Violation, error recovery, and enforcement in the bit transmission problem

Alessio Lomuscio Marek Sergot
Department of Computing
Imperial College of Science, Technology and Medicine
London SW7 2BZ, UK

Abstract

The design of complex multi-agent systems is increasingly having to confront the possibility that agents may not behave as they are supposed to. In addition to analysing the properties that hold if protocols are followed correctly, it is also necessary to predict, test, and verify the properties that would hold if these protocols were to be violated. We illustrate how the formal machinery of deontic interpreted systems can be applied to the analysis of such problems by considering three variations of the bit transmission problem. The first, an example in which an agent may fail to do something it is supposed to do, shows how we deal with violations of protocols and specifications generally. The second, an example in which an agent may do something it is not supposed to do, shows how it is possible to specify and analyse remedial or error-recovery procedures. The third combines both kinds of faults and introduces a new component into the system, a controller whose role is to enforce compliance with the protocol. In each case the formal analysis is used to test whether critical properties of the system are compromised, in this example, the reliable communication of information from one agent to the other.

1 Introduction

The design of complex multi-agent systems is increasingly having to confront the possibility that agents may not behave as they are supposed to. In e-commerce, in security, in automatic negotiation, in any application where agents are programmed by different parties with competing interests, it is unrealistic to assume that all agents will behave according to some given protocol or standard of behaviour. In addition to analysing the properties that hold if protocols are followed correctly, it is also necessary to predict, test, and verify the properties that would hold if these protocols were to be violated, and to test the effectiveness of introducing proposed enforcement mechanisms.

Consider an online auction mechanism modelled and implemented as a multi-agent system, where agents play the parts of auctioneers and bidders. At each round the protocol of the system gives the opportunity to a selected number of agents to bid for some goods, according to the specific auction protocol being employed. Suppose that an agent bids an amount which it is unable to fulfil (either by mistake, or perhaps with the sole intention of raising the bid so that an opponent will have to pay a higher price). If that bid is made in the prescribed manner then it would still count as a valid bid—even if it were to be regarded as illegal, anti-social, unethical to make a bid in those circumstances. Undesirable behaviour of this kind will often have adverse effects on the system as a whole. Perhaps the agent will be forced to de-commit from the commitment it entered upon by bidding, as a result of which the seller will (at least temporarily) lose the deal, which in turn may have consequences on other deals it has entered upon with other agents. At the very least, the resulting disputes will have to be resolved at a cost.

One way out of such problems is to attempt to devise tighter controls in the protocols, such as stipulating that a bid by an agent is valid only if made with previously cleared funds. The opportunities for inventing such controls are limited, however, and in any case have associated costs too (in the example, the costs of the clearing mechanism and the consequently diminished liquidity of the market). Moreover forcing agents of an open multi-agent system to behave according to a rigid set of rules is seen as increasingly unfeasible because of the inherent distribution of the system. One alternative, having specified what is correct, desirable, or acceptable behaviour of the agents within a given context, is to strive to handle and reason about violations of these norms within the system. The required formalisms not only have to be capable of handling *local violations*, but must also be capable of representing the usual attributes ascribed to agents in a multi-agent setting, such as knowledge, intentions, and goals (see e.g., [Woo00]).

This then is the context within which the work presented in this paper is being undertaken. It is important to emphasise that the focus here is not on formalisms used internally by the agents as part of their reasoning mechanisms, but rather for specification of properties of the system as a whole, ideally accompanied by verification mechanisms. The question of how an agent may reason internally about the norms that constrain its behaviour is an interesting one, but it is a different question that will not be pursued in this paper. Note that it is legitimate and perfectly meaningful to make this separation. Consider the familiar problem of controlling access to sensitive or confidential data. There is a specification of what access is permitted or forbidden at the system level but the agents who actually make the access need not be aware, and usually are not aware, of what this specification is.

These are inherently complex issues, and we do not expect to have methods able to give comprehensive answers to all of them in the near future. One approach being explored by the authors as a step in this direction is an attempt to extend interpreted systems [FHMV95], a mainstream and well-developed semantics for reasoning about knowledge in distributed and multi-agent systems, with a deontic component. Deontic interpreted systems [LS01a, LS01b] are designed to represent correct and incorrect functioning behaviour of agents as well as their epistemic properties.

While a considerable amount of theoretical work considers constructs not dissimilar to those employed here (among others: [ML85, Mey88, KM89, FM91, GMP92, Coe93, JS93, CJ96]), we are not aware of any study that puts these theoretical constructs to the test in an actual application. It is surprisingly difficult to find concrete examples in the literature, let alone case studies showing how the formal approaches may be applied in detail. Our aim in this paper is to advance the state-of-the-art by constructing a fully worked out formal analysis of a concrete example, and to illustrate and evalate the formalism of deontic interpreted systems in the process.

Specifically, we show how the formalism of deontic interpreted systems [LS01a, LS01b] may be applied to model and reason about protocol violations in a simple example, the bit-transmission problem [HZ92]. This is a much discussed scenario in distributed computing involving two agents attempting to communicate the value of a bit over a faulty communication channel. Of course, the example is trivial compared to the kinds of applications alluded to in the introductory sections. Still, we argue that it is rich enough to raise many important issues such as error correction and control/enforcement mechanisms, and in this sense is representative. Furthermore, using a small, well understood, and widely discussed problem offers the possibility of testing rigorously a formal apparatus and investigating how it performs in detail.

We will examine three variations of the bit-transmission problem. The first (Section 5) is an example in which an agent may fail to do something it is supposed to do, and shows how we deal with violations of protocols and specifications generally. The second (Section 6) is an example in which an agent may do something it is not supposed to do, and shows how it is possible to specify and analyse remedial or error-recovery procedures. The third (Section 7) combines both kinds of faults and introduces a new component into the system, a controller whose role is to *enforce* compliance with the protocol. In each case the formal analysis is used to test whether critical properties of the system are compromised, in this example, the reliable communication of information from one agent to the other.

2 Preliminaries

In this paper we use the machinery of interpreted systems as presented in [FHMV95], and the extensions for modelling correct and incorrect functioning behaviours as developed in [LS01a, LS01b]. We present the main definitions here.

Consider n non-empty sets $L_1, \ldots, L_n$ of local states, one for every agent of the system, and a set of states for the environment L_E. Elements of L_i will be denoted by $l_1, l_1', l_2, l_2', \ldots$. Elements of L_E will be denoted by $l_E, l_E', \ldots$.

Definition 1 (System of global states) *A* system of global states for n agents *S is a non-empty subset of a Cartesian product $L_E \times L_1 \times \cdots \times L_n$.*

An interpreted *system of global states is a pair $IS = (S, h)$ where S is a system of global states and $h : S \to 2^P$ is an interpretation function for a set of propositional variables P.*

When $g = (l_E, l_1, \ldots, l_n)$ is a global state of a system S, $l_i(g)$ denotes the local state of agent i in global state g. $l_E(g)$ denotes the local state of the environment in global state g.

Systems of global states can be used to interpret epistemic modalities K_i, one for each agent.

$$(IS, g) \models K_i \varphi \quad \text{if for all } g' \text{ we have that } l_i(g) = l_i(g') \text{ implies } (IS, g') \models \varphi.$$

Alternatively one can consider generated models $(S, \sim_1, \ldots, \sim_n, h)$ of the standard relational (Kripke) kind, where the equivalence relations $\sim_i$ are defined on equivalence of local states, and then interpret modalities in the usual way. The resulting logic for modalities K_i is of type $S5_n$; this models agents with complete introspection capabilities and veridical knowledge [Hin62, MH95], though it is important to see that the notion of knowledge modelled here is an information-theoretic one. An agent 'knows' φ when from an *external* perspective it has enough information to tell that φ is true.

The notion of interpreted systems can be extended to incorporate the idea of correct functioning behaviour of some or all of the components [LS01a, LS01b], as follows.

Definition 2 (Deontic system of global states) *Given n agents and $n+1$ non-empty sets $G_E, G_1, \ldots, G_n$, a* deontic system of global states *is any system of global states defined on $L_E \supseteq G_E, \ldots, L_n \supseteq G_n$. G_E is called the* set of green states for the environment*, and for any agent i, G_i is called* the set of green states for agent i. *The complement of G_E with respect to L_E (respectively G_i with respect to L_i) is called the* set of red states for the environment (respectively for agent i).

