One hundred prisoners and a lightbulb — the logic

Hans van Ditmarsch
Computer Science, University of Otago, New Zealand &
IRIT, University of Toulouse, France, hans@cs.otago.ac.nz

Jan van Eijck
CWI, Amsterdam & OTS, University of Utrecht, Netherlands, jve@cwi.nl

William Wu
Electrical Engineering, Stanford University, USA, willywu@stanford.edu

October 28, 2008

Abstract. We model the '100 prisoners and a lightbulb' puzzle in an epistemic logic incorporating dynamic operators for the effects of information changing events. Such events include both informative actions, where agents become more informed about the non-changing state of the world, and factual changes, wherein the world and the facts describing it change themselves as well. We specify the underlying non-deterministic protocol and verify its postconditions in a recent extension of the model checker DEMO with factual change. We also present a synchronized version of the riddle, for which there are also other protocols, and we report on efforts to minimize the expected termination of such protocols when assuming random scheduling.

Keywords: dynamic epistemic logic, puzzle math, distributed systems, model checking

1. One hundred prisoners and a lightbulb

A group of 100 prisoners, all together in the prison dining area, are told that they will be all put in isolation cells and then will be interrogated one by one in a room containing a light with an on/off switch. The prisoners may communicate with one another by toggling the light-switch (and that the only way in which they can communicate). The light is initially switched off. There is no fixed order of interrogation, or interval between interrogations, and the same prisoner may be interrogated again at any stage. When interrogated, a prisoner can either do nothing, or toggle the light-switch, or announce that all prisoners have been interrogated. If that announcement is true, the prisoners will (all) be set free, but if it is false, they will all be executed. While still in the dining room, and before the prisoners go to their isolation cells (forever), can the prisoners agree on a protocol that will set them free (assuming that at any stage every prisoner will be interrogated again sometime)?

This riddle is known as the 'one hundred prisoners and a lightbulb' problem.[1] Of course, the answer to the riddle is: "Yes, they can." The typical problem solver tries to assign all prisoners the same role. Then it cannot be solved. But because the prisoners are all together prior to the execution of a protocol, they can assign themselves different roles in a protocol—and that insight provides a solution of the riddle. For $n > 2$ prisoners, a protocol to solve the riddle is as follows:

Protocol 1 *The n prisoners appoint one amongst them as the counter. All non-counting prisoners follow the following protocol: the first time they enter the room when the light is off, they turn it on; on all other occasions, they do nothing. The counter follows a different protocol. The first* $n - 2$ *times that the light is on when he enters the interrogation room, he turns it off. The next time he enters the room when the light is on, he (truthfully) announces that everybody has been interrogated.*⊣

The riddle and its solution can be modelled in a dynamic epistemic logic, more precisely: in a logic wherein we can model knowledge but also factual and epistemic change. We need all three. **Knowledge:** The counter will make his announcement when he *knows* that all prisoners have been interrogated. That may be much later than the first time that all prisoners have been interrogated. The former is an *epistemic proposition*. Therefore, we need to model not just facts but also *knowledge*. **Factual change:** Switching the light changes the truth value of the proposition 'the light is on'. This is *factual change*. **Epistemic change:** When the counter enters the interrogation room and sees that the light is on, he makes an informative observation that results in the knowledge that one more prisoner has been interrogated. This is *epistemic change*. And the prisoners combine one with the other in one action, e.g., when they switch the light when they see it off for the first time.

In Section 2 we model the riddle in dynamic epistemic logic. The riddle can also be solved if it is not known whether the light is initially on, by a different protocol. See Section 3.1. If it is known when the

[1] We made some investigation on the puzzle's origin. Hans' source was the ESSLLI 2003 logic summer school in Vienna, where it was said to originate with Moshe Vardi. (Moshe Vardi could not direct us further back in time—but we thank him for his advice.) William heard about the riddle in 2001 and cites (see (Wu, 2002)) an IBM Research site `http://domino.watson.ibm.com/Comm/wwwr_ponder.nsf/challenges/July2002.html` wherein it is mentioned that "this puzzle has been making the rounds of Hungarian mathematicians' parties" in a 23 prisoner version. We did not find informal references before 2001. Recent appearances are in the Mathematical Intelligencer (Dehaye et al., 2003), Peter Winkler's 'Mathematical Puzzles: A Connoisseur's Collection' (Winkler, 2004), and in the (Dutch language) mathematics teachers journal 'Nieuwe Wiskrant' (van Ditmarsch, 2007).

interrogations take place, e.g., that there is one interrogation per day, yet other protocols exist to solve it that are far more efficient than the above protocol. This includes protocols where all prisoners play the same role, but where on different days the state of the light has a different meaning. See Section 4. In Section 5 we model the problem and its solution in the model checker DEMO.

2. Solution in dynamic epistemic logic

2.1. Epistemic logic with epistemic and factual change

Dynamic epistemic logics involving both epistemic and factual change have been proposed in (Baltag et al., 1999; Baltag, 2002; van Eijck, 2004; van Ditmarsch et al., 2005; van Ditmarsch, 2006; van Benthem et al., 2006; Herzig and Lima, 2006; Kooi, 2007; van Ditmarsch and Kooi, 2008). The Appendix (page 24) contains an overview of the language and its semantics, based on the presentation in (van Benthem et al., 2006; van Ditmarsch and Kooi, 2008). Below, we merely highlight the essential constructs.

The logical language contains atomic propositions, all the propositional inductive constructs, and clauses $K_a\varphi$, for 'agent a knows φ' (for example, the counter knows that all non-counters have been interrogated), $C_B\varphi$, for 'the agents in group B commonly know φ' (for example, the prisoners know that all prisoners have been interrogated), and $[\mathsf{U},\mathsf{e}]\varphi$, for 'after every execution of update (U,e), formula φ holds.' The distinct events that the counter and non-counters execute in the protocol will be modelled as such updates, for example, 'if the light is on, counter a turns it off.' We interpret the language on pointed Kripke models where the accessibility relations representing the knowledge of the players are equivalence relations. Updates (U,e) can also be seen as such structures, where (also called) event e is the designated point of update model U. If two events cannot be distinguished by an agent, they are in the same equivalence class in the update model, for example, at the time the interrogation takes place, the counter cannot distinguish any of the non-counters being interrogated. Each event in an update model has a precondition φ for execution and a postcondition consisting of a set of bindings $p := \psi$ to describe factual change (as above, in 'if the light is on, counter a turns it off'). This determines the execution of an update model in an epistemic model, that is a restricted modal product.

2.2. One hundred prisoners in dynamic epistemic logic

To model this problem as a multi-agent system, we need to specify the set of agents, the set of relevant atomic propositions, provide an initial epistemic model, and define the updates (a.k.a. events) that are possible in that model.

Agents As agents we take the n prisoners: $N = \{0, \ldots, n-1\}$. Prisoner 0 is the counter. The other prisoners are called non-counters.

