $\mathcal{C}+^{++}$ by adding to the $\mathcal{C}+$ action description $\mathsf{D}_{\mathrm{trains}}$ of section 2 the state permission law

$$\textsf{not-permitted } \textit{collision} \tag{6}$$

The coloured transition system defined by D_1 is the transition system of Fig. 1 with the collision state tt coloured red and all other states coloured green. The three transitions leading to the collision state are coloured red because of the green-green-green constraint; all other transistions, including the transition from the collision state to itself, are green.

Causal theories Any (definite) action description of $\mathcal{C}+^{++}$ can be translated to the language of (definite) causal theories, as follows. Let D be an action description and m a non-negative integer. The translation of the $\mathcal{C}+$ component D_{basic} of D proceeds as usual. For the permission laws, introduce two new fluent and action constants, status and trans respectively, both with possible values green and red. They will be used to represent the colour of a state and the colour of a transition, respectively.

For every state permission law not-permitted F and time index $i \in 0..m$, include in Γ^D_m a causal rule of the form status$[i]$=red $\Leftarrow F[i]$, and for every $i \in 0..m$, a causal rule of the form status$[i]$=green $\Leftarrow$ status$[i]$=green to specify the default colour of a state. A state permission rule of the form oblig F produces causal rules of the form status$[i]$=red $\Leftarrow \neg F[i]$.

For every action permission law $\mathsf{not\text{-}permitted}\ \alpha\ \mathsf{if}\ \psi$ and time index $i \in 0\,..\,m{-}1$, include in Γ^D_m a causal rule of the form $\mathsf{trans}[i]{=}\mathsf{red} \Leftarrow \alpha[i] \wedge \psi[i]$, and for every $i \in 0\,..\,m{-}1$, a causal rule of the form $\mathsf{trans}[i]{=}\mathsf{green} \Leftarrow \mathsf{trans}[i]{=}\mathsf{green}$ to specify the default colour of a transition. An action permission law of the form $\mathsf{oblig}\ \alpha\ \mathsf{if}\ \psi$ produces causal rules of the form $\mathsf{trans}[i]{=}\mathsf{red} \Leftarrow \neg\alpha[i] \wedge \psi[i]$.

Finally, to capture the 'green-green-green' constraint, include for every $i \in 0\,..\,m{-}1$ a causal rule of the form

$$\mathsf{trans}[i]{=}\mathsf{red} \ \Leftarrow\ \mathsf{status}[i]{=}\mathsf{green} \wedge \mathsf{status}[i{+}1]{=}\mathsf{red} \tag{7}$$

It is straightforward to show [9] that models of the causal theory Γ^D_m correspond to all paths of length m through the coloured transition system defined by D, where the fluent constant status and the action constant trans encode the colours of the states and transitions, respectively.

Notice that, although action descriptions in $\mathcal{C}{+}^{++}$ can be translated to causal theories, they cannot be translated to action descriptions of $\mathcal{C}+$: there is no form of causal law in $\mathcal{C}+$ which translates to the green-green-green constraint (7).

In addition to permission laws of the form (4) and (5), which are convenient but rather restrictive, the $\mathcal{C}{+}^{++}$ language allows distinguished fluent and action constants status and trans to be used explicitly in formulas and causal laws. The atoms $\mathsf{status}{=}\mathsf{red}$ and $\mathsf{trans}{=}\mathsf{red}$ can then be regarded as what are sometimes called 'violation constants' in deontic logic. It is also easy to allow more 'shades' of red and green to allow different notions of permitted/legal/acceptable to be mixed. We will not employ that device in the examples discussed in this paper.

Example (trains, continued)

The action description D_1 of the previous section states that collisions are not permitted but says nothing about how the trains should ensure that collisions are avoided. Suppose, for the sake of an example, that we impose additional norms (social laws), as follows: no train is permitted to enter the tunnel unless the other train has just emerged. (We assume that this will be observed by the train that is preparing to enter.) Will such a law be effective in avoiding collisions?