The terms 'green' and 'red' are chosen as neutral terms, to avoid overloading them with unintended readings and connotations. The term 'green' can be read as 'legal', 'acceptable', 'desirable', 'correct', depending on the context of a given application.

Deontic systems of global states are used to interpret modalities such as the following

$$(IS, g) \models O_i\,\varphi \quad \text{if for all } g' \text{ we have that } l_i(g') \in G_i \text{ implies } (IS, g') \models \varphi.$$

$O_i\,\varphi$ is used to represent that φ holds in all (global) states in which agent i is functioning correctly. Again, one can consider generated models $(S, \sim_1, \ldots, \sim_n, R_1^O, \ldots, R_n^O, h)$ of the standard kind, where the equivalence relations are defined as above and the relations R_i^O are defined by $g\,R_i^O\,g'$ if $l_i(g') \in G_i$, with a standard modal logic interpretation for the operators. In particular, the semantics of O_i is essentially that of a (strengthened) obligation operator in standard deontic logic (O_i is a normal modality of type $KD45$). However, it would not be appropriate to read the expression $O_i\,\varphi$ as 'it is obligatory for agent i that φ'.

Knowledge can be modelled on deontic interpreted systems in the same way as on interpreted systems, and one can study various combinations of the modalities such as $K_i\,O_j$, $O_j\,K_i$, and others. It is particularly important when reading these expressions, however, to remember that they express the external "bird's eye" view of the system: $O_j\,K_i\,\varphi$ says that in all states of the system in which agent j is functioning correctly, from an external perspective agent i has sufficient information to know that φ. There are many other senses of 'ought to know' that are not captured by this construction.

Another concept of particular interest is knowledge that an agent i has *on the assumption that the system (the environment, agent j, group of agents X) is functioning correctly.* We employ the (doubly relativised) modal operator $\widehat{K}_i^j$ for this notion, interpreted as follows:

$$(IS, g) \models \widehat{K}_i^j\,\varphi \quad \begin{array}{l}\text{if for all } g' \text{ such that } l_i(g) = l_i(g') \text{ and}\\ l_j(g') \in G_j \text{ we have that } (IS, g') \models \varphi.\end{array}$$

K_i, O_i, and $\widehat{K}_i^j$ are all normal modalities (of type S5, $KD45$, and $KD45$, respectively). Various relationships hold between them, such as $K_i\,\varphi \rightarrow \widehat{K}_i^j\,\varphi$, $K_i\,O_j\,\varphi \rightarrow \widehat{K}_i^j\,\varphi$, $O_j\,\varphi \rightarrow \widehat{K}_i^j\,\varphi$, and others. Further details, and metalogical properties such as frame correspondence and completeness, may be found in [LS01a, LS01b]. Our aim in this present paper is to illustrate the use of these modalities in the examples to follow.

Finally, interpreted systems can be extended to deal with temporal evolution. Consider a set of runs over global states $R = \{r : N \rightarrow S\}$, representing flows of time for the system. When this structure is in place one can interpret the usual temporal connectives on it (see e.g. [GHR93]).

3 The bit transmission problem

The bit-transmission problem [FHMV95] involves two agents, a *sender* S, and a *receiver* R, communicating over a possibly faulty communication channel. S wants to communicate some information—the value of a bit for the sake of the example—to R. We would like to design a protocol that accomplishes this objective while minimising the use of the communication channel.

Two (trivial) special cases of the scenario can immediately be solved. The first is the one in which the channel is actually working, or is at least operative for the first few rounds of computation. In this case we would require S to send the value of the bit once as the system comes alive. The other special case arises when the channel is constantly non-operative. There is obviously no protocol that can ensure that R receives the information in that case.

The interesting scenario arises when the channel is working correctly at certain times while failing at others. There are several ways in which this can be approached, for instance by stipulating that the channel delivers messages with a fixed probability $P > 0$ at any given round. In this paper we do not make any of these assumptions; instead, we analyse the most general case with respect to a most simple protocol. The protocol is as follows. S immediately starts sending the bit to R, and continues to do so until it receives an acknowledgement from R. R does nothing until it receives the bit; from then on it sends acknowledgements of receipt to S. S stops sending the bit to R when it receives an acknowledgement. Note that R will continue sending acknowledgments even after S has received its acknowledgement. Intuitively S will know for sure that the bit has been received by R when it gets an acknowledgement from R. R, on the other hand, will never be able to know whether its acknowledgement has been received since S does not answer the acknowledgement[1]. We refer to [HZ92, FHMV95] for further discussion.

4 Analysis

The bit-transmission problem can be analysed using the formalism of interpreted systems. To do this we follow the approach taken by Halpern and colleagues in

[1]One might think that this problem can be solved by insisting that S sends an acknowledgement of the acknowledgement, but by doing so we simply push the problem one level deeper, and S would never know whether its acknowledgement of the acknowledgement has been received by R.

[FHMV95].

There are three active components in the scenario: a sender, a receiver, and a communication channel. In line with the spirit of the formalism of interpreted systems, it is convenient to see sender and receiver as agents, and the communication channel as the environment. Each of these can be modelled by considering their local states. For the sender S, it is enough to consider four possible local states. They represent the value of the bit S is attempting to transmit, and whether or not S has received an acknowledgement from R. Three different local states are enough to capture the state of R: the value of the received bit, if any, and the circumstance in which no bit has been received yet, represented by ϵ. So we have

$$L_S = \{0,\, 1,\, (0, ack),\, (1, ack)\}, \quad L_R = \{0,\, 1,\, \epsilon\}.$$

To model the environment it is often convenient to represent in the local state (amongst other things) the combinations of messages that were sent in the previous round, by S and R, respectively. The four local states would then be:

$$L_E = \{(.,.),\ (sendbit, .),\ (., sendack),\ (sendbit, sendack)\},$$

where '.' represents configurations in which no message was sent by the corresponding agent. However, for the examples we will be analysing in this paper this level of detail is unnecessary, and so for simplicity we shall take the local states of the environment to be just a singleton:

$$L_E = \{\cdot\}.$$

This is to simplify (and shorten) the presentation. It should be obvious how the formulas of later sections can be adjusted for examples requiring a more complicated representation of the environment's local states.

Global states for the system G are defined as $G \subseteq L_S \times L_R \times L_E$. A global state $g = (l_S, l_R, l_E)$ gives a snapshot of the system at a given time. Note that not all triples of the product are admissible in principle, but only those that can be reached in a run of the protocol from given initial configurations, as will be explained below. Although in this simple example the L_E component is fixed, it is not redundant, and we will retain it in the global states as a reminder that reference to it would be required in more complicated settings.

The interpreted systems framework has the advantage of being suitable for integration with finer semantics representing actions and protocols. To do so consider a set of actions Act_i for every agent in the system and the environment. The actions for the agents S and R are as follows.

$$Act_S = \{sendbit,\, \lambda\}, \quad Act_R = \{sendack,\, \lambda\}$$

Here λ stands for no action ('no-op').

The actions Act_E for the environment correspond to the transmission of messages between S and R on the unreliable communication channel. To make the example sufficiently rich, we will assume that the communication channel can transmit messages in both directions simultaneously, and that a message travelling in one direction can get through while a message travelling in the opposite direction is lost. (Alternatively, think of a pair of unidirectional communication channels whose faults are independent of one other.) The set of actions for the environment is

$$Act_E = \{\leftrightarrow, \rightarrow, \leftarrow, -\}$$

$\leftrightarrow$ represents the action in which the channel transmits any message successfully in both directions, $\rightarrow$ that it transmits successfully from S to R but loses any message from R to S, $\leftarrow$ that it transmits successfully from R to S but loses any message from S to R, and $-$ that it loses any messages sent in either direction. Alternatively, we could model the communications channel by having a single action for the environment, *transmit*, and then representing the unreliability of the channel by four states in the local states of the environment. It comes to much the same thing, except that this alternative treatment makes the specification of the rest of the system more cumbersome.