Atoms and relevant epistemic formulas Atomic proposition p stands for 'the light is on'. Atomic propositions q_i, for $1 \leq i \leq n-1$, stand for '(now or at a prior interrogation) non-counter i has turned on the light'. Formula $\bigwedge_{i=1}^{n-1} q_i$—for which we write the shorthand $\bigwedge_i q_i$ from now on—means that all non-counters have been interrogated, and $K_0 \bigwedge_i q_i$ means that the counter knows that all non-counters have been interrogated. To observe the light, the counter must be under interrogation himself, so this implies that all prisoners have been interrogated. Therefore we do not need an atom q_0 expressing that the counter has been interrogated.

Initial epistemic model The initial model $\mathcal{I}_n$ consists of the single state where all atoms $p, q_1, \ldots, q_{n-1}$ are false, and that is accessible by all prisoners. This represents their state of knowledge when they are in the dining area together, prior to the start of the interrogations.

Update model for the interrogation An informal description of all relevant interrogation events is as follows. The lower index refers to the name of the prisoner involved in the event. The variable lower index i runs over all non-counters $1 \leq i \leq n-1$. The 'nothing happens' event is needed to express that the prisoners are uncertain about the interval between interrogations. We explain it below.

- $\mathsf{e}_\emptyset$: nothing happens
- $\mathsf{e}_i^{\neg p}$: if the light is off, then, if you have not turned on the light before, turn it on, or else, do nothing.
- e_i^{p}: if the light is on, do nothing.
- $\mathsf{e}_0^{\neg p}$: if the light is off, do nothing.
- e_0^{p}: if the light is on, turn it off.

event name	*precondition*	*postcondition*
$\mathsf{e}_\emptyset$	if $\top$	then do nothing
$\mathsf{e}_i^{\neg p}$	if $\neg p$	then $p := q_i \to p$ and $q_i := p \to q_i$
e_i^{p}	if p	then do nothing
$\mathsf{e}_0^{\neg p}$	if $\neg p$	then do nothing
e_0^{p}	if p	then $p := \bot$

Figure 1. Preconditions and postconditions of interrogation events

The formal description of these events is in Figure 1. In the figure, 'do nothing' (i.e., 'do not change facts') stands for the empty postcondition ϵ, and 'if $\top$ then do nothing' is the same as 'nothing happens': the trivial precondition is always satisfied. The update model I_n is non-deterministic choice between all these events, with the obvious partitions for the prisoners between the events: a prisoner i (counter or non-counter) can distinguish events involving himself from each other and from any other event: $\mathsf{e}_i^p \not\sim_i \mathsf{e}_i^{\neg p}$, and for $x = p, \neg p$ and $\mathsf{e} \neq \mathsf{e}_i^x$, $\mathsf{e}_i^x \not\sim_i \mathsf{e}$.

The 'nothing happens' event $\mathsf{e}_\emptyset$ is a peculiarity needed in our logic. It ensures that the prisoners remain uncertain of the state of the light, even when they are not interrogated themselves. For example, suppose we are in the initial situation and the counter is not the first to be interrogated. If the 'nothing happens' event were not there, the counter would now know that the light is on, even if he did not know which non-counter turned the light on: if $\mathsf{e}_0^{\neg p}$ did not take place, one of the events $\mathsf{e}_1^{\neg p}, \ldots, \mathsf{e}_{n-1}^{\neg p}$ must have taken place, all of which ensure that p becomes true, and therefore $K_0 p$ is true.

Instead of $\mathsf{e}_i^{\neg p}$ we also could have distinguished two events

$$\begin{array}{ll} \mathsf{e}_i^{\neg p \neg q}: & \text{if } \neg p \wedge \neg q_i \text{ then } p := \top \text{ and } q_i := \top \\ \mathsf{e}_i^{\neg p q}: & \text{if } \neg p \wedge q_i \quad \text{then do nothing} \end{array}$$

They have the same effect as the single event $\mathsf{e}_i^{\neg p}$. Assume that p is false. According to $p := q_i \to p$, if q_i is false, then $q_i \to p$ is true, so p becomes true; whereas if q_i is already true, $q_i \to p$ is false so p remains false. According to $q_i := p \to q_i$, q_i becomes true if it was false, or remains true if it was already true. We prefer the single event $\mathsf{e}_i^{\neg p}$, because the precondition then corresponds to the (fresh) observation of the non-counter.

There is still a discrepancy between our formalization and the formulation of the protocol. When the counter sees the light on for the $n-1$-st time, he announces that everybody has been interrogated *and he does not turn off the light* as before—as there is no reason left to do

event name	*precondition*
e_0^K	$K_0 \bigwedge_i q_i$
$\mathsf{e}_0^{\neg K}$	$\neg K_0 \bigwedge_i q_i$

Figure 2. Preconditions of announcements

so. We can make the match exact by defining

$$\mathsf{e}_0^p : \quad \text{if } p \text{ then } p := p \rightarrow \bigwedge_i q_i$$

thus ensuring that the light remains on if $\bigwedge_i q_i$ is true.

Update model for the announcement The counter's observation that the light is on is a *distinct* event from his announcement that all have been interrogated. The consequence of the observation that the light is on, is a model restriction. In the restricted model formula $K_0 \bigwedge_i q_i$ is true, and only then can the counter announce that. The result of that announcement is common knowledge that all have been interrogated: $C_N \bigwedge_i q_i$ is now true.

We model this announcement as a non-deterministic update I'_n consisting of two events that can be distinguished by all agents. The event $\mathsf{e}_0^{\neg K}$ of *not* announcing that everybody has been interrogated, is a non-event just as in the not-stepping-forward action in the Muddy Children Problem. Announcements do not involve factual change and we write '$K_0 \bigwedge_i q_i$' for 'if $K_0 \bigwedge_i q_i$ then do nothing.' This would otherwise look funny, as making an announcement seems to involve doing something. In dynamic epistemic logic, the announcement is treated as the *observation* by all agents (counter and non-counters) of the truth of the formula $K_0 \bigwedge_i q_i$; whereas 'do nothing' means 'do not change facts'. Indeed, no facts are changed.

Protocol The composition of two updates is also an update (see Appendix). The combined effect of interrogation and announcement is the event $(\mathsf{I}_n \circ \mathsf{I}'_n)$. A direct definition would have been less intuitive, as it would have preconditions involving updates, such as 'the light is on and after processing that information the counter knows that everybody has been interrogated'. Note that the update $(\mathsf{I}_n, \mathsf{e}_\emptyset)$ can only be followed by update $(\mathsf{I}'_n, \mathsf{e}_0^{\neg K})$.

Execution of the protocol consists of iteration of $(\mathsf{I}_n \circ \mathsf{I}'_n)$ until the termination condition $C_N \bigwedge_i q_i$ is satisfied. The termination condition holds after the counter announces $K_0 \bigwedge_i q_i$, i.e. after execution of $(\mathsf{I}'_n, \mathsf{e}_0^K)$.

