# The Deontic Component of Action Language $n\mathcal{C}+$

Marek Sergot and Robert Craven

Department of Computing, Imperial College London
{mjs, rac101}@doc.ic.ac.uk

**Abstract.** The action language $\mathcal{C}+$ of Giunchiglia, Lee, Lifschitz, McCain, and Turner is a formalism for specifying and reasoning about the effects of actions and the persistence ('inertia') of facts over time. An 'action description' in $\mathcal{C}+$ defines a labelled transition system of a certain kind. $n\mathcal{C}+$ (formerly known as $(\mathcal{C}+)^{++}$) is an extended form of $\mathcal{C}+$ designed for representing normative and institutional aspects of (human or computer) societies. The deontic component of $n\mathcal{C}+$ provides a means of specifying the permitted (acceptable, legal) states of a transition system and its permitted (acceptable, legal) transitions. We present this component of $n\mathcal{C}+$, motivating its details with reference to some small illustrative examples.

## 1 Introduction

The action language $\mathcal{C}+$ [1] is a formalism for specifying and reasoning about the effects of actions and the persistence ('inertia') of facts over time, building on a general purpose non-monotonic representation formalism called 'causal theories'. An 'action description' in $\mathcal{C}+$ is a set of $\mathcal{C}+$ rules which define a labelled transition system of a certain kind. Implementations supporting a wide range of querying and planning tasks are available, notably in the form of the 'Causal Calculator' CCALC [2]. $\mathcal{C}+$ and CCALC have been applied successfully to a number of benchmark examples in the knowledge representation literature (see e.g. [3] and the CCALC website [2]). We have used it in our own work to construct executable specifications of agent societies (see e.g. [4, 5]).

$n\mathcal{C}+$ [6, 7] is an extended form of $\mathcal{C}+$ designed for representing normative and institutional aspects of (human or computer) societies. There are 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. The aim of the paper is to present this component of $n\mathcal{C}+$ and some simple illustrative examples. $n\mathcal{C}+$ was called $(\mathcal{C}+)^{++}$ in earlier presentations.

$n\mathcal{C}+$ is intended for modelling system behaviour from an external 'bird's eye' perspective, that is to say, from the system designer's point of view. It may then be verified whether properties hold or not of the system specified (a process

L. Goble and J.-J.C. Meyer (Eds.): DEON 2006, LNAI 4048, pp. 222–237, 2006.

analogous to that described in [8, 9], which concentrates on epistemic properties and communicative acts). $n\mathcal{C}+$ is not intended for representing norms from an individual agent's perspective. We have a separate development, *agent-centric* $n\mathcal{C}+$, for specifying system norms as directives that constrain an individual agent's behaviour, in a form that can be used by a (computer) agent in its internal decision-making procedures. That development will be presented elsewhere.

We have three existing implementations of the $n\mathcal{C}+$ language. The first employs the 'Causal Calculator' CCalc. As explained later in the paper, the required modifications to CCalc are minor and very easily implemented. The second implementation provides an 'event calculus' style of computation with $\mathcal{C}+$ and $n\mathcal{C}+$ action descriptions. Given an action description and a 'narrative'—a record of what events have occurred—this implementation allows all past states, including what was permitted and obligatory at each past state, to be queried and computed. The third implementation connects $\mathcal{C}+$ and $n\mathcal{C}+$ to model checking software. System properties expressed in temporal logics such as CTL can then be verified by means of standard model checking techniques (specifically the model checker NuSMV) on transition systems defined using the $n\mathcal{C}+$ language. A small example is presented in [7]. We do not discuss the implementations further for lack of space, except to explain how the CCalc method works.

*Related work.* Some readers may see a resemblance between $n\mathcal{C}+$ and John-Jules Meyer's Dynamic Deontic Logic [10], and other well known works based on 'modal action logics' generally (e.g. [11, 12]). There are three fundamental differences. (1) $\mathcal{C}+$ and $n\mathcal{C}+$ are not variants of dynamic logic or modal action logic. They are languages for defining specific instances of labelled transitions systems. Other languages—we refer to them as 'query languages'—can then be interpreted on these structures. Dynamic logic is one candidate, the query language in CCalc is another, but there are many other possibilities: each $\mathcal{C}+$ or $n\mathcal{C}+$ action description defines a Kripke-structure, on which a variety of (modal) query languages, including a wide range of deontic and temporal operators, can be evaluated. We do not have space to discuss any of these possibilities in detail. (2) The representation of action is quite different from that in dynamic logic and modal action logic. (3) There are important differences of detail, in particular concerning the interactions between permitted states and permitted transitions between states.

The semantical devices employed in $n\mathcal{C}+$—classification of states and transitions into green/red (good/bad, ideal/sub-ideal), violation constants, explicit names for norms, and orderings of states according to how well they comply with these norms—are all frequently encountered in the deontic logic literature. The novelty here lies, first, in the details of how they are incorporated into labelled transition systems, and second, in the way the $n\mathcal{C}+$ language is used to define these structures.

Finally, $\mathcal{C}+$ is a (recent) member of a family of formalisms called 'causal action languages' in the AI literature. Several groups have suggested encoding normative concepts in such formalisms. We have done so ourselves in other work (see e.g. [13, 4, 5]) where we have used both $\mathcal{C}+$ and the 'event calculus' for

this purpose. Leon van der Torre [14] has made a suggestion along similar lines, though using a different causal action language and a different approach. See also the discussion in [12]. One feature that distinguishes $\mathcal{C}+$ from other AI action languages is that it has an explicit semantics in terms of transition systems. It thereby proves a bridge between AI formalisms and standard methods in other areas of computer science and logic. It is this feature that $n\mathcal{C}+$ seeks to exploit.