Under the assumptions of determinism we can model the evolution of the system by means of a transition function $\pi : G \times Act \rightarrow G$, where $Act = Act_S \times Act_R \times Act_E$ is the set of joint actions for the system. To conserve space we do not present the full definition of π for the example. The parts omitted are added straightforwardly. π codes the fact that the state of the communication channel determines whether the actions performed by the agents (i.e., the messages they send on the channel) get through or not, and what their effects are. For example, the definition of π contains the following:

$$\begin{array}{rcl}
\pi(\,(0,\epsilon,\cdot),\,(\mathit{sendbit},\lambda,\leftrightarrow)\,) & = & (0,0,\cdot),\\
\pi(\,(0,\epsilon,\cdot),\,(\mathit{sendbit},\lambda,\rightarrow)\,) & = & (0,0,\cdot),\\
\pi(\,(0,\epsilon,\cdot),\,(\mathit{sendbit},\lambda,\leftarrow)\,) & = & (0,\epsilon,\cdot),\\
\pi(\,(0,\epsilon,\cdot),\,(\mathit{sendbit},\lambda,-)\,) & = & (0,\epsilon,\cdot),
\end{array}$$

to capture that when the channel works properly the message *sendbit* from S gets through and gets processed accordingly by R.

Some other cases are as follows:

$$\begin{array}{rcl}
\pi(\,(0,0,\cdot),\,(\mathit{sendbit},\mathit{sendack},\gamma_E)\,) & = & ((0,\mathit{ack}),0,\cdot) \qquad (\gamma_E \in \{\leftarrow,\leftrightarrow\})\\
\pi(\,(0,0,\cdot),\,(\mathit{sendbit},\mathit{sendack},\gamma_E)\,) & = & (0,0,\cdot) \qquad (\gamma_E \in \{\rightarrow,-\})
\end{array}$$

Other cases can be similarly expressed. We leave the details to the reader.

For compliance with a given protocol, only certain actions are performable at a given time for an agent. For example if S has not yet received an acknowledgement from R, i.e., in the model when S is in the local state 0 or 1, then according to the simple protocol under consideration, S should send the value of the bit over the channel to R, i.e., perform the action *sendbit*. To capture such requirements the concept of *protocol* can be used. A protocol for agent i is a function $P_i : L_i \rightarrow 2^{Act_i}$ mapping sets of actions from a local state. $P_i(l_i)$ is the set of actions performable according to the protocol by agent i when its local state is l_i. For the example under consideration the protocol can be defined as follows:

$$\begin{array}{l}
P_S(0) = P_S(1) = \mathit{sendbit},\ P_S((0,\mathit{ack})) = P_S((1,\mathit{ack})) = \lambda,\\
P_R(\epsilon) = \lambda,\ P_R(0) = P_R(1) = \mathit{sendack}.
\end{array}$$

Throughout, as here, we omit brackets when writing singleton sets, to reduce clutter.

$(0,0)$		$((0,ack),0)$	
	$(1,1)$		$((1,ack),1)$
$(0,\epsilon)$	$(1,\epsilon)$		

Figure 1: The state space of the bit-transmission system. Columns (respectively rows) represent global states epistemically equivalent for S (respectively for R).

For the environment, we use the constant function:

$$P_E(l_E) = Act_E = \{\leftrightarrow, \rightarrow, \leftarrow, -\}, \quad \text{for all } l_E \in L_E$$

(in this simple example where L_E is a singleton, just $P_E(\cdot) = Act_E$).

If we assume that the system starts from a state $g_0 = (0, \epsilon, \cdot)$ or $g_0 = (1, \epsilon, \cdot)$, it is possible to show that S will start sending the bit and will only stop after having received an acknowledgement from R. In turn R will remain silent until it receives the bit, and it will never stop sending acknowledgements from then on. The analysis can be made formally by using the mechanism of *contexts* [FHMV97]; for our purposes it is not necessary to pursue that analysis here.

The set of global states reachable from the initial configurations $(0, \epsilon, \cdot)$ and $(1, \epsilon, \cdot)$ as defined by the transition function π and the protocol functions P_S, P_R and P_E, is summarised in Figure 1. The environment component is omitted for clarity.

Having defined the set of reachable global states, we can apply the tools of formal logic to the analysis of the scenario by considering an interpretation of a suitably chosen set of propositional variables. We shall use the set $P = \{\mathbf{bit = 0}, \mathbf{bit = 1}, \mathbf{recbit}, \mathbf{recack}\}$. We interpret these on the interpreted system $IS_{\mathrm{b}} = (G, \sim_S, \sim_R, h)$, where G is the set of *reachable global states* as defined by π, P_S, P_R and P_E, where $\sim_R$ and $\sim_S$ are equivalence relations on global states as defined in [FHMV95], and where h is an interpretation for the atoms in P such that the following holds:

$$\begin{array}{lll}
(IS_{\mathrm{b}}, g) \models \mathbf{bit = 0} & \text{if} & l_S(g) = 0, \text{ or } l_S(g) = (0, ack) \\
(IS_{\mathrm{b}}, g) \models \mathbf{bit = 1} & \text{if} & l_S(g) = 1, \text{ or } l_S(g) = (1, ack) \\
(IS_{\mathrm{b}}, g) \models \mathbf{recbit} & \text{if} & l_R(g) = 1, \text{ or } l_R(g) = 0 \\
(IS_{\mathrm{b}}, g) \models \mathbf{recack} & \text{if} & l_S(g) = (1, ack), \text{ or } l_S(g) = (0, ack).
\end{array}$$

This permits us to represent and check properties of the system directly on the semantical models. For example, by employing standard temporal connectives on the runs constructed above one can check that:

$$IS_{\mathrm{b}} \not\models \mathbf{recbit} \rightarrow \mathrm{G}_{\forall}\, \mathbf{recack}$$

($\mathrm{G}_{\forall}$ here means truth in some global state in all future branching runs) which represents the intrinsic unreliability of the channel.

Irrespective of the analysis of the dynamic properties of the system, there is one interesting static epistemic property that is worth observing. By ascribing knowledge to the agents using the standard [FHMV95] approach, it can be checked from Figure 1 that

$$IS_{\mathrm{b}} \models \mathbf{recbit} \rightarrow \big(K_R\,(\mathbf{bit = 0}) \vee K_R\,(\mathbf{bit = 1})\big) \tag{1}$$

which confirms our intuition about the model. Furthermore:

$$IS_{\mathrm{b}} \models \mathbf{recack} \rightarrow \mathbf{recbit} \tag{2}$$

$$IS_{\mathrm{b}} \models \mathbf{recack} \rightarrow \big(K_R\,(\mathbf{bit = 0}) \vee K_R\,(\mathbf{bit = 1})\big) \tag{3}$$

and perhaps most interestingly that

$$IS_{\mathrm{b}} \models \mathbf{recack} \rightarrow K_S\big(K_R\,(\mathbf{bit = 0}) \vee K_R\,(\mathbf{bit = 1})\big) \tag{4}$$

$$IS_{\mathrm{b}} \models \mathbf{recack} \wedge (\mathbf{bit = 0}) \rightarrow K_S\,K_R\,(\mathbf{bit = 0}) \tag{5}$$

(and similarly for the case $(\mathbf{bit = 1})$). So, if an *ack* is received by S, then S is sure that R knows the value of the bit. Intuitively this represents the fact that although the channel is potentially faulty, if messages do manage to travel back and forth the protocol is strong enough to eliminate any uncertainty in the communication. Whereas, for example,

$$IS_{\mathrm{b}} \not\models \mathbf{recack} \wedge (\mathbf{bit = 0}) \rightarrow K_R\,K_S\,K_R\,(\mathbf{bit = 0})$$

The properties (1–5) above can be checked by direct calculation on the table in Figure 1. Properties (3), (4) and (5) can also be derived very straightforwardly from (1) and (2) in the logic $\mathrm{S5}_n$. (3) follows from (1) and (2) by propositional logic. (4) follows from (3): indeed, suppose $IS_{\mathrm{b}} \models \mathbf{recack} \rightarrow \varphi$ for any formula φ. Then, since K_S is normal, we have both $IS_{\mathrm{b}} \models K_S(\mathbf{recack} \rightarrow \varphi)$ and $IS_{\mathrm{b}} \models K_S\,\mathbf{recack} \rightarrow K_S\,\varphi$. Now check from Figure 1 that $IS_{\mathrm{b}} \models \mathbf{recack} \rightarrow K_S\,\mathbf{recack}$ (and hence $IS_{\mathrm{b}} \models \mathbf{recack} \leftrightarrow K_S\,\mathbf{recack}$). The property $IS_{\mathrm{b}} \models \mathbf{recack} \rightarrow K_S\,\varphi$ follows immediately, and (4) is a special case. Property (5) can be derived in similar fashion. Again, the form of these derivations surely accords with our intuition about the example. The derivations will be useful, moreover, when we examine more complicated examples later.