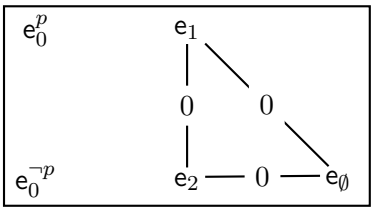

Figure 3. The update model I_3^0 for the interrogation of three prisoners.

2.3. The case of three prisoners

Protocol 1 only requires the counter to learn that all prisoners have been interrogated. Therefore, we are not interested in the knowledge of the non-counters. A solution in a single-agent logic is sufficient. Because we need not process the consequences of the observations of the non-counters for their own knowledge, we can merge the two events for non-counters into a single event without precondition and with the postcondition of $\mathsf{e}_i^{\neg p}$. This is, because the 'do nothing' postcondition in e_i^p has the same effect as '$p := q_i \to p$ and $q_i := p \to q_i$': if p is true, then according to $p := q_i \to p$ the new value of p remains true, and according to $q_i := p \to q_i$ the new value of q_i remains the old value of q_i. Finally, in the single-agent case the final announcement is meaningless. The resulting update model I_3^0 (an upper index 0 distinguishes the single-agent case event models and epistemic models from their multi-agent counterparts) consists of events:

$\mathsf{e}_\emptyset$	if $\top$	then do nothing	
e_1	if $\top$	then $p := q_1 \to p$ and	$q_1 := p \to q_1$
e_2	if $\top$	then $p := q_2 \to p$ and	$q_2 := p \to q_2$
e_0^p	if p	then $p := \bot$	
$\mathsf{e}_0^{\neg p}$	if $\neg p$	then do nothing	

The update model I_3^0 for the case of three prisoners is depicted is Figure 3, and the execution in initial epistemic model $\mathcal{I}_3^0$ of Protocol 1, consisting of repeated execution of update I_3^0 until termination, is depicted in Figure 4. We gave it in a concise graph representation corresponding to the tree-model generated by the initial state and the update model, and identification of bisimilar states.

The top state in Figure 4 represents that all prisoners are in the dining area, so the counter knows that nobody has been interrogated and that the light is off. We can think of the prisoners leaving the dining area as an execution of the update I_3: because nothing is known about the interval between interrogations, immediately after their departure

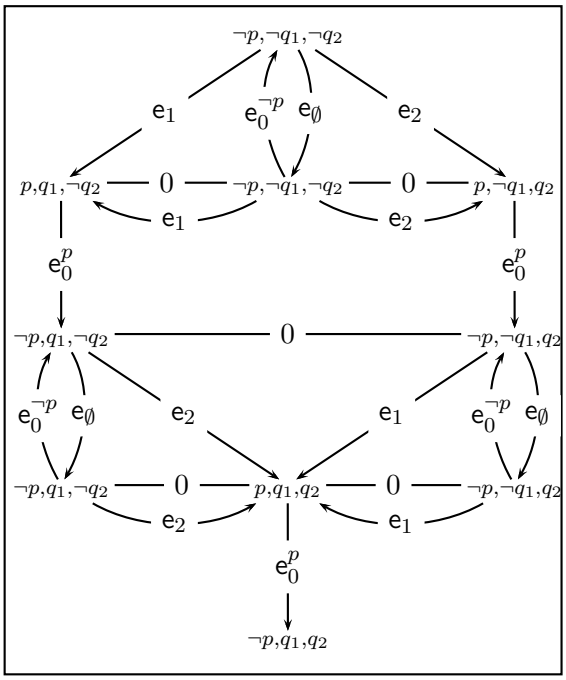

Figure 4. The initial epistemic state (top) and the results of executing the update model for all possible events are pictured for the case of three prisoners trying to escape. The states are indicated by an atomic description. We assume reflexivity and transitivity of access for agent 0. Reflexive arrows for (therefore uninformative) events have not been drawn. For example, in the top-left state events e_1, e_2, and $\mathsf{e}_\emptyset$ can also be executed but have no effect. In the bottom state counter 0 *knows* that non-counters 1 and 2 turned on the light at least once.

and when they have lost sight of each other, it becomes possible that one of them has been selected for interrogation. Even if this only takes place weeks later! The crucial 'nothing happens' even $\mathsf{e}_\emptyset$ ensures that this uncertainty arises, by the transition from the top of the figure to the node below it. It has the same valuation as the top but different epistemic properties: the counter now does not know whether the light is on, because he is uncertain if nothing happened, or 1 has been interrogated, or 2 has been interrogated. Imagine that the counter is *in fact* the first to be interrogated. He then finds the light still off. This is an execution of $\mathsf{e}_0^{\neg p}$, the transition back to the top state of the model. In Section 5 we verify these results in the model checker DEMO.

2.4. Expressive restrictions in the logic

For our modelling purposes, our logic has several restrictions: we cannot refer to past events in preconditions, we cannot express asynchronous behaviour, and we cannot express arbitrary finite execution ('Kleene-star') of events. It is worthwhile to live with such restrictions, as the underlying logic is axiomatizable, as their are model checking tools for verification, etc. Let us explore what the restrictions are.

Referring to past actions The protocol prescribes that a non-counter i turns the light on, *except when he has done so before*. We have introduced atomic propositions q_i in the language that are initially false, become true when non-counter i turns on the light, and then remain true forever. The protocol then prescribes that a non-counter turns the light on, except when q_i is true. An intuitively more appealing proposal with explicit reference to past actions and the past state of the light is tentatively formulated in Section 6.

Asynchronicity The problem states that the interval between interrogations is not known. In other words, the system is asynchronous. But in our logic all agents have some common awareness of the time: the execution of an update corresponds to a clock tick. Any real synchronization that we impose on our modelling must be a refinement of the event sequences: there can be more time points, but not less. Common awareness of the time is undesirable in our setting, as it may allow the prisoners to determine the state of the light without being interrogated. The ad-hoc solution in dynamic epistemic logic is to have a 'nothing happens' event $\mathsf{e}_\emptyset$, that is indistinguishable for an agent from events happening to other agents. Solutions assuming synchronicity are given in Section 4. Section 6 discusses the modelling of 'prisoners' in temporal epistemic logics, such as CTL, wherein these temporal notions are more naturally modelled.

Protocol execution Execution of Protocol 1 means finitely iterating execution of the update model representing non-deterministic choice between the possible events for counters and non-counters, until the termination condition holds. To express this in the logical language, we require an operation U^* on update models. See Section 6.