## 2 The Language $\mathcal{C}+$

We begin with a concise, and necessarily rather dense, summary of the $\mathcal{C}+$ language. Some features (notably 'statically determined fluents' and 'exogenous actions') are omitted for simplicity. There are also some minor syntactic and terminological differences from the version presented in [1]. See [6] for details.

A *multi-valued propositional signature* $\sigma$ is a set of symbols called *constants*. For each constant $c$ in $\sigma$ there is a non-empty set $dom(c)$ of values called the *domain* of $c$. For simplicity, in this paper we will assume that each $dom(c)$ is finite and has at least two elements. An *atom* of a signature $\sigma$ is an expression of the form $c = v$ where $c$ is a constant in $\sigma$ and $v \in dom(c)$. A *formula* $\varphi$ of signature $\sigma$ is any propositional compound of atoms of $\sigma$. The expressions $\top$ and $\bot$ are 0-ary connectives, with the usual interpretation.

A *Boolean* constant is one whose domain is the set of truth values $\{\mathsf{t}, \mathsf{f}\}$. If $p$ is a Boolean constant, $p$ is shorthand for the atom $p = \mathsf{t}$ and $\neg p$ for the atom $p = \mathsf{f}$. Notice that, as defined here, $\neg p$ is an *atom* when $p$ is a Boolean constant.

In $\mathcal{C}+$, the signature $\sigma$ is partitioned into a set $\sigma^{\mathrm{f}}$ of *fluent constants* (also known as 'state variables' in other areas of Computer Science) and a set $\sigma^{\mathrm{a}}$ of *action constants*. A *fluent formula* is a formula whose constants all belong to $\sigma^{\mathrm{f}}$; an *action formula* is a formula containing at least one action constant and no fluent constants.

An *interpretation* of a multi-valued signature $\sigma$ is a function that maps every constant $c$ in $\sigma$ to some value $v$ in $dom(c)$; an interpretation $I$ *satisfies* an atom $c = v$, written $I \models c = v$, if $I(c) = v$. The satisfaction relation $\models$ is extended from atoms to formulas in accordance with the standard truth tables for the propositional connectives. We write $\mathrm{I}(\sigma)$ for the set of interpretations of $\sigma$.

*Transition systems.* Every $\mathcal{C}+$ action description $D$ of signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$ 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 $\sigma^{\mathrm{f}}$ of $D$; $S \subseteq \mathrm{I}(\sigma^{\mathrm{f}})$;
- $\mathbf{A}$ is a set of *transition labels*, also called *events*; $\mathbf{A}$ is the set of interpretations of the action constants $\sigma^{\mathrm{a}}$, $\mathbf{A} = \mathrm{I}(\sigma^{\mathrm{a}})$;
- $R$ is a set of transitions, $R \subseteq S \times \mathbf{A} \times S$.

A *path* of length $m$ of the labelled transition system $\langle S, \mathbf{A}, R\rangle$ is a sequence $s_0\, \varepsilon_0\, s_1 \cdots s_{m-1}\, \varepsilon_{m-1}\, s_m \quad (m \geq 0)$ such that $(s_{i-1}, \varepsilon_{i-1}, s_i) \in R$ for $i \in 1..m$.

It is convenient in what follows to represent a state by the set of fluent atoms that it satisfies, i.e., $s = \{f = v \mid s \models f = v\}$. A state is then a (complete, and

consistent) set of fluent atoms. We sometimes say a formula $\varphi$ 'holds in' state $s$ or 'is true in' state $s$ as alternative ways of saying that $s$ satisfies $\varphi$.

Action constants in $\mathcal{C}+$ are used to name actions, attributes of actions, or properties of a transition as a whole. Since a transition label/event $\varepsilon$ is an interpretation of the action constants $\sigma^{\mathrm{a}}$, it is meaningful to say that $\varepsilon$ satisfies an action formula $\alpha$ ($\varepsilon \models \alpha$). When $\varepsilon \models \alpha$ we say that the transition $(s, \varepsilon, s')$ is a transition of type $\alpha$. Moreover, since a transition label is an interpretation of the action constants $\sigma^{\mathrm{a}}$, it can also be represented by the set of atoms that it satisfies.

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:

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

where $F$ and $G$ are fluent formulas. Static laws express constraints on states. A state $s$ satisfies a static law (1) if $s \models (G \rightarrow F)$. A *fluent dynamic law* is an expression:

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

where $F$ and $G$ are fluent formulas and $\psi$ 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 $(s, \varepsilon, s')$ with $s \cup \varepsilon \models \psi$, 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 $\alpha$ is an action formula and $\psi$ 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 $\alpha$ must also be of type $\alpha'$ (written $\alpha'$ if $\alpha$), or that any transition from a state satisfying fluent formula $G$ must be of type $\beta$ (written $\beta$ if $G$).

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

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

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

inertial $f$ states that values of the fluent constant $f$ persist by default (by '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.

*Causal theories.* The language $\mathcal{C}+$ is presented in [1] as a higher-level notation for defining particular classes of theories in a general-purpose non-monotonic formalism called 'causal theories'. 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 $\Gamma_m^D$ which encodes the paths of length $m$ in the transition system defined by $D$; moreoever, for every definite causal theory $\Gamma_m^D$ there is a formula $comp(\Gamma_m^D)$ 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 $comp(\Gamma_m^D)$ from the action description $D$ and then employ a (standard, classical) satisfaction solver to determine the models of $comp(\Gamma_m^D)$. This is the method employed in the 'Causal Calculator' CCalc.