5 Violation of specifications

In the previous section we have assumed that both agents and environment follow the protocols. We now apply the machinery of deontic interpreted systems [LS01a, LS01b] to analyse what happens when the specified protocols are violated in the bit-transmission problem. Because of space limitations, we examine in detail only the possibility that R is faulty. The possibility that S is faulty, and other combinations of faulty R, S and E, can be treated in similar fashion. Specifically, we shall consider in this section the possibility that R may send acknowledgements without having received the bit. This is a simple example of an agent doing something that it should not do. In the section that follows we shall consider an example where an agent does not do something that it should do.

In order to apply the machinery we modify the framework of the previous section so that the set of local states of agent i is composed of two disjoint sets of green (G_i) and red (R_i) local states, representing correct and incorrect functioning behaviour respectively. For S, since we are not admitting (for the purposes of the example) the possibility of faults, its local states are all green, that is to say, allowed by the protocol. We thus have:

$$L_S = G_S = \{0,\, 1,\, (0, ack),\, (1, ack)\}, \quad R_S = \emptyset.$$

For the case of the environment, we have admitted the possibility of faulty, or unreliable, behaviour but these 'faults' are not violations of the protocol under examination. Accordingly, all local states of the environment are also green, and we have:

$$L_E = G_E, \quad R_E = \emptyset.$$

It remains to model the local states of the, now potentially faulty, receiver R. Clearly we need to extend the set of local states $\{0, 1, \epsilon\}$ with an additional element (ϵ, ack_f) representing the (red) local state in which R has sent an acknowledgement without having received the value of the bit. However, we also want to consider local states in which R sent an acknowledgement in violation of the protocol but has received the value of the bit in the meantime. These are also 'red' local states. Accordingly, we define the local states of R as follows:

$$G'_R = \{0, 1, \epsilon\}, \quad R'_R = \{(0, ack_f),\ (1, ack_f),\ (\epsilon, ack_f)\}, \quad L'_R = G'_R \cup R'_R.$$

Here, ack_f in the 'red' states R'_R is used to indicate that at least one faulty acknowledgement was sent at some time before the value of the bit had been received[2].

We turn now to defining the protocol functions P'_R, P'_S, P'_E of this deontic interpreted system. Given that the two sets of local states for S and E have not changed we can keep the function P_S and P_E described in Section 4, i.e., we take $P'_S = P_S$ and $P'_E = P_E$. But we need to extend P_R so that it is defined also on the red local states of R. We want to define what we might call a 'monotonic' extension of the protocol function P_R: the protocol P'_R should be the same as P_R when evaluated on the green local states for R and it remains only to define its values for the red local states. The projection of the new interpreted system onto the green local states should result in the 'old' system IS_b of Section 4. So we have:

$$P'_R(\epsilon) = P_R(\epsilon) = \lambda, \quad P'_R(0) = P'_R(1) = P_R(0) = P_R(1) = sendack$$

How shall we define P'_R for the red local states $R_R = \{(0, ack_f), (1, ack_f), (\epsilon, ack_f)\}$? Of course, the protocol described in Section 3 *does not say*: as presented, it does not cover the possibility of violation, and does not specify what actions are to be taken if errors arise. For the sake of concreteness, let us define

$$P'_R((0, ack_f)) = P'_R((1, ack_f)) = P'_R((\epsilon, ack_f)) = Act_R = \{sendack, \lambda\}$$

In fact, in this example, it makes little difference how we define P'_R for red local states. As is perhaps obvious, there is no extension of the protocol that will recover effectively from the erroneous sending of an acknowledgement by R. The point, however, is that in general it is meaningful to define protocols on red states as well as green, whenever the protocol makes provision for remedial or error recovery actions.

[2] In employing this device we are effectively coding part of the history of the execution into the (red) local states for R. This allows us to avoid the complication of adding temporal constructs to the framework. We are investigating temporal extensions to deontic interpreted systems but will not discuss these further here. The non-temporal version is adequate for the purposes of the examples in this paper.

One question that we would like to ask is how the runs of the system change following the introduction of red local states for R. Suppose the system starts as before from the global state $g_0 = (0, \epsilon, \cdot)$ or $g_0 = (1, \epsilon, \cdot)$. It can either produce an error-free run or R can act faultily at any time during the evolution. R's faults may indeed inhibit the communication of the bit. For example, if at any point R sends an *ack* without having received the bit, this, if received by S, will make S stop sending messages, preventing any communication between the agents. This simple observation indicates that some of the properties that hold in the case of no incorrect behaviour will no longer be valid here. This should be reflected in the analysis.

Let us explore then how the analysis turns out by applying the same multi-modal language based on the set of atoms $P = \{\mathbf{bit = 0}, \mathbf{bit = 1}, \mathbf{recbit}, \mathbf{recack}\}$, and augmented by the modal operators $O_i, K_i, \widehat{K}_i^j$ described in Section 2. We interpret formulas on the deontic interpreted system $IS'_{\mathrm{b}} = (G', \sim'_R, \sim'_S, R_R^{O'}, R_S^{O'}, h')$ resulting from defining the relations $\sim'_i, R_i^{O'}, i \in \{S, R\}$ on the set of the reachable global states $G' \subset L_S \times L'_R \times L_E$.

For the computation of reachable global states G', it remains to define the evolution function π' for system IS'_{b}. Essentially we would like to extend the definition of π in system IS_{b} by insisting that R's local states will be red if R has sent an acknowledgement, either in the current round or in the past, without having received the bit first, and otherwise copy R's transitions in IS_{b}, i.e., R will correctly store the bit if it has received it and remain in the state ϵ otherwise. First, we specify the effects of protocol-violating actions in green R states. For the case where the bit is 0 (the other can be done similarly) we shall impose:

$$\begin{aligned}
\pi'(\,(0, \epsilon, \cdot), (sendbit, sendack, \leftrightarrow)\,) &= (\,(0, ack), (0, ack_f), \cdot\,) \\
\pi'(\,(0, \epsilon, \cdot), (sendbit, sendack, \rightarrow)\,) &= (\,0, (0, ack_f), \cdot\,) \\
\pi'(\,(0, \epsilon, \cdot), (sendbit, sendack, \leftarrow)\,) &= (\,(0, ack), (\epsilon, ack_f), \cdot\,) \\
\pi'(\,(0, \epsilon, \cdot), (sendbit, sendack, -)\,) &= (\,0, (\epsilon, ack_f), \cdot\,)
\end{aligned}$$

These definitions can be expressed more concisely but we present them in this long-winded style for easier reading. Note that in the first case above the result state is a faulty (red) state even though communication has taken place, and that in the second and fourth cases the result state is a faulty (red) state even though the erroneous acknowledgement was not received by S.

Now we extend the definition of π so that π' is defined also on red states. Once R is in a red state we will impose that it will remain in a red state, although it will correctly store messages, if received.

$$\begin{aligned}
\pi'(\,(0, (\epsilon, ack_f)), \cdot), (sendbit, sendack, \leftrightarrow)\,) &= (\,(0, ack), (0, ack_f), \cdot\,) \\
\pi'(\,(0, (\epsilon, ack_f)), \cdot), (sendbit, sendack, \rightarrow)\,) &= (\,0, (0, ack_f), \cdot\,) \\
\pi'(\,(0, (\epsilon, ack_f)), \cdot), (sendbit, sendack, \leftarrow)\,) &= (\,(0, ack), (\epsilon, ack_f), \cdot\,) \\
\pi'(\,(0, (\epsilon, ack_f)), \cdot), (sendbit, sendack, -)\,) &= (\,0, (\epsilon, ack_f), \cdot\,)
\end{aligned}$$

Similarly, for further illustration:

$$\pi'(\,(0, (0, ack_f)), \cdot), (sendbit, sendack, \gamma_E)\,) = (\,l'_S, l'_R, \cdot\,)$$

where $l'_S = (0, ack)$, $l'_R = (0, ack_f)$ for $\gamma_E = \leftrightarrow$ or $\gamma_E = \leftarrow$; and $l'_S = 0$, $l'_R = (0, ack_f)$ for $\gamma_E = \rightarrow$ or $\gamma_E = -$.