3. Variations of the asynchronous version

3.1. When the initial state of the light is not known

The riddle can also be solved when it is not known if the light is initially on or off. Why is this not trivial? Consider again the $n-1$ non-counting prisoners. Assume that the light was initially on, and execute Protocol 1. One of the counter's $n-1$ observations that the light is on is then due to the initial state of the light. Therefore, one of the non-counters may *never* have been interrogated. If that is indeed the case, the counter will then falsely announce that every prisoner has been interrogated, and all will be executed. So, that's a no-go. Suppose we increase the count by 1 in Protocol 1, and count to n. In that case, assume that the light was initially off. Now the protocol will not terminate, because the counter will observe only $n-1$ times that the light is on. The prisoners will remain in prison forever.

The information that it is not known what the state of the light is, creates an uncertainty in the count. We can overcome the uncertainty by having each non-counters count more than that disturbance. A solution is therefore to have the non-counting prisoners turn the light on *twice* only, if it is off, and have the counter announce that everybody has been interrogated after he has turned off the light $2n-2$ times.

Protocol 2 *The n prisoners appoint one amongst them as the counter. All non-counting prisoners follow the following protocol:* **the first two times** *they enter the room when the light is off, they turn it on; on all other occasions, they do nothing. The counter follows a different protocol.* **The first 2n − 3 times** *that the light is on when he enters the interrogation room, he turns it off. Then the next time he enters the room when the light is on, he (truthfully) announces that everybody has been interrogated.* ⊣

For example, in the situation where there are 3 prisoners, the counter has to count until $2{\cdot}3-2=4$. This comprises three cases: light originally off, and both non-counter 1 and non-counter 2 turn it on twice; light originally on, non-counter 1 turns it on twice, non-counter 2 turns it on once; light originally on, non-counter 2 turns it on once, non-counter 2 turns it on twice.

We can adjust the epistemic model and the update model by having additional atoms r_i, for 'non-counter i has turned on the light twice'. The events for the non-counter now become:

$\mathsf{e}_i^{\neg p}$	if $\neg p$ then $p := (q_i \wedge r_i) \to p$ and $q_i := p \to q_i$ and $r_i := (p \vee \neg q_i) \to r_i$
e_i^{p}	if p then do nothing

Similar to before, the event $\mathsf{e}_i^{\neg p}$ comprises more intuitive descriptions by reshuffling preconditions to the postcondition:

$$\begin{array}{lll} \mathsf{e}_i^{\neg p \neg q}: & \text{if } \neg p \wedge \neg q_i & \text{then } p := \top \text{ and } q_i := \top \\ \mathsf{e}_i^{\neg pq \neg r}: & \text{if } \neg p \wedge q_i \wedge \neg r_i & \text{then } p := \top \text{ and } r_i := \top \\ \mathsf{e}_i^{\neg pqr}: & \text{if } \neg p \wedge q_i \wedge r_i & \text{then do nothing} \end{array}$$

A further simplification for the single-agent epistemic case also proceeds as before.

3.2. Non-counters can count too

If we model the knowledge of the counter and the knowledge of the non-counters, there are scenarios where a non-counter may learn that all have been interrogated before the counter. For a simple case, consider, for three prisoners, the event sequence; ignore the non-announcement events after every interrogation event:

$$\mathsf{e}_1^{\neg p}, \mathsf{e}_0^{p}, \mathsf{e}_1^{\neg p}, \mathsf{e}_2^{\neg p}, \underline{\mathsf{e}_1^{p}}, \mathsf{e}_0^{p}, \mathsf{e}_0^{K}$$

Non-counter 1 turns on the light, then is interrogated again and sees the light off: he can conclude that the counter must have been interrogated. Then he is interrogated again and see the light on (underlined in the sequence above): this can only be because prisoner 2 has now been interrogated for the first time. He then knows that all have been interrogated, and could announce so. And this is *before* the counter is able to make that announcement.

Protocol 3 As protocol 1, plus: After they have initially turned off the light, all non-counters count the number of times he sees they see the sequence 'light off – light on'. A non-counter announces that all have been interrogated after he observed this sequence $n-2$ times. ⊣

We can adjust the event models by changing the announcement event I'_n into one for all prisoners. No other adjustment is required.

e_i^{K}	$\bigvee_{i=0}^{n-1} K_i \bigwedge_i q_i$
$\mathsf{e}_i^{\neg K}$	$\neg \bigvee_{i=0}^{n-1} K_i \bigwedge_i q_i$

It is important to observe that such knowledge *emerges* from the iterated execution of the updates. We do not have to keep count of the sequences 'light off – light on', whether this is all in proper order, and so on. This is of course just as before, for the knowledge of the counter to emerge: also there, we did not need to keep count.

This adjustment is in the style of the original riddle: the requirement is that *a* prisoner can announce that all prisoners have been interrogated, not that it should be a designated prisoner, the counter. If

the prisoners are randomly scheduled for interrogation, the probability that a non-counter can make this announcement before the counter is very low. This can in principle be computed using the techniques for expected duration of termination in the next section.

4. Synchronization

Assume a single interrogation per day takes place. Now we have, for example, that if the counter is not interrogated on the first day, he still learns that the light is on, as another prisoner *must* have been interrogated and turned on the light. In other words, we have *synchronization*. The logical modelling of the problem becomes simpler, because dynamic epistemic logic assumes synchronization. In the update model I_n of the previous section, simply remove the 'nothing happens' event $\mathsf{e}_\emptyset$. This formalizes the interrogation for the synchronized version of the riddle. This non-deterministic update model can be said to represent random scheduling, namely between n executable events $\mathsf{e}_0^p, \mathsf{e}_1^p, ..., \mathsf{e}_{n-1}^p$ if the light is on, or between n executable events $\mathsf{e}_0^{\neg p}, \mathsf{e}_1^{\neg p}, ..., \mathsf{e}_{n-1}^{\neg p}$ if the light is off, respectively. There are no other logical issues here.

When can the prisoners expect to be set free from prison? This question was asked and answered in (unpublished) (Wu, 2002), see also Wu's mathematical puzzle site `http://www.ocf.berkeley.edu/~wwu/riddles/hard.shtml`, that includes contributions of many other individuals. Before we jump in, note that for 100 prisoners:

- The minimum number of days for all prisoners to be interrogated is 100.
- The minimum duration of Protocol 1 is 200 days (e.g., with interrogation sequence $1, 0, 2, 0, 3, 0, ...99, 0$).
- The expected duration for all prisoners to be interrogated once given random scheduling is roughly $100 \ln 100 = 460$ days (see (Wu, 2002)).