A causal theory of signature $\sigma$ is a set of expressions ('causal rules') of the form

$$F \Leftarrow G$$

where $F$ and $G$ are formulas of signature $\sigma$. $F$ is the head of the rule and $G$ is the body. A rule $F \Leftarrow G$ is to be read as saying that there is a cause for $F$ when $G$ is true (which is not the same as saying that $G$ is the cause of $F$).

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

Given a definite action description $D$ in $\mathcal{C}+$, and any non-negative integer $m$, translation to the corresponding causal theory $\Gamma_m^D$ proceeds as follows. The signature of $\Gamma_m^D$ is obtained by time-stamping every fluent constant of $D$ with non-negative integers between 0 and $m$ and every action constant with integers between 0 and $m-1$: 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 $s_0\,\varepsilon_0\,\cdots\,\varepsilon_{m-1}\,s_m$ of the transition system defined by $D$; the atom $a[i]=v$ represents that action atom $a=v$ is satisfied by the transition $\varepsilon_i$ of such a path. The domain of each timestamped constant $c[i]$ is the domain of $c$. In what follows, $\psi[i]$ is shorthand for the formula obtained by replacing every atom $c=v$ in $\psi$ 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 $\Gamma_m^D$ a causal rule of the form

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

For every fluent dynamic law $F$ **if** $G$ **after** $\psi$ 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 $\alpha$ **if** $\psi$ 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:

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

It is straightforward to check [1] that the models of causal theory $\Gamma_m^D$, 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 $D$.

## 3 $n\mathcal{C}+$: Coloured Transition Systems

An action description of $n\mathcal{C}+$ 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 $S_{\mathrm{g}}$ the 'green' states of the system;
- $R_{\mathrm{g}} \subseteq R$, the set of 'permitted' ('acceptable', 'ideal', 'legal') transitions—we call $R_{\mathrm{g}}$ the 'green' transitions of the system.

We refer to the complements $S_{\mathrm{red}} = S - S_{\mathrm{g}}$ and $R_{\mathrm{red}} = R - R_{\mathrm{g}}$ 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. For example, Carmo and Jones [15] employ a structure which has both ideal/sub-ideal states and ideal/sub-ideal transitions (unlabelled). van der Meyden's 'Dynamic logic of permission' [16] employs a structure in which transitions, but not states, are classified as 'permitted/non-permitted'. van der Meyden's version was constructed as a response to problems of Meyer's 'Dynamic deontic logic' [10] which classifies transitions as 'permitted/non-permitted' by reference only to the state resulting from a transition. 'Deontic interpreted systems' [8] classify states as 'green'/'red', where these states have further internal structure to model the local states of agents in a multi-agent context. In all of these examples (and others) the task has been to find axiomatisations of such structures in one form of deontic logic or another. Here we are concerned with a different task, that of devising a language for *defining* coloured transition systems of the form described above.

A coloured transition system $\langle S, \mathbf{A}, R, S_{\mathrm{g}}, R_{\mathrm{g}}\rangle$ must further satisfy the following constraint, for all states $s$ and $s'$ in $S$ and all transitions $(s, \varepsilon, s')$ in $R$:

$$\text{if } (s, \varepsilon, s') \in R_{\mathrm{g}} \text{ and } s \in S_{\mathrm{g}} \text{ then } s' \in S_{\mathrm{g}} \tag{4}$$

We refer to this as the *green-green-green* constraint, or *ggg* for short. (It is difficult to find a suitable mnemonic.) The *ggg* constraint (4) expresses a kind of *well-formedness* principle: a green (permitted, acceptable, legal) transition in a green (permitted, acceptable, legal) state always leads to a green (acceptable, legal, permitted) state. What is the rationale? Since we are here classifying both states and transitions into green/red, it is natural to ask whether there are any relationships between the classification of states and the classification of transitions between them. As observed previously by Carmo and Jones [15] any such relationships are necessarily quite weak. In particular, and *contra* the assumptions underpinning John-Jules Meyer's construction of Dynamic Deontic Logic [10], a red (unacceptable, non-permitted) transition can result in a green (acceptable, permitted) state. Indeed such cases are frequent: suppose that there are two different transitions, $(s, \varepsilon_1, s')$ and $(s, \varepsilon_2, s')$, between a green or red state $s$ and a green state $s'$. It is entirely reasonable that the transition $(s, \varepsilon_1, s')$ is classified as green whereas $(s, \varepsilon_2, s')$ is classified as red. $(s, \varepsilon_1, s')$ might represent an action by one agent, for example, and $(s, \varepsilon_2, s')$ an action by another. This situation cannot arise if the transition system has a tree-like structure in which there is at most one transition between any pair of states, but we do not want to restrict attention to transition systems of this form. Similarly, it is easy to encounter cases in which a green (acceptable, permitted) transition can lead sensibly to a red (unacceptable, non-permitted) state: not all green (acceptable, permitted) transitions from a red state must be such that they restore the system to a green state. Some illustrations will arise in the examples later. The only plausible relationship between the classification of states and the classification of transitions, as also noted by Carmo and Jones [15], is what we called the *ggg* constraint above, if we regard it (as we do) as a required property of any well-formed system specification. Since the *ggg* constraint is so useful for the applications we have in mind, we choose to adopt it as a feature of every coloured transition system.