$(0,(0,ack_f))$		$((0,ack),(0,ack_f))$	
	$(1,(1,ack_f))$		$((1,ack),\ (1,ack_f))$
$(0,(\epsilon,ack_f))$	$(1,(\epsilon,ack_f))$	$((0,ack),(\epsilon,ack_f))$	$((1,ack),(\epsilon,ack_f))$
$(0,0)$		$((0,ack),0)$	
	$(1,1)$		$((1,ack),1)$
$(0,\epsilon)$	$(1,\epsilon)$		

Figure 2: The state space of the bit-transmission system in case R may send incorrect acknowledgements. Columns (respectively rows) represent global states epistemically equivalent for S (respectively for R).

Other cases for the remaining red states are expressed similarly and are left to the reader. For the rest of the system we define π' to behave in the same way as π in IS_{b}, i.e., the projection of π' onto the components of S and E coincides with π.

Given these definitions we can compute the set of reachable states as before. They are shown in Figure 2, again omitting the environment component for clarity. The interpretation h' for the new system is the interpretation h from the previous section with the interpretation of **recbit** updated, thus:

$$(IS'_{\mathrm{b}}, g) \models \mathbf{recbit} \text{ if } \quad \begin{array}{l} l_R(g)=1, \text{ or } l_R(g)=0, \text{ or} \\ l_R(g)=(0,ack_f), \text{ or } l_R(g)=(1,ack_f). \end{array}$$

We can now investigate whether or not the formulas that held true in the scenario with no fault remain true here. It is easily checked that

$$IS'_{\mathrm{b}} \models \mathbf{recbit} \rightarrow \big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big) \tag{6}$$

which confirms our intuition about the model. Even if faults occur, if the bit has been received, R will know its value. It is the mechanism of acknowledgements that is no longer reliable. Indeed we find in our model that:

$$\begin{array}{lll} IS'_{\mathrm{b}} & \not\models & \mathbf{recack} \rightarrow \mathbf{recbit} \\ IS'_{\mathrm{b}} & \not\models & \mathbf{recack} \rightarrow \big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big) \end{array}$$

and as expected:

$$\begin{array}{lll} IS'_{\mathrm{b}} & \not\models & \mathbf{recack} \rightarrow K_S\big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big) \\ IS'_{\mathrm{b}} & \not\models & \mathbf{recack} \wedge (\mathbf{bit}=\mathbf{0}) \rightarrow K_S\,K_R\,(\mathbf{bit}=\mathbf{0}) \end{array}$$

Notice however that using the operator O_R introduced in Section 2, which represents what holds in states where R is operating correctly, we have the following:

$$IS'_{\mathrm{b}} \models O_R(\,\mathbf{recack} \rightarrow \mathbf{recbit}\,) \tag{7}$$

$$IS'_{\mathrm{b}} \models O_R(\,\mathbf{recack} \rightarrow \big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big)\,) \tag{8}$$

More interesting is that a particular form of knowledge also still holds. Intuitively if S makes the assumption of R's correct functioning behaviour, then, upon receipt of an acknowledgement, it would make sense for it to assume that

R does know the value of the bit. To model this intuition we use the operator $\widehat{K}_i^j$ "knowledge under the assumption of correct behaviour" as presented in Section 2. This describes the knowledge agent i has if attention is restricted to states in which j is performing as intended. We refer to [LS01b] for more details, but note that unlike the usual epistemic operators associated with interpreted systems, $\widehat{K}_i^j$ is not an S5 operator, and in particular it does not have axiom T, i.e., knowledge under assumptions of correct functioning behaviour does not imply truth. This operator is of particular interest here because it captures precisely our intuition about the example. We have:

$$IS'_{\mathrm{b}} \models \mathbf{recack} \rightarrow \widehat{K}_S^R\big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big) \tag{9}$$

$$IS'_{\mathrm{b}} \models \mathbf{recack} \wedge (\mathbf{bit}=\mathbf{0}) \rightarrow \widehat{K}_S^R\,K_R\,(\mathbf{bit}=\mathbf{0}) \tag{10}$$

Again, as in Section 4, we can check these properties by calculation from the table in Figure 2, and we can derive them syntactically from properties simpler to evaluate. Check first from Figure 2 that properties (6) and (7) hold. These are the key properties. ¿From (6) and (7), and O_R normal, we easily derive (8). To derive (9) and (10) we modify the argument used in Section 4 allowing now for the presence of O_R. Suppose that the following generalised form of property (8) is valid in the model: $IS'_{\mathrm{b}} \models O_R(\,\mathbf{recack} \rightarrow \varphi\,)$. Since K_S is normal, we get $IS'_{\mathrm{b}} \models K_S\,O_R(\,\mathbf{recack} \rightarrow \varphi\,)$. The general property $K_S\,O_R\,\varphi \rightarrow \widehat{K}_S^R\,\varphi$ is valid in the class of all deontic interpreted systems (see the summary of the logic in Section 2 and [LS01a] for details), so we have $IS'_{\mathrm{b}} \models \widehat{K}_S^R(\,\mathbf{recack} \rightarrow \varphi\,)$, and hence also $IS'_{\mathrm{b}} \models \widehat{K}_S^R\,\mathbf{recack} \rightarrow \widehat{K}_S^R\,\varphi$ because $\widehat{K}_S^R$ is normal. Now $IS'_{\mathrm{b}} \models \mathbf{recack} \rightarrow K_S\,\mathbf{recack}$ still holds, as is easily checked from Figure 2, and the general property $K_S\,\varphi \rightarrow \widehat{K}_S^R\,\varphi$ (see Section 2) gives $IS'_{\mathrm{b}} \models \mathbf{recack} \rightarrow \widehat{K}_S^R\,\mathbf{recack}$. So we get $IS'_{\mathrm{b}} \models \mathbf{recack} \rightarrow \widehat{K}_S^R\,\varphi$, of which (9) is a special case. The derivation of property (10) can be obtained in similar fashion.

To summarise: we have modified the scenario of the bit-transmission problem, relaxing slightly the assumptions of correct functioning behaviour that hold true for it. In particular, we have allowed for the possibility that one of the agents, the receiver R, may violate the protocol by performing an action it should not perform according to the protocol. We have seen that some key properties of the system then no longer hold. In particular under the assumptions we have studied, S will never know whether or not R knows the value of the bit; at best S can know this only under the assumption of correct functioning behaviour for R[3]. This is an intuitive result that was validated semantically on the model, and derived syntactically from some simple key properties of the model.

In specifying the extended protocol, the value of P'_R for the red local states of R was chosen arbitrarily. It seems intuitively obvious that no other specification of P'_R on the red states would allow us to recover the key property we require

[3]In the case where the communication channel operates symmetrically in both directions, i.e., in the case where either the channel loses messages in both directions or both messages get through, the sender S knows that receiver R knows the value of the bit when S receives an acknowledgement, even without assuming correct behaviour by R. This is because if S receives an acknowledgement, its transmission of the bit must also have got through to R. The reader may care to re-work the analysis for the simpler case where $Act_E = \{\leftrightarrow, -\}$ to check that the formal model confirms this informal argument.

of the protocol. In other words, it should be possible to show that **recack** $\wedge$ $(\mathbf{bit} = \mathbf{0}) \rightarrow K_S K_R (\mathbf{bit} = \mathbf{0})$ is not valid in IS'_{b} no matter how we extend the definition of P_R to cover the red local states of R—there is no protocol that will ensure recovery of communication once a violation has occurred.

How could the system recover after R has sent an incorrect *ack*? It seems as though the only way to proceed is for the two agents to re-synchronise, perhaps with a message from R signalling the failure of all communication so far with a request to restart the protocol from scratch. Clearly this request could *per se* fail to reach S, so we would need to require S to acknowledge the request to R. So the roles of S and R would need to be swapped before communication can be resumed. We see no conceptual difficulty in modelling this setting with the tools presented so far.

6 An error correcting protocol

Consider again the bit-transmission problem as modelled in Section 4, but assume now that R can be faulty in a different way, in that it may fail to send acknowledgements when in fact it has received the bit. To what extent would this second kind of fault compromise communication, and are there ways of recovering from it? In line with the development of the previous section, let us define the elements of the model for this new setting: sets of green and red local states, protocols, the transition function, and reachable global states.