Expected termination for protocol 1 Consider again Protocol 1. Let $n = 100$. For the light to be turned on, a non-counter has to be interrogated. We assume random scheduling. The probability of a non-counter to be interrogated is $\frac{99}{100}$. Then the counter has to be interrogated. The probability of that is $\frac{1}{100}$. Then another non-counter has to be interrogated. *Any* other counter. The probability of that is $\frac{98}{100}$. The expectations of those events, in number of days, are $\frac{100}{99}$, $\frac{100}{1}$, $\frac{100}{98}$,

$\frac{100}{1}$, etc., until $\frac{100}{2}$ (before last non-counter), $\frac{100}{1}$ (counter), $\frac{100}{1}$ (last non-counter), $\frac{100}{1}$ (counter, and announcement). This sum is easily computed:

$$\sum_{i=1}^{99}(\frac{100}{i}+\frac{100}{1}) = 99 \cdot 100 + 100 \cdot \sum_{i=1}^{99}\frac{1}{i} \approx 9,900 + 518 = 10,418$$

This amounts to approximately 28.5 years.

Dynamic counter assignment There was a bit of grumbling among the prisoners when this became known. Should it not be easier to stage a break-out now, instead of waiting almost 30 years? Wait, said the logicians among them, there is a faster way to solve this! We can shave off a few years by dividing the protocol in two stages. In the first stage, it is determined who the counter will be. In the second stage, we proceed as before—but we use what we learnt from the first stage.

Protocol 4 *The protocol is divided in two stages. Stage i takes n days. During the first $n-1$ days of this stage, the first prisoner to enter the room twice turns on the light. Suppose this is on day m. At day n of stage i: if the light is off, announce that everybody has been interrogated. Otherwise, turn off the light. Stage ii starts on day $n+1$. The designated counter is the prisoner twice interrogated on day m in stage i. In stage ii, execute Protocol 1, except that: the counter turns off the light $n-(m-3)$ times only and announces the $n-(m-2)$nd time he sees the light on that everybody has been interrogated (he knows that during the first m days of stage i already $m-2$ other prisoners have entered the interrogation room); non-counters who only saw the light off in stage i do nothing; the remaining non-counters act according to Protocol 1.* ⊣

For $n = 100$, if the room is entered twice first on day m, in phase *ii* the counter only has to count up to $99-(m-2)$. The expected number of days before a prisoner enters the room twice is 13 (see (Wu, 2002)). This means that this prisoner knows that 11 other prisoners have already been interrogated. In phase 2, instead of counting to 99, it therefore suffices to count to $99-11 = 88$. The expected termination of Protocol 4 is then about 25 years.[2]

Head counter and assistant counters This is a more involved scenario from (Wu, 2002). It employs a head counter and assistant counters.

[2] What counts most is the obligation for the counter to be interrogated again, and again, each time with an expectancy of 100 days. So 11 days less, means 1100 days off the previous estimate, plus a bit extra given the non-counters that have no job to perform. This shaves off nearly four years from prison.

The protocol consists of two stages, both finite. These are repeated until termination. We describe them informally, as there is a formal follow-up in the next paragraph.

Assume there are 100 prisoners. There is one head counter, and there are nine assistants. In each iteration, in stage i both head counter and assistants act as the counter in Protocol 1, but they stop turning off lights after they have reached a maximum count of 9 (together they can therefore count all non-counters). The other prisoners act as non-counters in Protocol 1. In stage ii the non-counters do nothing, the assistants act as non-counters in Protocol 1, where now turning on a light means that they completed their count to 9, and the counter adds 9 to his current count every time he sees a light on, and then turns it off. On the final day of stage ii, unless the announcement is made, turn it off, and repeat stages i and ii, until termination.

We can still adjust the protocol, by doing something special on the last day of every stage ii in case the light is on (and the count not complete): turn it off, and add 9 to your current count, whether you are counter, assistant, or non-counter. Let the result be k. Stage i and ii are now repeated until termination, but all prisoners remember their count and act accordingly. If the last person was the counter, he only has to collect $\frac{90-k}{9}$ additional lights in the next stage ii. If it was an assistant that had already turned in his count to the head counter, he does nothing in stage i and turns his 9 in again in stage ii, and if he had not done so yet, he will do it twice in stage ii. If the last person was a non-counter, he now has to turn on the light the first 9 times in stage i, and even 10, if he hadn't done so yet in the previous stage i. Anyone else who has already turned on the light before, does nothing.

For 100 prisoners, the expected duration of this protocol, give and take various adjustments such as in the previous paragraph, is about 10 years (see (Wu, 2002)). And this is also the case for the next generalization, the binary tokens protocol. It is not known if the expected termination can be much less than 10 years.

Binary tokens protocol The basic idea behind the head counter and assistant counters protocol was that in order to speed things up, we should sometimes count in clumps rather than one-by-one. In the first stage, assistant counters were assigned to count one-by-one, and the second stage, the master counter counted the clumps collected by the assistant counters.

The head counter and assistant counters protocol can be thought of in terms of exchanging "tokens" with variable point values. To make the analogy clear, imagine that all prisoners not assigned any counting roles start with a token worth one point. During Stage 1, these prisoners

deposit their one-point tokens into the central room by turning on the bulb when they can, and assistant counters collect them. Suppose assistant counters are ordered to count up to 10. Then, in Stage 2, assistant counters can deposit their 10-point tokens into the room by turning on the bulb, and the master counter collects these bigger tokens. Thus, a lighted bulb will represent a different number of points depending on the stage we are in, and we may reach 100 more quickly by counting in terms of higher denomination tokens.

The binary tokens scheme generalizes these ideas. It is presented in (Dehaye et al., 2003). We corrected some minor errors found there. It is defined for n a power of 2, but it can easily be adjusted to any number of prisoners.

Protocol 5 (Binary tokens scheme) *Let n be the total number of prisoners, and suppose n is a power of 2. Define a sequence (P_k) that dictates the number of points a lighted bulb is worth on day k. Furthermore, every P_k will be some nonnegative power of 2. Then, rather than assigning different roles, all prisoners follow the same instructions:*

- *Keep an integer in your head; call it T. Initialize it to $T = 1$.*
- *Let T_m denote the m^{th} bit of T expressed in binary (where the first bit is called the 0th bit).*
- *Upon entering the room on day k, where $P_k = 2^m$, go through four steps:*
  1. *If the light is on, set $T := T + P_{k-1}$, and turn it off.*
  2. *If $T \geq n$, make the announcement.*
  3. *If $T_m = 1$, turn the light on, and set $T := T - P_k$.*
  4. *Else, if $T_m = 0$, leave the light off (i.e., do nothing).* ⊣

Notice that Step (1) amounts to taking a token worth P_{k-1} points left over from the previous day, and Step (3) amounts to depositing a token worth P_k points. In short, all prisoners will collect and deposit tokens whenever they may legally do so, where the value of tokens are universally dictated by a prespecified sequence P_k that is *only* a function of what day it is. Whenever someone accumulates n points worth of tokens, the announcement is made.