Note that the *ggg* constraint (4) may be written equivalently as:

$$\text{if } (s, \varepsilon, s') \in R \text{ and } s \in S_{\text{g}} \text{ and } s' \in S_{\text{red}} \text{ then } (s, \varepsilon, s') \in R_{\text{red}} \tag{5}$$

Any transition from a green (acceptable, permitted) state to a red (unacceptable, non-permitted) state must itself be red, in a well-formed system specification.

One can consider a range of other properties that we might require of a coloured transition system: that the transition relation must be serial, for example, or that there must be at least one green state, or that from every green state there must be at least one green transition, or that from every green state reachable from some specified initial state(s) there must be at least one green transition, and so on. The investigation of these, and other, properties is worthwhile but not something we undertake here. We place no restrictions on coloured transition systems, beyond the *ggg* constraint.

*The language $n\mathcal{C}+$.* To avoid having to specify separately which states and transitions are green and which are red, an $n\mathcal{C}+$ action description specifies

those that are red and leaves the remainder to be classified as green by default. This is for convenience, and also to ensure that all states and transitions are classified completely and consistently. (One might ask why the defaults are not chosen to operate the other way round. It is very much more awkward to specify concisely what is green and allow the remainder to be red by default.)

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

$$n\colon \textsf{not-permitted } F \textsf{ if } G \tag{6}$$

where $n$ is an (optional) identifier for the rule and $F$ and $G$ are fluent formulas. $\textsf{not-permitted } F$ is a shorthand for $\textsf{not-permitted } F \textsf{ if } \top$. An *action permission law* is an expression of the form

$$n\colon \textsf{not-permitted } \alpha \textsf{ if } \psi \tag{7}$$

where $n$ is an (optional) identifier for the rule, $\alpha$ is an action formula and $\psi$ is any formula of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$. $\textsf{not-permitted } \alpha$ is shorthand for $\textsf{not-permitted } \alpha \textsf{ if } \top$. We also allow $\textsf{oblig } F$ as an abbreviation for $\textsf{not-permitted } \neg F$ and $\textsf{oblig } \alpha$ as an abbreviation for $\textsf{not-permitted } \neg\alpha$.[1]

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

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

$$\begin{aligned} S_{\mathrm{red}} =_{\mathrm{def}} & \{s \mid s \models F \wedge G \text{ for some law of the form (6) in } D\} \\ R_{\mathrm{red}} =_{\mathrm{def}} & \{(s, \varepsilon, s') \mid s \cup \varepsilon \models \psi,\ \varepsilon \models \alpha \text{ for some law of the form (7) in } D\} \\ & \cup\ \{(s, \varepsilon, s') \mid s \in S_{\mathrm{g}} \text{ and } s' \in S_{\mathrm{red}}\} \end{aligned}$$

The second component of the $R_{\mathrm{red}}$ definition ensures that the *ggg* constraint is satisfied. (The state permission laws $\textsf{not-permitted } F \textsf{ if } G$ and $\textsf{not-permitted } (F \wedge G)$ are thus equivalent in $n\mathcal{C}+$; we allow both forms for convenience.) It can be shown easily [6] that the coloured transition system defined in this way is unique and satisfies the *ggg* constraint. The definition of course does not guarantee that the coloured transition system satisfies any of the other possible properties that we mentioned earlier. If they are felt to be desirable in some particular

[1] This does not raise the issue of 'action negation' as encountered in modal action logics. (See e.g. [12].) In $\mathcal{C}+$ and $n\mathcal{C}+$, $\alpha$ is not the name of an action but a formula expressing a property of transitions.

application, they must be checked separately as part of the specification process. (These checks are easily implemented.)

The overall effect is thus:

- a state is green unless coloured red by some static permission law;
- a transition is red if it is coloured red by some action permission law, or by the *ggg* constraint; otherwise it is green.

That the colouring of transitions is dependent on the colouring of states should *not* be interpreted as a commitment to any philosophical position about the priority of the ought-to-be and the ought-to-do, and the derivability of one from the other. It is merely a consequence of, first, adopting the *ggg* constraint as an expression of the well-formedness of a system specification, and second, of choosing to specify explicitly what is red and letting green be determined by default.

*Causal theories.* Any (definite) action description of $n\mathcal{C}+$ 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 $D_{\text{basic}}$ of $D$ proceeds as usual. For the permission laws, introduce two new fluent and action constants, $\mathsf{status}$ and $\mathsf{trans}$ respectively, both with possible values $\mathsf{green}$ and $\mathsf{red}$. They will be used to represent the colour of a state and the colour of a transition, respectively.

For every state permission law $n$: $\mathsf{not\text{-}permitted}\ F\ \mathsf{if}\ G$ and time index $i \in 0\,..\,m$, include in $\Gamma_m^D$ a causal rule of the form

$$\mathsf{status}[i] = \mathsf{red} \Leftarrow F[i] \wedge G[i] \tag{8}$$

and for every $i \in 0\,..\,m$, a causal rule of the form

$$\mathsf{status}[i] = \mathsf{green} \Leftarrow \mathsf{status}[i] = \mathsf{green} \tag{9}$$

to specify the default colour of a state. A state permission rule of the form $n$: $\mathsf{oblig}\ F\ \mathsf{if}\ G$ produces causal rules of the form $\mathsf{status}[i] = \mathsf{red} \Leftarrow \neg F[i] \wedge G[i]$.