To construct a $\mathcal{C}{+}^{++}$ action description D_2 for this version, ignore (6) and instead add to the $\mathcal{C}+$ action description $\mathsf{D}_{\mathrm{trains}}$ the following laws. First, it is convenenient to define the following auxiliary action constants (all Boolean):

$$\begin{array}{l} \mathit{enter}\,(a)\ \mathsf{iff}\ \ a{=}\mathit{go} \wedge \mathit{loc}\,(a){=}\mathsf{W} \\ \mathit{exit}\,(a)\ \mathsf{iff}\ \ a{=}\mathit{go} \wedge \mathit{loc}\,(a){=}\mathsf{t} \\ \mathit{enter}\,(b)\ \mathsf{iff}\ \ b{=}\mathit{go} \wedge \mathit{loc}\,(b){=}\mathsf{E} \\ \mathit{exit}\,(b)\ \mathsf{iff}\ \ b{=}\mathit{go} \wedge \mathit{loc}\,(b){=}\mathsf{t} \end{array} \tag{8}$$

Again, these are introduced merely for convenience; the example can be constructed easily enough without them. Now we formulate the social laws:

$$\begin{array}{l}\mathsf{not\text{-}permitted}\ \mathit{enter}\,(a)\ \mathsf{if}\ \mathit{loc}\,(b){\neq}\mathsf{W}\\ \mathsf{not\text{-}permitted}\ \mathit{enter}\,(b)\ \mathsf{if}\ \mathit{loc}\,(a){\neq}\mathsf{E}\end{array} \tag{9}$$

The coloured transition system for this version of the example is shown in Fig. 2. Notice that since we are now using green/red to represent what is permitted/not permitted from the point of view of train behaviour, we have discarded the state permission law (6). Consequently the collision state tt is coloured green not red. We could combine the two notions of permission expressed by laws (6) and (9), for instance by introducing two different shades of green and red and relating them to each other, but we do not have space to discuss that option here.

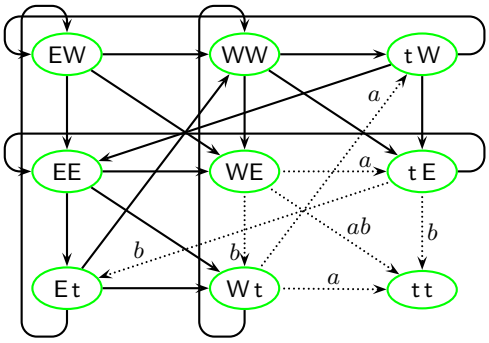


Fig. 2. Coloured transition system defined by action description D_2. Dotted lines indicate red transitions. All states and all other transitions are green. Reflexive edges (all green) are omitted for clarity.

How do we test the effectiveness of the social laws (9)? Since the causal theory $\Gamma_1^{\mathsf{D}_2}$ encodes the transitions defined by D_2, the following captures the property that if both trains comply with the social laws, no collisions will occur.

$$\mathit{comp}(\Gamma_1^{\mathsf{D}_2}) \models \neg \mathit{collision}[0] \wedge \mathsf{trans}[0]{=}\mathsf{green} \rightarrow \neg \mathit{collision}[1] \tag{10}$$

This can be checked, as in CCALC, by using a standard sat-solver to determine that the formula $\mathit{comp}(\Gamma_1^{\mathsf{D}_2}) \wedge \neg \mathit{collision}[0] \wedge \mathsf{trans}[0]{=}\mathsf{green} \wedge \mathit{collision}[1]$ is not satisfiable. The property (10) is equivalently expressed as:

$$\mathit{comp}(\Gamma_1^{\mathsf{D}_2}) \models \neg \mathit{collision}[0] \wedge \mathit{collision}[1] \rightarrow \mathsf{trans}[0]{=}\mathsf{red} \tag{11}$$

which says that a collision occurs only following a transition in which either one train or both violate the norms.