It remains to specify what the point sequence (P_k) should be. The sequence should start with a block of consecutive ones, since everyone starts with only one point. If this block is long enough, there will be many prisoners who have collect more than one point, and perhaps a subsequent block of twos would be effective. It can be proved (see (Wu,

2002), using a coupon collection analysis) that the following sequence has an average runtime of $O(n(\ln n)^2)$.

$$(P_k) = (\ \underbrace{1,1,\ldots,1}_{n\ln n+n\ln\ln n}\quad,\quad \underbrace{2,2,\ldots,2}_{n\ln n+n\ln\ln n}\quad,\quad \underbrace{4,4,\ldots,4}_{n\ln n+n\ln\ln n}\quad,\ldots,\quad \underbrace{\frac{n}{2},\frac{n}{2},\ldots,\frac{n}{2}}_{n\ln n+n\ln\ln n}).$$

Notice that (P_k) consists of $\log_2 n$ blocks each of size $n\ln n + n\ln\ln n$, where the terms in the k^{th} block are set to 2^k, where k indexes from 0 to $(\log_2 n) - 1$.

Lastly, if we do not succeed by the time this sequence of length $(\log_2 n)(n\ln n + n\ln\ln n)$ expires, the prisoners still maintain the integers in their heads, and the $V(n)$ sequence restarts on itself. That is, the sequence $V(n)$ is periodic. So we can think of the protocol as going through cycles, where each cycle has $\log_2 n$ stages.

5. DEMO implementation

The epistemic model checker DEMO, based on Haskell, has been developed by Jan van Eijck (van Eijck, 2007), and continues to remain under development. A minor addition in functionality allows the specification of events also involving factual change. With that, we can model 'prisoners' completely in DEMO. We are using the DEMO version that was developed as part of an ESSLLI'08 course in Hamburg. See `http://www.cwi.nl/~jve/courses/esslli08/`.

The following is a literate version of the DEMO program `LB.hs`, that specifies the epistemic model $\mathcal{I}_3^0$ and the update model I_3^0 for three prisoners. First we declare the lightbulb module and import some relevant DEMO files (used in the ESSLLI'08 course):

```
module LB
where

import List
import HFKR
import RPAU
import RAMU
import IEMC
```

Define the relevant formulas p, q_1, q_2, and the crucial formula $K_a(q_1 \wedge q_2)$. Counter 0 is called agent `a` in the program. In DEMO, agents cannot be given numbers as names.

```
p, q1, q2, form :: Form
p  = Prop (P 0)
q1 = Prop (Q 1)
q2 = Prop (Q 2)
form = K a (Conj [q1,q2])
```

Declare the initial model $\mathcal{I}_3^0$ for agent `a`.

```
type EM = EpistM State

initm :: EM
initm = Mo [0] [a] val acc [0]
  where
  val = [(0,[])]
  acc = [(a,0,0)]
```

In `initm :: EM` we specify that `EM` is the type of the epistemic model `initm`. It is then created in `initm = Mo [0] [a] val acc [0]` stating that its domain consists of 0 only, that `a` is the agent, with valuation `val` and access `acc` given on the line below it, and with points (set of designated states) `[0]`; then `val = [(0,[])]` specifies that no facts are true in state 0, and `acc = [(a,0,0)]` specifies that state 0 is accessible by `a` to itself (i.e., universal access).

Instead of a single update model I_3^0 we define, only for convenience, three update models, namely for the interrogation of prisoners $0, 1, 2$, where 0 now indeed is the counter.

```
type UM = FACM State

interrog :: Integer -> UM
interrog 0 = \_ -> Acm
  [0,1]
  [a]
  [(0,(p,[(P 0,Neg Top)])),
   (1,(Neg p,[]))]
  [(a,0,0),(a,1,1)]
  [0,1]
interrog 1 = \_ -> Acm
  [0,1,2]
  [a]
  [(0,(Top, [(P 0,q1 `impl` p),(Q 1,p `impl` q1)])),
   (1,(Top, [(P 0,q2 `impl` p),(Q 2,p `impl` q2)])),
   (2,(Top, []))]
  [(a,x,y) | x <- [0..2], y <- [0..2] ]
```

```
  [0]
interrog 2 = \_ -> Acm
  [0,1,2]
  [a]
  [(0,(Top, [(P 0,q1 `impl` p),(Q 1,p `impl` q1)])),
   (1,(Top, [(P 0,q2 `impl` p),(Q 2,p `impl` q2)])),
   (2,(Top, []))]
  [(a,x,y) | x <- [0..2], y <- [0..2] ]
  [1]
```

In all DEMO update and epistemic models, domain elements have to be numbered starting from 0. The pair wherein the first argument is the number (name) for an event, has as second argument a two-element list containing the precondition and the postcondition in that order. Events `0`, `1` in `interrog 1` correspond to, respectively, $\mathtt{e}_0^p$ and $\mathtt{e}_0^{\neg p}$. Events `0`, `1`, and `2` in `interrog 2` (and also in `interrog 3`) correspond to $\mathtt{e}_1$, $\mathtt{e}_2$, and $\mathtt{e}_\emptyset$, respectively. For example, `(0,(p,[(P 0,Neg Top)]))` is as 'If p then $p := \bot$' for event $\mathtt{e}_0^p$ (`p` is the placeholder for actual propositional variable `P 0`, a syntax peculiarity we can overlook here), and `(0,(Top,[(P 0,q1 `impl` p),(Q 1,p `impl` q1)]))` states that in $\mathtt{e}_1$, with precondition $\top$, the postcondition is that $p := q_1 \to p$ and $q_1 := p \to q_1$. Note that there is no separate interrogation event for 'nothing happens': this reflects that in actual interrogation sequences we can ignore that event, it is merely there to model ignorance of the counter appropriately.

Here are some example update results. After interrogation of prisoners 1 and 0, in that order, the epistemic situation is like this:

```
LB> displayS5 (upds initm [interrog 1, interrog 0])
[0,1]
[(0,[q1]),(1,[q2])]
(a,[[0,1]])
[0]
```

The light is off (for the counter has switched it off), and the counter knows that either prisoner 1 or prisoner 2 was interrogated before him, for he has found the light on. The actual state of affairs is 0, for *in fact* it was prisoner 1 who has switched on the light. The counter knows that the light is off.

Now assume that after these events prisoner 2 gets interrogated. We get the following epistemic situation:

```
LB> displayS5 (upds initm [interrog 1, interrog 0, interrog 2])
[0,1,2]
```

```
[(0,[q1]),(1,[p,q1,q2]),(2,[q2])]
(a,[[0,1,2]])
[1]
```

The light is on now, for prisoner 2 has switched it on. In fact, prisoners 1 and 2 have now both been interrogated, but the counter does not know this. He cannot distinguish the actual situation 1 from the situation where only prisoner 1 has been interrogated and the situation where only prisoner 2 has been interrogated. This changes at the moment where prisoner 0 gets interrogated again. Now the counter knows that all have been interrogated:

```
LB> displayS5 (upds initm [interrog 1, interrog 0, interrog 2, interrog 0])
[0]
[(0,[q1,q2])]
(a,[[0]])
[0]
```

Finally, let us define the protocol. The protocol specifies for every sequence of interrogations what happens to the knowledge state of the counter. Since DEMO is implemented in a lazy functional language, it can handle infinite sequences as arguments.

```
protocol :: [Integer] -> [EM]
protocol = protocol' initm
  where
  protocol' m (i:is) | isTrue m form = [m]
                     | otherwise     =
                       m : protocol' (upd m (interrog i)) is
```

The part `protocol` given as `protocol' initm` applies to a list `[Integer]` of integers describing an interrogation sequence. At any stage, given intermediate result epistemic model `m` and remaining sequence `(i:is)` starting with the interrogation of prisoner `i`, first check if the termination condition `form` is satisfied, and if so, the protocol terminates and the model `m` is output (`isTrue m form = [m]`), otherwise, apply the result of the update of `m` with the interrogation by prisoner `i` (`(upd m (interrog i))`) to the remaining interrogation sequence `is`.

