The bit transmission problem revisited

Alessio Lomuscio Marek Sergot
Department of Computing
Imperial College of Science, Technology and Medicine
London SW7 2BZ, United Kingdom
{A.Lomuscio,M.Sergot}@doc.ic.ac.uk

January 17, 2002

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 of this note 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 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.

The specific aims of this paper are to illustrate how deontic interpreted systems may be applied to model and reason about 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. Nevertheless, using a small, well understood problem has several advantages, including:

- It provides a simple yet interesting scenario on which to test formal apparati that deal with violations, self-correcting protocols, and enforcement mechanisms.
- Because the example is small, it is possible to show in some detail how these features can be treated.
- It is possible to reason about it by means of semantics with finer or coarser levels of detail.
- A syntactic analysis of the scenario is possible by means of standard temporal epistemic languages.

We will consider 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 presented 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 $(S, \sim_1, \ldots, \sim_n, h)$, where the equivalence relations $\sim_i$ are defined on equivalence of local states, and then interpret modalities in the standard modal tradition [HC96]. The resulting logic for modalities K_i is $S5_n$; this models agents with complete introspection capabilities and veridical knowledge [Hin62, MH95].

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].

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 $(S, \sim_1, \ldots, \sim_n, R_1, \ldots, R_n, h)$, where the equivalence relations are defined as above and the R_i's are defined by gR_ig' if $l_i(g') \in G_i$, and give a standard modal logic interpretation.

Knowledge can be modelled on deontic interpreted systems 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. 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:

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

Uses of this modal operator will be illustrated in the examples to follow.

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

[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.

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 [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, and ϵ representing a circumstance under which no bit has been received yet. So we have

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

To model the environment we consider four different local states, representing the possible combinations of messages that have been sent in the current round, by S and R, respectively. The four local states are:

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

where '.' represents configurations in which no message has been sent by the corresponding agent.

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, as will be explained below.

This simple formalisation 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.

$$Act_S = \{sendbit, \lambda\}, \quad Act_R = \{sendack, \lambda\},$$
$$Act_E = \{transmit, lose\}.$$

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

Under the assumptions of determinism we can model the evolution of the system by means of a transition function $\pi : G \times Act \to G$, 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. Intuitively, π codes the fact that the state of the environment determines whether the actions performed by the agents (i.e., the messages they send on the channel) are effective or not. For example, the definition of π contains the following:

$$\pi((0, \epsilon, (sendbit, .)), (sendbit, \lambda, transmit)) = (0, 0, (sendbit, .)),$$
$$\pi((0, \epsilon, (sendbit, .)), (sendbit, \lambda, lose)) = (0, \epsilon, (sendbit, .)),$$

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

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

to capture that when the channel works properly the message does get through and gets processed accordingly by R. 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 $P_i : L_i \to 2^{Act_i}$ 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:

$$P_S(0) = P_S(1) = sendbit,\ P_S((0,ack)) = P_S((1,ack)) = \lambda,$$
$$P_R(\epsilon) = \lambda,\ P_R(0) = P_R(1) = sendack.$$

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

For the environment, we use the constant function:

$$P_E(X) = \{transmit, lose\}$$

where X is a variable ranging over L_E. Note that while we have assumed determinism for the agents, we work under the assumption of nondeterminism for the environment (but without making any probabilistic assumptions about its behaviour). If we assume that the system starts from a state $g_0 = (0, \epsilon, (.,.))$, 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. This analysis can formally be made by using the mechanism of *contexts* [FHMV97] — for our purposes it is not necessary to pursue this analysis here.

The set of global states reachable from the initial configurations $\{(0,\epsilon,(.,.)),(1,\epsilon,(.,.))\}$ 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[2].

[2]More precisely, the table represents the quotient set of the set of reachable states with respect to an equivalence relation defined by $(l_s, l_r, l_e) \sim (l'_s, l'_r, l'_e)$ if $l_s = l'_s$ and $l_r = l'_r$.

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 $P = \{\mathbf{bit} = \mathbf{0}, \mathbf{bit} = \mathbf{1}, \mathbf{recbit}, \mathbf{recack}\}$. 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, $\sim_R, \sim_S$ are equivalence relations on global states as defined in [FHMV95], and h is an interpretation for the atoms in P such that the following holds:

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

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

$$IS_{\mathrm{b}} \not\models \mathbf{recbit} \rightarrow \Diamond \mathbf{recack}$$

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 that

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

which confirms our intuition about the model. It is also worth noting that (since $IS_{\mathrm{b}} \models \mathbf{recack} \rightarrow \mathbf{recbit}$):

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

and perhaps most interestingly that

$$\begin{array}{lll}
IS_{\mathrm{b}} & \models & \mathbf{recack} \rightarrow K_S\big(K_R(\mathbf{bit} = \mathbf{0}) \vee K_R(\mathbf{bit} = \mathbf{1})\big) \\
IS_{\mathrm{b}} & \models & \mathbf{recack} \wedge (\mathbf{bit} = \mathbf{0}) \rightarrow K_S K_R(\mathbf{bit} = \mathbf{0})
\end{array}$$

(and similarly for the case $(\mathbf{bit} = \mathbf{1})$). 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 problem.