For every action permission law $n$: $\mathsf{not\text{-}permitted}\ \alpha\ \mathsf{if}\ \psi$ and time index $i \in 0\,..\,m{-}1$, include in $\Gamma_m^D$ a causal rule of the form

$$\mathsf{trans}[i] = \mathsf{red} \Leftarrow \alpha[i] \wedge \psi[i] \tag{10}$$

and for every $i \in 0\,..\,m{-}1$, a causal rule of the form

$$\mathsf{trans}[i] = \mathsf{green} \Leftarrow \mathsf{trans}[i] = \mathsf{green} \tag{11}$$

to specify the default colour of a transition. An action permission law of the form $n$: $\mathsf{oblig}\ \alpha\ \mathsf{if}\ \psi$ produces causal rules of the form $\mathsf{trans}[i] = \mathsf{red} \Leftarrow \neg\alpha[i] \wedge \psi[i]$.

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

$$\mathsf{trans}[i] = \mathsf{red} \ \Leftarrow\ \mathsf{status}[i] = \mathsf{green} \wedge \mathsf{status}[i{+}1] = \mathsf{red} \tag{12}$$

It is straightforward to show [6] that models of the causal theory $\Gamma_m^D$ 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.

The translation of $n\mathcal{C}+$ into causal theories effectively treats status = red and trans = red as 'violation constants'. Notice that, although action descriptions in $n\mathcal{C}+$ 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 *ggg* constraint (12). However, implementation in CCALC requires only that the causal laws (8)–(12) are included in the translation to causal theories, which is a very simple modification.

## 4 Examples

The examples in this section are deliberately chosen to be as simple as possible, so that in each case we can show the transition system defined in its entirety. Other examples may be found in [6, 7]. The first example illustrates the use of $n\mathcal{C}+$ in a typical (but very simple) system specification. The second is to motivate the more complicated account to come in Section 5.

*Example (File system).* $I$ is some piece of (confidential) information. $I$, or material from which $I$ can be derived, is stored in a file. Let $x$ range over some set of agent names. Boolean fluent constants $Kx$ represent that agent $x$ has access to information $I$, that $x$ 'knows' $I$. Boolean fluent constants $Fx$ represent that $x$ has read access to the file containing $I$. If $x$ has read access to the file ($Fx$) then $x$ knows $I$ ($Kx$). $Fx$ is inertial: both $Fx$ and $\neg Fx$ persist by default. $\neg Kx$ persists by default but once $Kx$ holds, it holds for ever.

Suppose, for simplicity, that there are two agents, $a$ and $b$. Suppose moreover that the file is the only source of information $I$, in the sense that if $Kx$ holds for any $x$ then either $Fa$ or $Fb$. This does not change the essence of the example but it reduces the number of states and simplifies the diagrams.

There are two types of acts: Boolean action constants $read(x)$ represent that $x$ is given read access to the file containing $I$. Boolean action constant $a$ *tells* $b$ represents that $a$ communicates to $b$ the information $I$ (whether or not $b$ knows it already), and $b$ *tells* $a$ that $b$ communicates it to $a$. In this simple example there are no actions by which read access to the file is removed once it is granted.

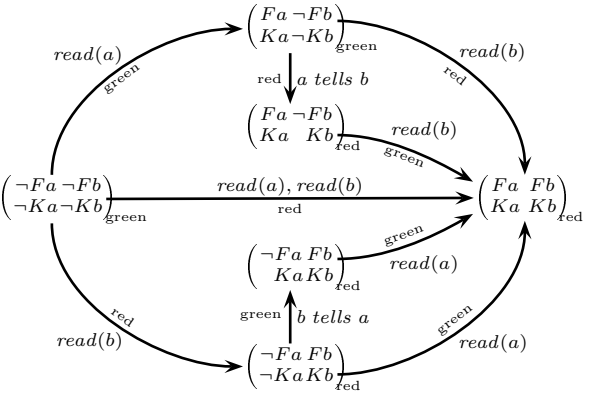
We can represent the above as a definite action description as follows, for $x$ ranging over $a$ and $b$.

| | |
|---|---|
| inertial $Fx$ | $read(x)$ causes $Fx$ |
| $\neg Kx$ if $\neg Kx$ after $\neg Kx$ | $a$ *tells* $b$ causes $Kb$ |
| $Kx$ if $\top$ after $Kx$ | $b$ *tells* $a$ causes $Ka$ |
| | nonexecutable $a$ *tells* $b$ if $\neg Ka$ |
| $Kx$ if $Fx$ | nonexecutable $b$ *tells* $a$ if $\neg Kb$ |
| $\bot$ if $Kx \wedge \neg Fa \wedge \neg Fb$ | nonexecutable $read(x)$ if $Fx$ |

Now suppose that $a$ is permitted to know $I$, and $b$ is not. We add the following law to the action description. ($Ka$ is permitted by default.)

$p(b)$: not-permitted $Kb$

The transition system defined by this action description is shown below. The labels *read*($a$), *read*($b$), *a tells b*, *b tells a* stand for the transition labels {*read*($a$), ¬*read*($b$), ¬*a tells b*, ¬*b tells a*}, {¬*read*($a$), *read*($b$), ¬*a tells b*, ¬*b tells a*} and so on, respectively. The label *read*($a$), *read*($b$) is shorthand for the transition label {*read*($a$), *read*($b$), ¬*a tells b*, ¬*b tells a*}. Reflexive arcs, corresponding to the 'null event' or to transitions of type *a tells b* and *b tells a* from state $\{Fa, Ka, \neg Fb, Kb\}$ to itself, are omitted from the diagram to reduce clutter. Also omitted from the diagram are transitions of type *read*($a$) ∧ *a tells b*, *a tells b* ∧ *b tells a*, etc. Again, this is just to reduce clutter.