A run of the protocol displays how the knowledge state evolves as the interrogations proceed:

```
run :: [Integer] -> IO ()
run process = sequence_ (map displayS5 (protocol process))
```

The program can easily be generalized to the multi-agent situation where we consider the knowledge of all three prisoners and to the version of the riddle where it is unknown whether the light is on initially.

6. Further research

In temporal logic fair scheduling of prisoners and correctness of the protocol can be expressed directly, unlike in dynamic epistemic logic. This is outlined in Subsection 6.1. In a tentative dynamic epistemic logic with (arbitrary) past operators, we can express directly that a non-counter will turn on light *unless he has done it before*, such that a single atom suffices to model the entire riddle. This is outlined in Subsection 6.2. Issues concerning the optimality of average runtime of protocols have already been addressed in Section 4.

6.1. Dynamic epistemic logic and temporal logic

The semantics of DEL is such that we reason about what the effect of an update event would be, if it were to take place. For example, consider the three interrogation events $0, 1, 2$ in the DEMO program for three prisoners. If the light is on and if event 0 (interrogation of the counter) takes place, then afterwards the light is off, and the counter knows that it is off: $p \to [0]K_0\neg p$ holds. This is not quite what we need for temporal reasoning, because $\langle 0\rangle\top$ does **not** express that the next interrogation *will be* an interrogation of 0. It merely expresses that an update with this interrogation event would succeed.

We want to be able to state the following crucial property of the protocol: If we have a fair interrogation sequence, i.e., a sequence with the property that at every time in the future, no prisoner is excluded from further interrogation, then the protocol should guarantee that the counter will know that each prisoner has been interrogated. This is possible in a linear time temporal logic (LTL) with the usual F, P, G, H operators, expanded with epistemic operators K_a and C_B, and with two dynamic operators $[e]\varphi$ and $(e)\varphi$, for e one of the interrogation events $0, 1, 2$. The intended semantics of $[e]\varphi$ is the DEL semantics: either update with event e fails, or in the updated model φ holds. The intended semantics of $(e)\varphi$ is that the next event is an e event, and that after that event φ holds.

Interpretation takes place with respect to infinite sequences of events π that look like $\pi_1, \pi_2, \ldots$ (compare the DEMO implementation above). Let $(\mathcal{I}_3)_{\pi,n}$ be the model which results from doing updates $\pi_1, \ldots, \pi_n$ on the initial model where the light is off and 0 knows that. One can

now define $\pi, n \models \varphi$ (shorthand for $(\mathcal{I}_3)_{\pi,n} \models \varphi$) as usual, with crucial clause "$\pi, n \models (e)\varphi$ iff $\pi_n = e$ and $\pi, n+1 \models \varphi$" ($\pi_n = e$ means that the next event in the sequence π equals e), and temporal clauses as usual, e.g. "$\pi, n \models F\varphi$ iff for some $m > n$, $\pi, m \models \varphi$." Fairness of an interrogation sequence can now be expressed as:

$$G(F(0)\top \wedge F(1)\top \wedge F(2)\top).$$

Knowledge of 0 that prisoners 1 and 2 have been interrogated is expressed as:

$$K_0(P(1)\top \wedge P(2)\top).$$

Correctness of the protocol, finally, is expressed as:

$$G(F(0)\top \wedge F(1)\top \wedge F(2)\top) \rightarrow FK_0(P(1)\top \wedge P(2)\top).$$

6.2. Temporal uncertainty in dynamic epistemic logic

Protocol 1 prescribes that a non-counter i turns the light on, *except when he has done so before*, which we have modelled with atomic propositions q_i that are initially false, become true when non-counter i turns on the light, and then remain true forever. The q_i are modelling artifacts. A logical language wherein we can explicitly refer to past events would be more natural. We consider an expansion of the dynamic epistemic language in Section 2, that is motivated by the history operators found in (Yap, 2006; Sack, 2007), by quantifying over events as suggested in (Balbiani et al., 2008), and by iterated relativization (iterated announcements) as in (Miller and Moss, 2005). Given an update π, as additional operations we now also allow arbitrary finite iteration π^* and the converse update π^{-1}.

Now let p be the *single* atom in that language, standing for 'the light is on', and let U be the *unique* update model for interrogation that is constructed as the composition of three update models I_n^-, I_n^+, I'_n, where I_n^-, I_n^+ are defined below and I'_n is as before. We recall that $[\mathsf{U}]\varphi$ is by definition $\bigwedge_{\mathsf{e}\in\mathsf{U}}[\mathsf{U},\mathsf{e}]\varphi$. Define $G\varphi$ by abbreviation as $[\mathsf{U}^*]\varphi$ and $H\varphi$ by abbreviation as $[(\mathsf{U}^*)^{-1}]\varphi$, and their duals $P\varphi$ and $F\varphi$ as usual. We interpret this language on tree models (forests) à la (van Benthem et al., 2007) with roots $(\mathcal{I}_n, 0)$. The tentative part is the semantics for the past operator that quantifies over histories of different lengths. (We conjecture that the 'nothing happens' event $\mathsf{e}_\emptyset$ can be employed in a technically preferable semantics over histories of the same length.)

Before, if $\neg p \wedge q_i$ was true, non-counter i has already observed the light being off in the past. We now express this without an atom q_i. In that case, there was a situation in the past wherein non-counter i knew

that the light was off and that was not the initial situation, wherein this was common knowledge that the light was off. In that state, $K_i\neg p \wedge \neg C_N\neg p$ is true. In our previous modelling the prisoner turns the light on at the same time as he observes that it is off, so that we cannot check whether $K_i\neg p \wedge \neg C_N\neg p$. Therefore, we now distinguish events wherein prisoners enter the room and observe the state of the light, from events wherein they reset the light, subject to the consequences of their observations.

Define update models I_n^+ and I_n^- as follows. Update model I_n^+ consists of observations only. The indistinguishability relations are obvious.