The set of local red and green states for S and E are the same as in the models of Sections 4 and 5. The local states for R are as follows:

$$G''_R = \{0, 1, \epsilon\}, \quad R''_R = \{(0, f), (1, f)\}, \quad L''_R = G''_R \cup R''_R.$$

Here f in the red local states indicates that the bit has been received but R has failed (at least once) to send an acknowledgement.

The protocol functions we use for S and E are again the ones we employed in Sections 4 and 5; what changes is the protocol function for agent R. Again, we wish to define an extension P''_R of P_R that has the same values as P_R for the green local states and specifies in addition the results of protocol-violating actions by R in green local states and how R should behave when in a red local state. In this example, unlike the one of Section 5, there is an obvious way of extending the protocol so that we obtain error-correcting behaviour in case a fault has occurred: if R has failed to send an acknowledgement, we simply require that R does so at the next round. Formally:

$$P''_R(\epsilon) = \lambda, \quad P''_R(0) = P''_R(1) = sendack, \quad P''_R((0, f)) = P''_R((1, f)) = sendack.$$

So in this case, it is easy to spell out the conditions for recovery from a red state. To check that recovery is indeed obtained we evaluate formulas on the evolution of the deontic interpreted system just constructed. It remains to define the transition function of the system, so that we can compute the set of reachable global states.

As usual, consider first the conditions under which we move to a red local state for agent R, and then the outcome of transitions originating from red local states for agent R (the definitions for bit=1 are analogous). For all $\gamma_E \in Act_E$,

$(0,(0,f))$		$((0,ack),(0,f))$	
	$(1,(1,f))$		$((1,ack),(1,f))$
$(0,0)$		$((0,ack),0)$	
	$(1,1)$		$((1,ack),1)$
$(0,\epsilon)$	$(1,\epsilon)$		

Figure 3: The state space of the bit-transmission system in case R may fail to send acknowledgements when supposed to do so. Columns (respectively rows) represent global states epistemically equivalent for S (respectively for R).

unless stated otherwise:

$$\begin{array}{rcl} \pi''(\,(0,0,\cdot),(sendbit,\lambda,\gamma_E)\,) & = & (0,(0,f),\cdot) \\ \pi''(\,((0,ack),0,\cdot),(\lambda,\lambda,\gamma_E)\,) & = & ((0,ack),(0,f),\cdot) \end{array}$$

$$\begin{array}{rcll} \pi''(\,(0,(0,f),\cdot),(sendbit,\lambda,\gamma_E)\,) & = & (0,(0,f),\cdot) & \\ \pi''(\,(0,(0,f),\cdot),(sendbit,sendack,\gamma_E)\,) & = & ((0,ack),0,\cdot) & (\gamma_E \in \{\leftrightarrow,\leftarrow\}) \\ \pi''(\,(0,(0,f),\cdot),(sendbit,sendack,\gamma_E)\,) & = & (0,0,\cdot) & (\gamma_E \in \{\rightarrow,-\}) \end{array}$$

Note that in the last two cases we have chosen to say that sending an acknowledgement in a red local state puts R back into a green local state—the record of the protocol-violating fault is wiped out. For completeness:

$$\begin{array}{rcl} \pi''(\,((0,ack),(0,f),\cdot),(\lambda,\lambda,\gamma_E)\,) & = & ((0,ack),(0,f),\cdot) \\ \pi''(\,((0,ack),(0,f),\cdot),(\lambda,sendack,\gamma_E)\,) & = & ((0,ack),0,\cdot) \end{array}$$

With this information, we can compute the set of reachable global states. Once again we do not show the entire computation here, but simply summarise the results in Figure 3, with the environment component omitted as usual.

Following what is by now our standard procedure we can determine the corresponding deontic interpreted system IS'' and interpret the formulas of interest. The interpretation function h'' for atoms is as before but with a tiny adjustment: h'' is h with the obvious extension to make atom **recbit** true also in all global states g with $l_R(g) = (0,f)$ or $l_R(g) = (1,f)$. It is easily confirmed that we have all of:

$$\begin{array}{rcl} IS''_{\mathrm{b}} & \models & \mathbf{recack} \rightarrow \big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big) \\ IS''_{\mathrm{b}} & \models & \mathbf{recack} \rightarrow K_S\big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big) \\ IS''_{\mathrm{b}} & \models & \mathbf{recack} \wedge (\mathbf{bit}=\mathbf{0}) \rightarrow K_S\,K_R\,(\mathbf{bit}=\mathbf{0}) \end{array}$$

Indeed, these follow immediately by checking in Figure 3 that

$$\begin{array}{rcl} IS''_{\mathrm{b}} & \models & \mathbf{recbit} \rightarrow \big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big) \\ IS''_{\mathrm{b}} & \models & \mathbf{recack} \rightarrow \mathbf{recbit} \end{array}$$

The rest then follows by the same syntactic derivation as in Section 4.

Naturally, assuming correctness of R's behaviour does not invalidate any of the above, so that we have also, e.g.:

$$IS''_{\mathrm{b}} \;\models\; \mathbf{recack} \wedge (\mathbf{bit}=\mathbf{0}) \rightarrow \widehat{K}^R_S\,K_R\,(\mathbf{bit}=\mathbf{0})$$

Indeed, $K_S \varphi \to \widehat{K}_S^R \varphi$ is valid in the class of all deontic interpreted systems: $K_i \varphi \to \widehat{K}_i^j \varphi$ is a property of the logic for all pairs of agents i and j (see [LS01a, LS01b] for details).

This example demonstrates how the formalism of deontic interpreted systems deals with two rather intuitive points. First, not all incorrect behaviours by the participating agents necessarily compromise the validity of crucial properties of the system being modelled. Secondly, it is possible and useful to reason about these incorrect error states and devise protocols that attempt recovery from them.

7 Control

In the previous section it was possible to devise a simple error-correcting protocol that, if followed, ensures that the system will get back to a state in which all the agents are in a green local state. This is not always possible, of course. Given a system in which agents cannot be assumed to behave in accordance with the specified standards of behaviour or to follow the prescribed protocols, it is natural to seek additional control or enforcement mechanisms that can be introduced to encourage or even force agents to comply. Of these the simplest (but by no means the only) strategy is to look for a way of constraining the agents' behaviour so that the possibility of violation is simply eliminated. In the present example, for instance, we could add an additional filter on the transmission of messages from R to block out those that would cause violations of the protocol. If successful, we would have an example of an enforcement strategy called 'regimentation' in [JS93]. Discussion of other possible strategies is beyond the scope of this paper.

So let us consider another variation of the bit-transmission problem. This time we will assume that R may develop faults of *either* of the two kinds analysed in Sections 5 and 6. We also introduce a third agent C into the system, a *controller* whose function is to constrain the behaviour of R. The controller agent C is a kind of 'regimentation' device and, as we understand it, a (simple) example of what is called an 'inter-agent' in [RAMN$^+$98].

C's actions are the following:

$$Act_C = \{allow, block\}.$$

These actions provide an additional filter on the attempted actions of receiver R. The *block* action will be used to override any transmit action of the environment when R attempts to send an erroneous acknowledgement; *allow* allows a message from R to enter the transmission channel to S.

The controller C must be able to detect violations of the protocol by R. Now, it is a fundamental assumption of the formalism of interpreted systems that all local states are private and only actions are public. This it to model what actually happens in distributed and multi-agent systems. The local state of an agent i is invisible to a different agent j, unless there are actions by means of which agent i communicates selected elements of its local state to agent j. And likewise for the local state of the environment. Given this, the controller may block R's messages only on the basis of observations of the actions performed. For this example, the behaviour of C can be modelled by giving it two local states, ok and $\overline{ok}$, say. We shall not model (in this paper) the possibility that the controller C fails to act correctly, so both of its local states are green:

$$G_C = L_C = \{ok, \overline{ok}\}, \quad R_C = \emptyset.$$

We use ok for the state where C has not detected an attempted violation by R and so is allowing messages from R to enter the communication channel, and $\overline{ok}$ for the state where C has detected an attempted violation by R and is thus blocking messages from R. The protocol of C is simply:

$$P'''_C(ok) = allow, \quad P'''_C(\overline{ok}) = block.$$

It remains to specify how the controller's local state switches from ok to $\overline{ok}$ bearing in mind that the local state of R is invisible to C. That will be done when specifying the evolution function of the system.