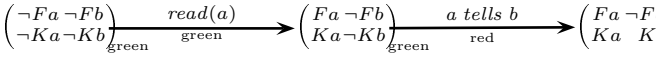



Notice that transitions of type *read*($b$) are red because of the *ggg* constraint, except that *read*($b$) transitions come out to be green in states where $Kb$ already holds. If the latter is felt to be undesirable, one could add another action permission law not-permitted *read*($b$), or a state permission law not-permitted $Fb$. We will discuss some of these options in more detail later.

In a computerised system, $b$'s access to information $I$ would be controlled by the file access system. Naturally the file access system cannot determine whether $b$ knows $I$: in practice, a specification of the computer system would simply say that *read*($b$) actions are nonexecutable, or simply that $Fb$ is false. The latter can be expressed by adding the following static law to the action description:

$\perp$ if $Fb$

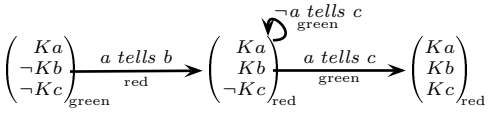
This eliminates all states in which $Fb$ holds from the transition system. The transition system defined by this extended action description is the following:



As usual, reflexive arcs are omitted from the diagram for clarity. Here, the action *read*($a$) is under the control of the file access system, and *a tells b* is an action that can be performed by agent $a$. This difference is not explicit in the semantics of $\mathcal{C}+$ nor of $n\mathcal{C}+$. The *agent-centric* version of $n\mathcal{C}+$, alluded to in the introduction, allows such distinctions to be made.

*Example (Secrets).* Suppose we have agents $a$, $b$, and $c$, Boolean fluent constants $Ka$, $Kb$, and $Kc$ as in the previous example, and Boolean action constants *a tells b* and *a tells c*. We ignore the file system and read access to it from now on since they play no role in this example, and we leave out other possible actions such as *b tells a*, *c tells a*, etc, to simplify the diagrams. The persistence of $Ka$, $Kb$, and $Kc$, and the effects of *a tells b* and *a tells c* actions are represented using $\mathcal{C}+$ laws as shown earlier.

Suppose now that $b$ is not permitted to know $I$. The coloured transition system contains the following fragment:




The states $\{Ka, Kb, \neg Kc\}$ and $\{Ka, Kb, Kc\}$ are red because $Kb$ is not permitted. The transition labelled *a tells b* is red because of the *ggg* constraint. The transition labelled *a tells c* is not forced to be red by the *ggg* constraint and so becomes green by default. The (reflexive) transition labelled $\neg$*a tells c* is also green for the same reason.

But suppose now that we change the example, by adding that $c$ is also not permitted to know $I$. The fragment of the transition system shown above remains unchanged (because the state $\{Ka, Kb, Kc\}$ was already red). The transition labelled *a tells c* is green even though *a tells c* results in $Kc$, and $Kc$ is not permitted. We have here an instance of a general phenomenon: once a state is red, all transitions from it (including actions by all other agents) become green by default unless explicitly coloured red by action permission laws.

One possibility is to leave some transitions uncoloured, or what comes to the same thing, remove the default colouring of transitions and allow an $n\mathcal{C}+$ action description to define multiple transition systems differing in the colours assigned to some transitions. This is easy to encode, and easy to implement, but it is too weak for what we want: we would then never be able to conclude that there are (necessarily) green transitions from any red state.

Our diagnosis is that the classification of states into red/green is too crude. Why should we think that *a tells c* transitions should be inferred red even after *a tells b* has occurred and $Kb$ holds? Because *a tells c* would lead to violation of another norm which says that $Kc$ is not permitted. So we will introduce names for (instances of) norms and then classify states according to how well, or how badly, they comply with these norms.

## 5 $n\mathcal{C}+$: Graded Transition Systems

A *graded transition system* is a structure of the form: $\langle S, \mathbf{A}, R, R_{\mathrm{g}}, \prec \rangle$ where $\langle S, \mathbf{A}, R \rangle$ is a labelled transition system of the kind defined by $\mathcal{C}+$ action descriptions, and where

- $R_{\mathrm{g}} \subseteq R$ is the set of 'green' transitions;
- $\prec$ is a (strict, partial) ordering on $S$: $s \prec s'$ represents that state $s'$ is worse than state $s$.

We refer to $R_{\mathrm{red}} = R - R_{\mathrm{g}}$ as the 'red transitions' as usual.

Notice that we have chosen to grade/rank states but not transitions: transitions are still either green or red. There may be good reasons to rank transitions as well, but we will not do so here.

As in the case of coloured transition systems, we further impose a well-formedness constraint, analogous to the *ggg* constraint. The natural generalization of *ggg* is to require that in any green transition $(s, \varepsilon, s')$ from state $s$ to state $s'$, the resulting state $s'$ must be no worse than the state $s$: $(s, \varepsilon, s') \in R_{\mathrm{g}}$ implies $s \not\prec s'$. This constraint may be written equivalently as:

$$\text{if } (s, \varepsilon, s') \in R \text{ and } s \prec s' \text{ then } (s, \varepsilon, s') \in R_{\mathrm{red}} \tag{13}$$

We refer to (13) as the *BRW* constraint (short for 'better-red-worse'), again apologising for the ugliness of the label.