Notice that $comp(\Gamma_1^{\mathsf{D}_2}) \not\models \mathsf{trans}[0]{=}\mathsf{green} \rightarrow \neg collision[1]$: as formulated by D_2, the transition from a collision state to itself is green.

One major advantage of taking $\mathcal{C}+$ as the basic action formalism, as we see it, is its explicit transition system semantics, which enables a wide range of other analytical techniques to be applied. In particular, system properties can be expressed in the branching time temporal logic CTL and verified on the transition system defined by a $\mathcal{C}+$ or $\mathcal{C}+^{++}$ action description using standard model checking systems.

We will say that a formula φ of CTL is *valid* on a (coloured) transition system $\langle S, \mathrm{I}(\sigma^{\mathrm{a}}), R, S_{\mathrm{g}}, R_{\mathrm{g}}\rangle$ defined by $\mathcal{C}+^{++}$ action description D when $s \cup \epsilon \models \varphi$ for every $s \cup \epsilon$ such that $(s, \epsilon, s') \in R$ for some state s'. The definition is quite standard, except for a small adjustment to allow action constants in φ to continue to be evaluated on transition labels ϵ. (And we do not distinguish any particular set of initial states; all sets in S are initial states.) We will also say in that case that formula φ is valid on the action description D.

In CTL, the formula $\mathsf{AX}\,\varphi$ expresses that φ is satisfied in the next state in all future branching paths from now.[3] EX is the dual of AX: $\mathsf{EX}\,\varphi \equiv \neg\mathsf{AX}\,\neg\varphi$. $\mathsf{EX}\,\varphi$ expresses that φ is satisfied in the next state of some future branching path from now. The properties (10) and (11) can thus be expressed in CTL as follows:

$$\neg collision \wedge \mathsf{trans}{=}\mathsf{green} \rightarrow \mathsf{AX}\,\neg collision \tag{12}$$

or equivalently $\neg collision \wedge \mathsf{EX}\, collision \rightarrow \mathsf{trans}{=}\mathsf{red}$. It is easily verified by reference to Fig. 2 that these formulas are valid on the action description D_2. Also valid is the CTL formula $\mathsf{EX}\,\mathsf{trans}{=}\mathsf{green}$ which expresses that there is always a permitted action for both trains. This is true even in collision states, since the only available transition is then the one where both trains remain idle, and that transition is green. The CTL formula $\mathsf{EF}\, collision$ is also valid on D_2, signifying that in every state there is at least one path from then on with *collision* true somewhere in the future.[4]

4 Example: a simple co-ordination mechanism

We now consider a slightly more elaborate version of the trains example. In general, we want to be able to verify formally whether the introduction of additional control mechanisms—additional controller agents, communication devices, restrictions on agents' possible actions—are effective in ensuring that agents comply with the norms ('social laws') that govern their behaviour. For the trains, we might consider a controller of some kind, or traffic lights, or some mechanism by which the trains communicate their locations to one another. For the sake of an example, we will suppose that there is a physical

[3] $s_0 \cup \epsilon_0 \models \mathsf{AX}\,\varphi$ if for every infinite path $s_0\,\epsilon_0\,s_1\,\epsilon_1\,\cdots$ we have that $s_1 \cup \epsilon_1 \models \varphi$.

[4] $s_0 \cup \epsilon_0 \models \mathsf{EF}\,\varphi$ if there is an (infinite) path $s_0\,\epsilon_0\,\cdots\,s_m\,\epsilon_m\,\cdots$ with $s_m \cup \epsilon_m \models \varphi$ for some $m \geq 0$.

token (a metal ring, say) which has to be collected before a train can enter the tunnel. A train must pick up the token before entering the tunnel, and it must deposit it outside the tunnel as it exits. No train may enter the tunnel without possession of the token.