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 of R being 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 following section 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; $R_E = \emptyset$, $L_E = G_E$, and we have:

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

How shall we model the local states of the, now potentially faulty, receiver R? One possibility is to extend the set of local states $\{0, 1, \epsilon\}$ with an additional element (ϵ, ack) 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:

$$G_R = \{0, 1, \epsilon\}, \; R_R = \{(0, f), (1, f), (\epsilon, f)\}, \; L_R = G_R \cup R_R.$$

Here, f in the 'red' states R_R is intended to indicate that at least one faulty acknowledgement was sent before the value of the bit had been received[3].

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 functions P_S and P_E described in Section 4, i.e., we take $P'_S = P_S$, $P'_E = P_E$, 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'

[3]In employing this device we are effectively coding something of the past 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.

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 how it should be evaluated on 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, as for P_R:

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

How shall we define P'_R for the red local states $R_R = \{(0, f), (1, f), (\epsilon, f)\}$? 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 any remedial or recovery actions to be taken if errors arise. For the sake of concreteness, let us define

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

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.

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 $g_0 = (0, \lambda, .)$. 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 true in the case of no incorrect behaviour will no longer be valid here. This should be reflected in the analysis.

Let us then explore this further 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 $IS'_{\mathrm{b}} = (G', \sim'_R, \sim'_S, R'_R, R'_S, h')$ resulting from defining the relations $\sim'_i, R'_i, i \in \{S, R\}$ 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. R will 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 undetermined state ϵ otherwise. More formally, for the case of the bit being 0 (the other can

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

Figure 2: The state space of the bit-transmission system in case R may send incorrect acknowledgements.

be done similarly) we shall impose:

$$\pi'((0,\epsilon,X),(sendbit,sendack,lose)) = \\ ((0,ack),(\epsilon,f),(sendbit,sendack))$$
$$\pi'((0,\epsilon,X),(sendbit,sendack,transmit)) = \\ ((0,ack),(0,f),(sendbit,sendack))$$

where X is a variable ranging over L_E. Note that in the second case the result state is a faulty state even though communication has taken place. Once R is in a red state we also impose that it will forever be in a red state, although it will correctly store messages, if received.

$$\pi'((0,(\epsilon,f)),X),(sendbit,sendack,transmit)) = \\ ((0,ack),(\epsilon,f),(sendbit,sendack))$$
$$\pi'((0,\epsilon,X),(sendbit,sendack,lose)) = \\ (0,(0,f),(sendbit,sendack))$$

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 Figure 2, again omitting the environment component for clarity.

The interpretation h' for the new system is the extension of h from the previous section in which the only atom for which the interpretation needs updating is **recbit**, giving:

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

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

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

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

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

and as expected:

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

But note 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:

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

What is more interesting though, is that a particular form of knowledge also still holds. Intuitively if S could make the assumption of R's correct functioning behaviour, then, upon receipt of an acknowledgement, it would then make sense for it to know 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 validate axiom T, i.e., knowledge under assumptions of correct functioning behaviour does not imply truth. This operator is of particular interest in this circumstance because it captures precisely our intuition about the example.

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

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. This is an intuitive result that was validated syntactically, and semantically on 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 of the protocol. In other words, it should be possible to show that $\mathbf{recack} \wedge (\mathbf{bit}=\mathbf{0}) \rightarrow K_S\,K_R\,(\mathbf{bit}=\mathbf{0})$ 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 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 insist on 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. Indeed, it is clear that in this simple example there are only two ways in which R can violate the protocol of Section 4 — the one described in the previous section and this one. 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 those of Section 5 except that we do not include the red local state (ϵ, f) — in this new setting, no fault can have occurred if the bit has not been received yet. So we have:

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

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 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:

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

So in this case, it is easy to spell out the conditions for recovery from a red state. To check that this is indeed the case 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.

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

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

Figure 3: The state space of the bit-transmission system in case R may fail to send acknowledgements when supposed to do so.

for agent R (the results for bit=1 are analogous):

$$\begin{aligned}
&\pi''((0,0,X),(sendbit,.,\alpha)) = (0,(0,f),(sendbit,.))\\
&\pi''(((0,ack),0,X),(sendbit,.,\alpha)) =\\
&\qquad\qquad ((0,ack),(0,f),(sendbit,.))\\
&\pi''((0,(0,f),X),(sendbit,.,\alpha)) = (0,(0,f),(sendbit,.))\\
&\pi''((0,(0,f),X),(sendbit,sendack,lose)) =\\
&\qquad\qquad (0,0,(sendbit,sendack))\\
&\pi''((0,(0,f),X),(sendbit,sendack,transmit)) =\\
&\qquad\qquad ((0,ack),0,(sendbit,sendack)),
\end{aligned}$$

where $X \in L_e$, and $\alpha \in Act_e$. 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 needs no adjustment: h'' is h' with the obvious adjustment to restrict its domain. It is easily confirmed that we have all of:

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

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

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

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

This example shows two rather intuitive points about the formalism of deontic interpreted systems. 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 'Regimentation'

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. We will assume that R may develop faults of either of the two kinds analysed in Sections 5 and 6. We formally introduce a third agent C in the system, a controller whose function is to constrain the behaviour of R. As we understand it, the controller agent C is a (simple) example of what is called an 'inter-agent' in [RAMN+98]. Its local states are:

$$G_C = L_C = \{green, red\}, \quad R_R = \emptyset.$$

The controller C must be able to detect violations of the protocol by R: we use *green* to represent the case where faults have not developed in the current round or in the past, and *red* to represent states in which they have.