A coloured transition system is thus a special case of a graded transition system in which $s \prec s'$ iff $s \in S_{\mathrm{g}}$ and $s' \in S_{\mathrm{red}}$. In that case the *BRW* constraint (13) takes the form: if $(s, \varepsilon, s') \in R$ and $s \in S_{\mathrm{g}}$ and $s' \in S_{\mathrm{red}}$ then $(s, \varepsilon, s') \in R_{\mathrm{red}}$, which is equivalent to the formulation of the *ggg* constraint given earlier (4).

Note that according to the *BRW* constraint, a transition $(s, \varepsilon, s')$ is not forced to be red if $s \not\prec s'$. In particular, when $s$ and $s'$ are not comparable in $\prec$, the *BRW* constraint does not apply. This is deliberate. One could look for stronger well-formedness constraints than *BRW*. The requirement that $(s, \varepsilon, s') \in R_{\mathrm{g}}$ implies $s' \prec s$ is clearly much too strong, but there are several candidates stronger than *BRW* for which a plausible case can be made. We are inclined, however, not to adopt any of these stronger constraints as a fixed feature of graded transition systems.

*Violation orderings.* Of particular interest is the special case of graded transition systems where the ordering on states is determined by how well, or how badly, each state complies with a set of explicitly named norms.

A normative code $\mathcal{N}$ is a finite set of pairs $\langle n, o(F/G) \rangle$ where $n$ is an identifier for a norm, and $F$ and $G$ are fluent formulas. Note that we do not require that norm labels are unique in $\mathcal{N}$.

The *violation set* $V_{\mathcal{N}}(s)$ of a state $s$ in $S$ is the set of norm identifiers in $\mathcal{N}$ that are violated in $s$.

$$V_{\mathcal{N}}(s) =_{\mathrm{def}} \{n \mid \langle n, o(F/G) \rangle \in \mathcal{N} \text{ and } s \models G \wedge \neg F\} \tag{14}$$

Now we can define an ordering on states by comparing their violation sets. A state $s$ is better than a state $s'$ if the violation set of $s$ is a proper subset of the violation set of $s'$:

$$s \prec_{\mathcal{N}} s' \text{ iff } V_{\mathcal{N}}(s) \subset V_{\mathcal{N}}(s') \tag{15}$$

It would be easy to add weights or priorities on norms, and adjust the definition of $\prec_{\mathcal{N}}$ to take these weights into account. The details are straightforward and we omit them.

Let $D$ be an action description of $n\mathcal{C}+$. The graded transition system defined by $D$ is $\langle S, \mathbf{A}, R, R_{\rm g}, \prec_{\mathcal{N}}\rangle$ where the states $S$, transition labels/events $\mathbf{A}$, and transitions $R$ are exactly as in the coloured transition system described in Section 3; $R_{\rm g} = R - R_{\rm red}$ where the red transitions $R_{\rm red}$ are determined by the action permission laws and the *BRW* constraint; and where the ordering $\prec_{\mathcal{N}}$ on states is the ordering defined in (15) by the normative code $\mathcal{N}$ consisting of elements $\langle n, o(\neg F/G)\rangle$ where $n$: **not-permitted** $F$ **if** $G$ is a law in $D$ and elements $\langle n, o(F/G)\rangle$ where $n$: **oblig** $F$ **if** $G$ is a law in $D$.

*Encoding in causal theories.* Let the Boolean fluent constant $\mathsf{viol}(n)$ represent that norm $n$ in $\mathcal{N}$ is violated. For every state permission law $n$: **not-permitted** $F$ **if** $G$ and time index $i \in 0\,..\,m$, include in the causal theory $\Gamma^D_m$ the causal rules:

$$\neg\mathsf{viol}(n)[i] \Leftarrow \neg\mathsf{viol}(n)[i] \tag{16}$$

$$\mathsf{viol}(n)[i] \Leftarrow F[i] \wedge G[i] \tag{17}$$

A state permission rule of the form $n$: **oblig** $F$ **if** $G$ produces causal rules of the form $\mathsf{viol}(n)[i] \Leftarrow \neg F[i] \wedge G[i]$.

(In place of the Boolean violation constants $\mathsf{viol}(n)[i]$ we could have used fluent constants $\mathsf{status}(n)[i]$ with possible values **green** and **red**.)

In order to encode the *BRW* constraint, it is not necessary to compute and compare violation sets for each state. Instead, we can encode the *BRW* constraint as follows. Include in $\Gamma^D_m$, for every $i \in 0\,..\,m{-}1$ and every norm identifier $n$ in $\mathcal{N}$, the causal rules:

$$\mathsf{trans}[i] = \mathsf{green} \;\Leftarrow\; \mathsf{trans}[i] = \mathsf{green} \tag{18}$$

$$\mathsf{trans}[i] = \mathsf{red} \;\Leftarrow\; \mathsf{viol}(n)[i{+}1] \wedge \neg\mathsf{viol}(n)[i] \wedge \neg\mathsf{q}[i] \tag{19}$$

$$\neg\mathsf{q}[i] \;\Leftarrow\; \neg\mathsf{q}[i] \tag{20}$$

$$\mathsf{q}[i] \;\Leftarrow\; \mathsf{viol}(n)[i] \wedge \neg\mathsf{viol}(n)[i{+}1] \tag{21}$$

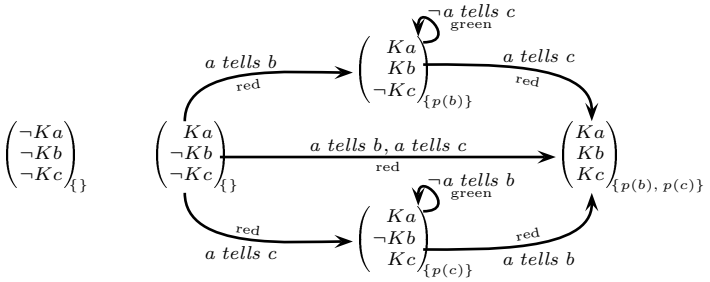
Causal rules (18) and (19) generalise the causal rules (11) and (12) used to encode the *ggg* constraint in causal theories. They make use of auxiliary constants $\mathsf{q}[i]$ defined in (20)–(21). One can easily check that in the case where the action description contains a single state permission law, causal rules (18)–(21) collapse to a form equivalent to the causal rules (11) and (12) encoding the *ggg* constraint. In the case where the action description contains no state permission law, these two sets of causal laws are trivially equivalent, since both are empty.