To construct the $\mathcal{C}+^{++}$ action description D_3 for this version of the example, we begin as usual with the $\mathcal{C}+$ action description $\mathsf{D}_{\text{trains}}$ of section 2. We add a fluent constant *tok* to represent the position of the token. It has values $\{\mathsf{W}, \mathsf{E}, a, b\}$. $tok{=}\mathsf{W}$ represents that the token is lying at the West end of the tunnel, $tok{=}a$ that the token is currently held by train a, and so on. We add Boolean action constants $pick\,(a)$, $pick\,(b)$ to represent that a (resp., b) picks up the token, and $drop\,(a)$, $drop\,(b)$ to represent that a (resp., b) drops the token at its current location. For convenience, we will keep the action constants $enter\,(a)$, $enter\,(b)$, $exit\,(a)$, $exit\,(b)$ defined as in D_2 of the previous section.

The following causal laws describe the effects of picking up and dropping the token. To avoid duplication, x and l are variables ranging over a and b and locations W, E, t respectively.

$$\begin{array}{l}\textsf{inertial } tok\\[1ex] tok{=}l \textsf{ if } loc\,(x){=}l \textsf{ after } drop\,(x)\\ \textsf{nonexecutable } drop\,(x) \textsf{ if } tok{\neq}x\\[1ex] pick\,(x) \textsf{ causes } tok{=}x\\ \textsf{nonexecutable } pick\,(x) \textsf{ if } loc\,(x){\neq}tok\end{array}$$

The above specifies that the token can be dropped by train x only if train x has the token ($tok{=}x$), and it can be picked up by train x only if train x and the token are currently at the same location ($loc\,(x){=}tok$). Notice that, as defined, an action $drop\,(x) \wedge x{=}stay$ drops the token at the current location of train x, and $drop\,(x) \wedge x{=}go$ drops it at the location of train x *after* it has moved. (The fluent dynamic law $drop\,(x)$ causes $tok{=}l$ if $tok{=}x \wedge loc\,(x){=}l$ would drop the token at the location of train x before it moved.) Since $tok{=}\mathsf{t}$ is not a well-formed atom, it is not possible that (there is no transition in which) the token is dropped inside the tunnel. $pick\,(x) \wedge x{=}go$ represents an action in which train x picks up the token and moves with it. More refined representations could of course be constructed but this simple version will suffice for present purposes.

The action description D_3 is completed by adding the following permission laws:

$$\begin{array}{l}\textsf{not-permitted } enter\,(x) \textsf{ if } \;\; tok{\neq}x \wedge \neg pick\,(x)\\ \textsf{oblig } drop\,(x) \textsf{ if } \;\; exit\,(x) \wedge tok{=}x\end{array} \tag{13}$$

It may be helpful to note that in $\mathcal{C}+^{++}$, the first of these laws is equivalent to

$$\textsf{oblig } pick\,(x) \textsf{ if } \;\; enter\,(x) \wedge tok{\neq}x$$

The coloured transition system defined by action description D_3 is larger and more complicated than that for D_2 of the previous section, and cannot be drawn easily in its entirety. A fragment is shown in Fig. 3.

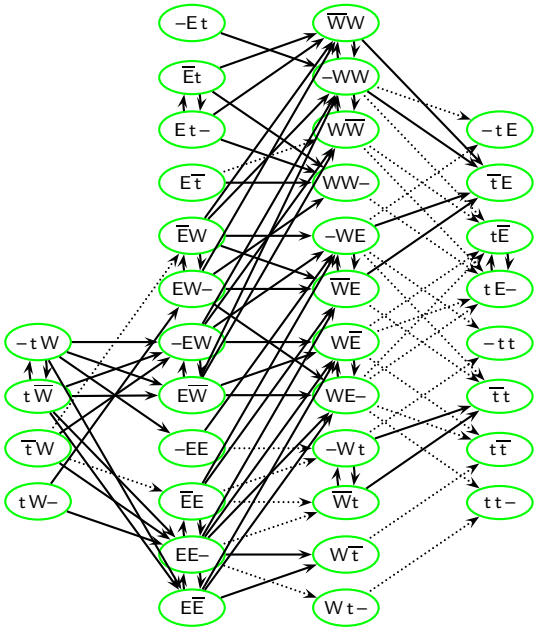


Fig. 3. Fragment of the coloured transition system defined by D_3. The figure shows all states but not all transitions. The dash in state labels indicates the position of the token: it is at W/E when the dash is on the left/right, and with train a/b when the dash appears above the location of a/b. Dotted lines depict red transitions. All other depicted transitions, and all states, are green.