The protocols for S and E are the same as those we have employed so far. As for R, we are now assuming it can develop both kinds of faults discussed in previous sections. So, in this version of the example its local states are $L'''_R = G'''_R \cup R'''_R$ where the green states $G'''_R = \{0, 1, \epsilon\}$ as usual, and the red states are the following:

$$R'''_R = \{(0, ack_f), (1, ack_f), (\epsilon, ack_f), (0, f), (1, f), (0, ack_f, f), (1, ack_f, f)\}$$

It is possible to devise a simpler representation of R's red local states but we will work with this larger one as it is clearer how it combines elements from the two previous analyses.

For the protocol function for R, we want P'''_R to be the same as P_R when evaluated on green local states, as usual, and to have the error-correcting behaviour discussed in Section 6 for the local states $(0, f)$ and $(1, f)$. For these two states, P'''_R has the value of P''_R. We also want this behaviour for the local states $(0, ack_f, f)$ and $(1, ack_f, f)$. For the remaining states $(0, ack_f)$, $(1, ack_f)$, (ϵ, ack_f), the value of P'''_R can be chosen arbitrarily since again nothing of interest will depend on this. For concreteness, we will follow the choice in Section 5 and define it to be $Act_R = \{sendack, \lambda\}$. So we have:

$$\begin{aligned}
&P'''_R(\epsilon) = \lambda, \quad P'''_R(0) = P'''_R(1) = sendack,\\
&P'''_R((0, f)) = P'''_R((1, f)) = P'''_R((0, ack_f, f)) = P'''_R((1, ack_f, f)) = sendack,\\
&P'''_R((0, ack_f)) = P'''_R((1, ack_f)) = P'''_R((\epsilon, ack_f)) = \{sendack, \lambda\}
\end{aligned}$$

Note that while C is expected to block erroneous $sendack$ messages when R is in the critically faulty states (those with ack_f), it cannot *prevent* R from entering these or any other faulty states.

We now consider the transition function of this system. In this case both global states and joint actions are 4-tuples. Let us first consider agent C. The controller may block R's messages only on the basis of observations of the joint actions performed. In this example, we get the desired effect by stipulating that C starts in state $\overline{ok}$ and remains in state $\overline{ok}$ until the successful transmission of a bit from S to R is observed, i.e., given our assumptions about the communication channel, until there occurs a $sendbit$ action by S when the environment action is either $\leftrightarrow$ or $\rightarrow$. When this happens, C switches to local state ok, and it remains in that state for the rest of the run.

It is clearer (and shorter) to define π''' in terms of another function π'''_+ which is a combination of the effects of π' and π'' defined in earlier sections, extended to cover the red local states for R with both faults ack_f and f. π'''_+ corresponds to the evolution function of the system without the presence of the controller agent C. We sketch its definition presently. The definition of π''' is (for all $\gamma_R \in Act_R$, $\gamma_S \in Act_S$, $\gamma_E \in Act_E$):

$$\begin{aligned}
\pi'''(\,(l_S, l_R, \overline{ok}, l_E), (sendbit, \gamma_R, block, -)\,) &= \pi'''_+(\,(l_S, l_R, l_C, l_E), (sendbit, \gamma_R, -)\,) : \overline{ok} \\
\pi'''(\,(l_S, l_R, \overline{ok}, l_E), (sendbit, \gamma_R, block, \leftarrow)\,) &= \pi'''_+(\,(l_S, l_R, l_C, l_E), (sendbit, \gamma_R, -)\,) : \overline{ok} \\
\pi'''(\,(l_S, l_R, \overline{ok}, l_E), (sendbit, \gamma_R, block, \rightarrow)\,) &= \pi'''_+(\,(l_S, l_R, l_C, l_E), (sendbit, \gamma_R, \rightarrow)\,) : ok \\
\pi'''(\,(l_S, l_R, \overline{ok}, l_E), (sendbit, \gamma_R, block, \leftrightarrow)\,) &= \pi'''_+(\,(l_S, l_R, l_C, l_E), (sendbit, \gamma_R, \leftarrow)\,) : ok \\
\pi'''(\,(l_S, l_R, ok, l_E), (\gamma_S, \gamma_R, allow, \gamma_E)\,) &= \pi'''_+(\,(l_S, l_R, l_C, l_E), (\gamma_S, \gamma_R, \gamma_E)\,) : ok
\end{aligned}$$

where $(l_S, l_R, l_E) : l_C$ represents the 4-tuple (l_S, l_R, l_C, l_E).

When C is in state $\overline{ok}$ it blocks all attempts to send messages by R. For example:

$$\begin{aligned}
\pi'''(\,(0, \epsilon, \overline{ok}, \cdot), (sendbit, sendack, block, \leftrightarrow)\,) &= (0, (0, ack_f), ok, \cdot) \\
\pi'''(\,(0, \epsilon, \overline{ok}, \cdot), (sendbit, sendack, block, \leftarrow)\,) &= (0, (\epsilon, ack_f), \overline{ok}, \cdot)
\end{aligned}$$

Notice that the first of the above is a case where C observes the successful arrival of a bit message at R and so is a case where C's local state switches from $\overline{ok}$ to ok. Indeed, all such instances are already subsumed under the general definition of π''' above. Note also that only R's messages can be blocked by C.

It remains to define the function π'''_+. We wish to combine the effects of π' and π'' from previous sections. So, for the case where the local state l_R of R is green and the action γ_R of R belongs to $P_R(l_R)$, the value of π'''_+ is the value of π. For the case where $l_R \in L'_R$ and $\gamma_R \in P'_R(l_R)$ (which by construction subsumes the previous 'green' case also) the value of π'''_+ is the value of π'. And for the case where $l_R \in L''_R$ and $\gamma_R \in P''_R(l_R)$ (which also subsumes the 'green' case) the value of π'''_+ is the value of π''. The remaining cases are those dealing with the new red local states. They are as follows (with the corresponding entry for bit=1):

$$\begin{aligned}
\pi'''_+(\,(0, (0, ack_f), \cdot), (sendbit, \lambda, \gamma_E)\,) &= (0, (0, ack_f, f), \cdot) \\
\pi'''_+(\,((0, ack), (0, ack_f), \cdot), (\lambda, \lambda, \gamma_E)\,) &= ((0, ack), (0, ack_f, f), \cdot) \\
\pi'''_+(\,(0, (0, ack_f, f), \cdot), (sendbit, \lambda, \gamma_E)\,) &= (0, (0, ack_f, f), \cdot) \\
\pi'''_+(\,(0, (0, ack_f, f), \cdot), (sendbit, \lambda, \gamma_E)\,) &= ((0, ack), (0, ack_f, f), \cdot), \quad (\gamma_E \in \{\leftrightarrow, \leftarrow\}) \\
\pi'''_+(\,(0, (0, ack_f, f), \cdot), (sendbit, \lambda, \gamma_E)\,) &= (0, (0, ack_f, f), \cdot), \quad (\gamma_E \in \{\rightarrow, -\}) \\
\pi'''_+(\,((0, ack), (0, ack_f, f), \cdot), (\lambda, sendack, \gamma_E)\,) &= ((0, ack), (0, ack_f), \cdot) \\
\pi'''_+(\,((0, ack), (0, ack_f, f), \cdot), (\lambda, \lambda, \gamma_E)\,) &= ((0, ack), (0, ack_f, f), \cdot)
\end{aligned}$$

We may now move to consider the deontic interpreted system $IS'''_{\mathrm{b}} = (G''', \sim'''_S, \sim'''_R, \sim'''_C, R^{O'''}_S, R^{O'''}_R, R^O_C, h''')$ generated by the system above. The set of global states $G''' \subset L_S \times \overline{L'''_R} \times L_C \times L_E$ can be computed from the initial states $(0, \epsilon, \overline{ok}, \cdot)$ and $(1, \epsilon, \overline{ok}, \cdot)$ from the transition function π''' described above. The resulting set of reachable global states is summarised in Figure 4. Given that the local states of C do not affect the epistemic states of S and R, which is our