*Example (Secrets, contd).* Suppose we formulate the example of Section 4 using two state permission laws as follows:

$$p(b)\text{: not-permitted } Kb \tag{22}$$

$$p(c)\text{: not-permitted } Kc \tag{23}$$

The graded transition system defined is as follows, where the annotations on the states show the respective violation sets.




'Null events' and other reflexive arcs are omitted from the diagram for clarity.

Contrast this with a different version of the example. Suppose that instead of permission laws (22) and (23) with two explicit norms labelled $p(b)$ and $p(c)$ we specify just one explicit norm with a single label $p$, either in the form

$$p\text{: not-permitted } Kb \tag{24}$$

$$p\text{: not-permitted } Kc \tag{25}$$

or equivalently as a single state permission law: $p$: not-permitted $(Kb \vee Kc)$.

In this version of the action description, the three states $\{Ka, Kb, \neg Kc\}$, $\{Ka, \neg Kb, Kc\}$, and $\{Ka, \neg Kb, \neg Kc\}$ all have the same violation set, $\{p\}$. Since they are now not strictly worse than each other, the *BRW* constraint does not colour transitions between them red: they are all green by default. This transition system corresponds to the example in Section 4, but now with states annotated by violation sets $\{\}$ and $\{p\}$ rather than colours 'green' and 'red'.

## 6 Conclusion

$n\mathcal{C}+$ adds a simple deontic component to $\mathcal{C}+$, intended to support system specifications where there is a need to distinguish between acceptable/permitted and unacceptable/non-permitted system states and behaviours.

It was our intention to continue the discussion of the examples to show how $n\mathcal{C}+$ copes with (temporal) 'contrary-to-duty' structures. For instance, a natural extension of the examples would take the form '$a$ must not tell $b$ and $a$ must not tell $c$; but if $a$ tells $b$ it must tell $c$, and if $a$ tells $c$ it must tell $b$', as in Belzer's Reykjavik scenario [17]. Another interesting variant is '$b$ is not permitted to know $I$ and $c$ is not permitted to know $I$; but if $a$ tells $b$ it must tell $c$, and if $a$ tells $c$ it must tell $b$'. We leave that discussion for a separate paper.

Besides investigating variants of the *BRW* well-formedness constraint and other desirable properties of coloured/graded transition systems, we are developing a refined version of $n\mathcal{C}+$ to make explicit the distinction between actions and transitions, and an agent-centric $n\mathcal{C}+$ for specifying system norms as directives that constrain an individual agent's behaviour.

## References


1. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. Artificial Intelligence **153** (2004) 49–104
2. CCalc: `http://www.cs.utexas.edu/users/tag/cc`.
3. Akman, V., Erdoğan, S.T., Lee, J., Lifschitz, V., Turner, H.: Representing the Zoo World and the Traffic World in the language of the Causal Calculator. Artificial Intelligence **153** (2004) 105–140
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. 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. Sergot, M.: $(\mathcal{C}+)^{++}$: An action language for modelling norms and institutions. Technical Report 2004/8, Dept. of Computing, Imperial College London (2004)
7. Sergot, M.J.: Modelling unreliable and untrustworthy agent behaviour. In Dunin-Keplicz, B., Jankowski, A., Skowron, A., Szczuka, M., eds.: Monitoring, Security, and Rescue Techniques in Multiagent Systems. Advances in Soft Computing. Springer-Verlag (2005) 161–178
8. Lomuscio, A., Sergot, M.J.: Deontic interpreted systems. Studia Logica **75**(1) (2003) 63–92
9. Lomuscio, A., Sergot, M.J.: A formalisation of violation, error recovery, and enforcement in the bit transmission problem. J. of Applied Logic **2** (2004) 93–116
10. Meyer, J.J.C.: A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. Notre Dame J. of Formal Logic **29**(1) (1988) 109–136
11. Maibaum, T.: Temporal Reasoning over Deontic Specifications. In Meyer, J.J.C., Wieringa, R.J., eds.: Deontic Logic in Computer Science: Normative System Specification. John Wiley & Sons, Chichester, England (1993) 141–202
12. Broersen, J.: Modal Action Logics for Reasoning about Reactive Systems. PhD thesis, Vrije Universiteit Amsterdam (2003)
13. 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
14. van der Torre, L.: Causal deontic logic. In: Proceedings of the Fifth Workshop on Deontic Logic in Computer Science (Deon2000). (2000) 351–367
15. Carmo, J., Jones, A.J.I.: Deontic database constraints, violation and recovery. Studia Logica **57**(1) (1996) 139–165
16. Meyden, R.: The dynamic logic of permission. Journal of Logic and Computation **6**(3) (1996) 465–479
17. Belzer, M.: Legal reasoning in 3-D. In: Proc. 1st International Conf. on Artificial Intelligence and Law, Boston, ACM Press (1987) 155–163