C's actions are the following:

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

The idea is that 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 protocols for S and E are the same as those we have employed so far. As for R, we want the protocol function 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 $(1, f)$. It thus has the value of P_R'' for those states. The value of P_R''' in the state (ϵ, f) can be chosen arbitrarily since

nothing of interest will depend on this; for concreteness, we will define it to be λ. So we have:

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

Note that while C is expected to block erroneous *ack* messages when R is in the critically faulty state (ϵ, f), it cannot prevent R entering this or any other faulty state.

The protocol of C is simple: it either blocks the transmission of messages from R or lets them through, depending on its local state:

$$P_C'''(green) = allow, \quad P_C'''(red) = block.$$

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. It is an assumption of the formalism of interpreted systems that all local states are private to the agents whereas actions are public. Given this, the controller may block R's messages only by performing a sequence of observations of the joint actions performed. We get the desired effect by stipulating that C remains in state *red* until the successful transmission of a bit from S to R is observed, i.e., since S is stipulated to work correctly, the environment action is *transmit*; when this happens, C switches to state *green*, and it remains in that state for the rest of the run. So:

$$\pi'''((B,X,red,Y),(sendbit,\beta,block,lose)) = (B,X',red,Y')$$
$$\pi'''((B,X,red,Y),(sendbit,\beta,block,transmit)) = (B,X'',green,Y'')$$

where (to save space) B stands for 0 or 1, and $(B,X',Y') = \pi''((B,X,Y),(sendbit,\beta,lose))$, $(B,X'',Y'') = \pi''((B,X,Y),(sendbit,\beta,transmit))$. Further:

$$\pi'''((W,X,green,Y),(\alpha,\beta,allow,transmit)) = (W',X',green,Y')$$
$$\pi'''((W,X,green,Y),(\alpha,\beta,allow,lose)) = (W'',X'',green,Y'')$$

where $(W',X',Y') = \pi''((W,X,Y),(\alpha,\beta,transmit))$ and $(W'',X'',Y'') = \pi''((W,X,Y),(\alpha,\beta,lose))$. In all of the above W,X,Y (possibly with superscripts) are variables ranging over the respective local states, and α,β are variables ranging over actions.

When C is in a red state it blocks all attempts to send messages by R:

$$\pi'''(0,\epsilon,red,X),(sendbit,sendack,block,transmit)) = (0,0,green,(sendbit,sendack))$$
$$\pi'''(0,\epsilon,red,X),(sendbit,sendack,block,lose)) = (0,(\epsilon,f),red,(sendbit,sendack))$$

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

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.

Note that the first of the above is a case where C observes the successful arrival of a bit value message at R and so is also a case where C's local state switches from *red* to *green*. Note also that only R's messages can be blocked by C. Last, given the presence of C, we can allow for R to recover from a critical fault and return to one of its green states:

$$\pi'''(0,(\epsilon,f),red,X),(sendbit,.,block,lose)) = \\ (0,\epsilon,red,(sendbit,.))$$

Once again we have presented only the main cases of the definition here, and leave it to the reader to complete the other (straightforward) cases. We assume that C's state does not affect the transitions of E, i.e., the transition function π''' is equal to π'' when restricted to the states of E.

We may now move to consider the deontic interpreted system $IS''' = (G''', \sim_S, \sim_R, \sim_C, R_S, R_R, R_C, h''')$ generated by the system above. The set of global states $G''' \subset L_S \times L_R \times L_C$, can be computed from the initial states $(0,\epsilon,red,(.))$, $(1,\epsilon,red,(.))$ 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 prime object of interest in this analysis, we leave out C's component from the table as we have done for E earlier. Comparing Figure 4 with Figure 2 one can see that the regimentation mechanism amounts to making unreachable the third and fourth global states in the third row of Figure 2.

Let us now go back to the formulas we analysed before and check whether they hold true on IS'''. We should expect that the introduction of the controller mechanism eliminates the possibility of incorrect acknowledgements reaching S. This is indeed what we find by analysing the formulas.

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

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 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, and encourages us to believe that useful tools and methods can be developed on this basis.

Apart from examining further examples, we have two main lines of development that we are pursuing. First, the investigation of examples has identified a number of technical questions concerning what we called 'monotonic' extensions of protocol functions to include red states. Second, we are investigating an alternative 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].

Acknowledgements

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

References

[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.

[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.

[HC96] G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic.* Routledge, New York, 1996.

[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.

[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.

[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.

[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.