$(0,(0,ack_f,f))$		$((0,ack),(0,ack_f,f))$	
	$(1,(1,ack_f,f))$		$((1,ack),(1,ack_f,f))$
$(0,(0,f))$		$((0,ack),(0,f))$	
	$(1,(1,f))$		$((1,ack),(1,f))$
$(0,(0,ack_f))$		$((0,ack),(0,ack_f))$	
	$(1,(1,ack_f))$		$((1,ack),(1,ack_f))$
$(0,(\epsilon,ack_f))$	$(1,(\epsilon,ack_f))$	—	—
$(0,0)$		$((0,ack),0)$	
	$(1,1)$		$((1,ack),1)$
$(0,\epsilon)$	$(1,\epsilon)$		

Figure 4: The state space of the 'regimented' bit-transmission system in case R may both fail to send acknowledgements when supposed to, and send acknowledgements when supposed not to. Columns (respectively rows) represent global states epistemically equivalent for S (respectively for R). — indicates the states eliminated by introduction of the controller C.

prime object of interest in this analysis, we leave out C's component from the table as we have done for E earlier[4]. The table also shows which global states are eliminated (made unreachable from the initial states) by introduction of the regimentation mechanism C. Finally, we need to adjust the interpretation function h''' in the obvious way so that the atom **recbit** is true in all global states g except those with $l_R(g) = \epsilon$ or $l_R(g) = (\epsilon, ack_f)$.

Let us now go back to the formulas we analysed before and check whether they hold true on IS'''_{b}. We should expect that the introduction of the controller C eliminates the possibility of incorrect acknowledgements reaching S. This is indeed what we find by analysing the formulas. It is easy to check from Figure 4 that:

$$\begin{array}{lcl} IS'''_{\mathrm{b}} & \models & \mathbf{recbit} \rightarrow \big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big) \\ IS'''_{\mathrm{b}} & \models & \mathbf{recack} \rightarrow \mathbf{recbit} \end{array}$$

from which follows, by exactly the same syntactic derivations as in Section 4:

$$\begin{array}{lcl} IS'''_{\mathrm{b}} & \models & \mathbf{recack} \rightarrow \big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big) \\ IS'''_{\mathrm{b}} & \models & \mathbf{recack} \rightarrow K_S\big(K_R\,(\mathbf{bit}=\mathbf{0}) \vee K_R\,(\mathbf{bit}=\mathbf{1})\big) \\ IS'''_{\mathrm{b}} & \models & \mathbf{recack} \wedge (\mathbf{bit}=\mathbf{0}) \rightarrow K_S\,K_R\,(\mathbf{bit}=\mathbf{0}) \end{array}$$

These can also be confirmed by direct calculation from the table in Figure 4.

8 Conclusions

We have presented three variations of the bit-transmission problem to illustrate, and evaluate, how the machinery of deontic interpreted systems provides a means of analysing violations and (certain) enforcement mechanisms. Clearly

[4]More precisely, the table represents the quotient set of the set of reachable states with respect to an equivalence relation defined by $(l_S, l_R, l_C, l_E) \sim (l'_S, l'_R, l'_C, l'_E)$ if $l_S = l'_S$ and $l_R = l'_R$.

the example is trivial compared to the complex multi-agent systems applications referred to in the introductory section. Nevertheless, it does exhibit many of the features we shall have to confront in these complex examples. The fact that we have been able to carry out a detailed formal analysis of different kinds of faults, and of the effects of enforcement and control mechanisms, encourages us to believe that useful tools and methods can be developed on this basis.

Apart from examining further examples, we are pursuing three main lines of development. First, we have identified a number of technical questions concerning what we called 'monotonic' extensions of protocol functions to include red states. Generally, we can find ways of structuring the definition of protocol functions and system evolution functions to make them easier to construct and maintain as the applications become more complex. Space limitations prevented us from discussing alternative styles of definitions in this paper. Second, we are investigating an extended formalism which colours *transitions* red or green and not just states, combining the formalism of interpreted systems with the construction of a dynamic logic of permission reported in [Mey96]. Third, and perhaps most important, is the question of how these methods will scale up to deal with realistic examples with many agents and many kinds of faults.

The global reachable states shown in the figures of this paper were computed by hand. We have managed to complete a similar exercise for a more complicated version of the problem where the sender S may also violate its operating protocol. This is about the limit of what could be undertaken with confidence without some kind of computational assistance. Moreover, it is fortunate that in the bit transmission problem it is possible to derive the properties of interest syntactically from very simple properties that are easy to check semantically on the model. This will not always be the case. We do believe, however, that computational support can be provided to enable the analysis of large, realistic problems. We have been experimenting with model checking software for this purpose. Specifically, we use the NuSMV temporal model checker [CCGR99] to compute the set of runs of the system. From these we extract the set of reachable global states, and feed them into Akka[5], a software system for testing the validity of multimodal formulas in a Kripke model, to verify the epistemic properties of interest. Preliminary experiments using this method are reported in [LRS02]. We are also experimenting with the use of action description formalisms of the kind found in AI to make the definition of protocol and system evolution functions more concise.

Acknowledgements

This work was supported by the EU-funded FET project *ALFEBIITE* (IST-1999-10298).

References

[CCGR99] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model verifier. *Lecture Notes in Computer Science*, 1633, 1999.

[CJ96] José Carmo and Andrew J. I. Jones. Deontic database constraints, violation and recovery. *Studia Logica*, 57(1):139–165, 1996.

[5] `http://turing.wins.uva.nl/Ĩhendrik/`

[Coe93] J. Coenen. Top-down development of layered fault-tolerant systems and its problems. *Annals of Mathematics and Artificial Intelligence*, 9:133–150, 1993.

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.

[FHMV97] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. Knowledge-based programs. *Distributed Computing*, 10(4):199–225, 1997.

[FM91] J. Fiadeiro and T. S. E. Maibaum. Temporal reasoning over deontic specifications. *Journal of Logic and Computation*, 1(3):357–395, 1991.

[GHR93] D. M. Gabbay, I. M. Hodkinson, and M. A. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects, Volume 1:Mathematical Foundations*. Oxford University Press, 1993.

[GMP92] Janice Glasgow, Glenn MacEwen, and Prakash Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3):226–264, August 1992.

[Hin62] J. Hintikka. *Knowledge and Belief, an introduction to the logic of the two notions*. Cornell University Press, Ithaca (NY) and London, 1962.

[HZ92] J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: Knowledge-based derivations and correctness proofs for a family of protocols. *Journal of the ACM*, 39(3):449–478, 1992.

[JS93] A. J. I. Jones and M. J. Sergot. On the Characterisation of Law and Computer Systems: The Normative Systems Perspective. In John-Jules Ch. Meyer and Roel J. Wieringa, editors, *Deontic Logic in Computer Science: Normative System Specification*, chapter 12, pages 275–307. John Wiley & Sons, Chichester, England, 1993.

[KM89] S. Khosla and T. Maibaum. The prescription and description of state-based systems. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, number 398 in LNCS, pages 243–294. Springer-Verlag, 1989.

[LRS02] A. Lomuscio, F. Raimondi, and M. Sergot. Towards model checking interpreted systems. Submitted, 2002.

[LS01a] A. Lomuscio and M. Sergot. Extending interpreting systems with some deontic concepts. In J. van Benthem, editor, *Proceedings of TARK 2001*, pages 207—218, San Francisco, CA, July 2001. Morgan Kauffman.

[LS01b] A. Lomuscio and M. Sergot. On multi-agent systems specification via deontic logic. In J.-J Meyer, editor, *Proceedings of ATAL 2001*. Springer Verlag, July 2001. To Appear.

[Mey88] John-Jules Ch. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1):109–136, 1988.

[Mey96] R. van der Meyden. The dynamic logic of permission. *Journal of Logic and Computation*, 6(3):465–479, 1996.

[MH95] J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*, volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambdridge University Press, 1995.

[ML85] N. H. Minsky and A. Lockman. Ensuring integrity by adding obligations to privileges. In *Proc. 8th International Conf. on Software Engineering*, pages 92–102, August 1985.

[RAMN+98] J. A. Rodriguez-Aguilar, F. J. Martin, P. Noriega, P. Garcia, and C. Sierra. Towards a test-bed for trading agents in electronic auction markets. *AI Communications*, 11:5–19, 1998.

[Woo00] M. Wooldridge. *Reasoning about rational agents*. MIT Press, July 2000.