One property we might wish to verify on D_3 is that collisions are guaranteed to be avoided if both trains comply with the norms ('social laws'). Using the 'Causal Calculator' CCalc, we can try to determine whether

$$comp(\Gamma^{\mathsf{D}_3}_m) \;\models\; \neg collision[0] \wedge \mathsf{trans}[0]{=}\mathsf{green} \wedge \ldots \wedge \mathsf{trans}[m{-}1]{=}\mathsf{green} \rightarrow \neg collision[m]$$

that is, whether the formula $comp(\Gamma^D_m) \wedge \neg collision[0] \wedge \mathsf{trans}[0]{=}\mathsf{green} \wedge \cdots \wedge \mathsf{trans}[m{-}1]{=}\mathsf{green} \wedge collision[m]$ is satisfiable. But what should we take as the length m of the longest path to be considered? In some circumstances it is possible to find a suitable value m for the longest path to be considered but it is far from obvious in this example what that value is, or even if one exists.

The problem can be formulated conveniently as a model checking problem in CTL. The CTL formula $\mathsf{E}[\mathsf{trans}{=}\mathsf{green}\,\mathsf{U}\,collision]$ expresses that there is at least one path with *collision* true at some future state and $\mathsf{trans}{=}\mathsf{green}$

true on all intervening transitions.[5] So the property we want to verify can be expressed in CTL as follows:

$$\neg \mathit{collision} \rightarrow \neg \mathsf{E}[\mathsf{trans}{=}\mathsf{green}\,\mathsf{U}\,\mathit{collision}] \tag{14}$$

It can be seen from Fig. 3 that property (14) is not valid on the action description D_3: there are green transitions leading to collision states, from states where there is already a train inside the tunnel without the token. However, as long as we consider states in which both trains are initially outside the tunnel, the safety property we seek can be verified. The following formula is valid on D_3:

$$\mathit{loc}\,(a){\neq}\mathsf{t} \wedge \mathit{loc}\,(b){\neq}\mathsf{t} \rightarrow \neg \mathsf{E}[\mathsf{trans}{=}\mathsf{green}\,\mathsf{U}\,\mathit{collision}] \tag{15}$$

We are often interested in the verification of formulas such as (14) and (15) which express system properties conditional on norm compliance (conditional on all transitions being green). Verfication of such properties is particularly easy: translate the coloured transition system $\mathcal{M} = \langle S, \mathbf{A}, R, S_{\mathrm{g}}, R_{\mathrm{g}}\rangle$ to the transition system $\mathcal{M}' = \langle S_{\mathrm{g}}, \mathbf{A}, R_{\mathrm{g}}\rangle$ obtained by deleting all red states and red transitions from $\mathcal{M}$. Now, since in CTL $\mathsf{E}[\top\,\mathsf{U}\,\varphi] \equiv \mathsf{EF}\,\varphi$, instead of checking, for example, formula (15) on $\mathcal{M}$ we can check whether

$$\mathit{loc}\,(a){\neq}\mathsf{t} \wedge \mathit{loc}\,(b){\neq}\mathsf{t} \rightarrow \neg \mathsf{EF}\,\mathit{collision} \tag{16}$$

is valid on $\mathcal{M}'$. This is a standard model checking problem.

5 Conclusion

We have presented the permission component of the action language $\mathcal{C}{+}^{++}$ and sketched how it can be applied to modelling systems in which agents do not necessarily comply with the norms ('social laws') that govern their behaviour. Space limitations prevented us from discussing more elaborate examples where non-compliance with one set of norms imposes further norms for reparation and/or recovery. We are currently working on the use of $\mathcal{C}{+}^{++}$ as the input language for various temporal logic model checkers, CTL in particular. Scaleability is of course an issue; however, here the limits are set by the model checking techniques employed and not by the use of $\mathcal{C}{+}^{++}$. At the MSRAS workshop, our attention was drawn to the model checking technique of [15] which uses program transformations on constraint logic programs representing transition systems to verify formulas of CTL. Since action descriptions in $\mathcal{C}{+}$, and in $\mathcal{C}{+}^{++}$, can be related via causal theories to logic programs [10], this presents an interesting avenue to explore.

[5] $s_0 \cup \epsilon_0 \models \mathsf{E}[\varphi_1\,\mathsf{U}\,\varphi_2]$ if there is an (infinite) path $s_0\,\epsilon_0 \cdots s_m\,\epsilon_m \cdots$ with $s_m \cup \epsilon_m \models \varphi_2$ for some $m \geq 0$ and with $s_i \cup \epsilon_i \models \varphi_1$ for all $0 \leq i < m$.

References

1. Subrahmanian, V.S., Bonatti, P., Dix, J., Eiter, T., Kraus, S., Ozcan, F., Ross, R.: Heterogeneous Agent Systems. MIT Press, Cambridge (2000)
2. Rissanen, E., Sadighi Firozabadi, B., Sergot, M.J.: Towards a mechanism for discretionary overriding of access control (position paper). In: Proc. 12th International Workshop on Security Protocols, Cambridge, April 2004. (2004)
3. Artikis, A., Pitt, J., Sergot, M.J.: Animated specification of computational societies. In Castelfranchi, C., Johnson, W.L., eds.: Proc. 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02), Bologna, ACM Press (2002) 1053–1062
4. Artikis, A., Sergot, M.J., Pitt, J.: Specifying electronic societies with the Causal Calculator. In Giunchiglia, F., Odell, J., Weiss, G., eds.: Agent-Oriented Software Engineering III. Proc. 3rd International Workshop (AOSE 2002), Bologna, July 2002. LNCS 2585, Springer (2003) 1–15
5. Artikis, A., Sergot, M.J., Pitt, J.: An executable specification of an argumentation protocol. In: Proc. 9th International Conference on Artificial Intelligence and Law (ICAIL'03), Edinburgh, ACM Press (2003) 1–11
6. Lomuscio, A., Sergot, M.J.: Deontic interpreted systems. Studia Logica **75** (2003) 63–92
7. Lomuscio, A., Sergot, M.J.: A formalisation of violation, error recovery, and enforcement in the bit transmission problem. Journal of Applied Logic **2** (2004) 93–116
8. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
9. Sergot, M.: The language $\mathcal{C}+^{++}$. In Pitt, J., ed.: The Open Agent Society. Wiley (2004) (In press). Extended version: Technical Report 2004/8. Department of Computing, Imperial College, London.
10. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. Artificial Intelligence **153** (2004) 49–104
11. Giunchiglia, E., Lifschitz, V.: An action language based on causal explanation: Preliminary report. In: Proc. AAAI-98, AAAI Press (1998) 623–630
12. van der Hoek, W., Roberts, M., Wooldridge, M.: Social laws in alternating time: Effectiveness, feasibility, and synthesis. Technical report, Dept. of Computer Science, University of Liverpool (2004) Submitted.
13. Jamroga, W., van der Hoek, W., Wooldridge, M.: On obligations and abilities. In Lomuscio, A., Nute, D., eds.: Proc. 7th International Workshop on Deontic Logic in Computer Science (DEON'04), Madeira, May 2004. LNAI 3065, Springer (2004) 165–181
14. Meyer, J.J.: A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. Notre Dame Journal of Formal Logic **29** (1988) 109–136
15. Fioravanti, F., Pettorossi, A., Proietti, M.: Verifying CTL properties of infinite state systems by specializing constraint logic programs. In: Proceedings of Second ACM-Sigplan International Workshop on Verification and Computational Logic (VCL'01), Florence, September 2001. (2001) 85–96 Expanded version: Technical Report R.544, IASI-CNR, Rome.