$\mathsf{e}_\emptyset$	$\top$
$\mathsf{e}_i^{\neg p}$	$\neg p$
e_i^{p}	p
$\mathsf{e}_0^{\neg p}$	$\neg p$
e_0^{p}	p

Update model I_n^- is as follows:

$\mathsf{e}_\emptyset$	if $(**)$	then do nothing
$\mathsf{e}_i^{\neg p\neg q}$	if $K_i\neg p \wedge \neg P(K_i\neg p \wedge \neg C_N\neg p)$	then $p := \top$
$\mathsf{e}_i^{\neg pq}$	if $K_i\neg p \wedge P(K_i\neg p \wedge \neg C_N\neg p)$	then do nothing
e_i^{p}	if K_ip	then do nothing
$\mathsf{e}_0^{\neg p}$	if $K_i\neg p$	then do nothing
e_0^{p}	if K_ip	then $p := \bot$

The precondition $(**)$ is 'no prisoner knows whether the light is on', i.e., $\neg(K_0p \vee K_0\neg p) \vee \bigvee_{i=1}^{n-1}(K_ip \vee K_i\neg p))$. This complication is needed to keep the update model always executable, even if nothing happened in I_n^+, and thus the modelling asynchronous.

The composition $\mathsf{I}_n^+ \circ \mathsf{I}_n^-$ has the same update effect as I_n, before. The sequence $\mathsf{I}_n^+ \circ \mathsf{I}_n^- \circ \mathsf{I}'_n$ (where I'_n is as before the update model for the announcement) is the update model U, as above, needed to define the past and future operators used in the preconditions of events $\mathsf{e}_i^{\neg p\neg q}$ and $\mathsf{e}_i^{\neg p\neg q}$ above. So the preconditions for events in U contain a past operator defined in terms of that update U, as $P\varphi$ means $\langle(\mathsf{U}^*)^{-1}\rangle\varphi$. Obviously we cannot yet shout victory for this semantics, but the road ahead is clear.

Acknowledgements

We thank Barteld Kooi for his contributions to and for his comments on this work. We also thank ESSLLI'08 participants Andrew Priddle-Higson and Stefan Minica for their solutions in DEMO of the prisoners riddle.

References

Balbiani, P., A. Baltag, H. van Ditmarsch, A. Herzig, T. Hoshi, and T. D. Lima: 2008, "Knowable' as 'known after an announcement". *Review of Symbolic Logic*. to appear.

Baltag, A.: 2002, 'A Logic for Suspicious Players: Epistemic Actions and Belief Updates in Games'. *Bulletin of Economic Research* **54(1)**, 1–45.

Baltag, A., L. Moss, and S. Solecki: 1999, 'The logic of public announcements, common knowledge, and private suspicions'. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam. CWI Report SEN-R9922.

Dehaye, P., D. Ford, and H. Segerman: 2003, 'One hundred prisoners and a lightbulb'. *Mathematical Intelligencer* **25(4)**, 53–61.

Herzig, A. and T. D. Lima: 2006, 'Epistemic Actions and Ontic Actions: A Unified Logical Framework'. In: J. Sichman et al. (eds.): *IBERAMIA-SBIA 2006, LNAI 4140*. pp. 409–418.

Kooi, B.: 2007, 'Expressivity and completeness for public update logics via reduction axioms'. *Journal of Applied Non-Classical Logics* **17(2)**, 231–254.

Miller, J. and L. Moss: 2005, 'The Undecidability of Iterated Modal Relativization'. *Studia Logica* **79(3)**, 373–407.

Sack, Y.: 2007, 'Adding Temporal Logic to Dynamic Epistemic Logic'. Ph.D. thesis, Indiana University, Bloomington, USA.

van Benthem, J., J. Gerbrandy, and E. Pacuit: 2007, 'Merging frameworks for interaction: DEL and ETL'. In: D. Samet (ed.): *Proceedings of TARK 2007*. pp. 72–81.

van Benthem, J., J. van Eijck, and B. Kooi: 2006, 'Logics of Communication and Change'. *Information and Computation* **204(11)**, 1620–1662.

van Ditmarsch, H.: 2006, 'The logic of Pit'. *Knowledge, Rationality & Action (Synthese)* **149(2)**, 343–375.

van Ditmarsch, H.: 2007, 'Honderd gevangenen en een gloeilamp'. *Nieuwe Wiskrant* **27(1)**, 15–18. (in Dutch).

van Ditmarsch, H. and B. Kooi: 2008, 'Semantic Results for Ontic and Epistemic Change'. In: G. Bonanno, W. van der Hoek, and M. Wooldridge (eds.): *Logic and the Foundations of Game and Decision Theory (LOFT 7)*, Texts in Logic and Games 3. Amsterdam: Amsterdam University Press, pp. 87–117.

van Ditmarsch, H., W. van der Hoek, and B. Kooi: 2005, 'Dynamic Epistemic Logic with Assignment'. In: *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 05)*. New York, pp. 141–148.

van Eijck, J.: 2004, 'Guarded Actions'. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam. CWI Report SEN-E0425.

van Eijck, J.: 2007, 'DEMO — A Demo of Epistemic Modelling'. In: J. van Benthem, D. Gabbay, and B. Löwe (eds.): *Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop*, No. 1 in Texts in Logic and Games. Amsterdam University Press, pp. 305–363.

Winkler, P.: 2004, *Mathematical Puzzles: A Connoisseur's Collection*. AK Peters.

Wu, W.: 2002, '100 prisoners and a lightbulb'. Work in progress available at `http://www.ocf.berkeley.edu/~wwu/papers/100prisonersLightBulb.pdf`.

Yap, A.: 2006, 'Product Update and Looking Backward'. Technical report, University of Amsterdam. ILLC Research Report PP-2006-39.

Appendix: dynamic epistemic logic with factual change

Epistemic model The models which adequately present an information state in a multi-agent environment are the Kripke models from epistemic logic. The set of states together with the accessibility relations represent the information the agents have. If one state s has access to another state t for an agent a, this means that, if the actual situation is s, then according to a's information it is possible that t is the actual situation.

Let a finite set of agents N and a countable set of propositional variables P be given. An *epistemic model* is a triple $M = (S, R, V)$ such that

- S is a non-empty set of possible states,
- $R : N \to \wp(S \times S)$ assigns an accessibility relation to each agent a,
- $V : P \to \wp(S)$ assigns a set of states to each propositional variable.

A pair (M, s), with $s \in S$, is called an *epistemic state*.

Update model An epistemic model represents the information of the agents. *Information change* should therefore be modelled as changes of such a model. One can model an information changing event in the same way as an information state, namely as some kind of Kripke model: there are various possible events, which the agents may not be able to distinguish. This is the domain of the model. Rather than a valuation, a precondition captures the conditions under which such events may occur.

An *update model* (event model) for a finite set of agents N and a language $\mathscr{L}$ is a quadruple $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{post})$ where

- E is a finite non-empty set of events,
- $\mathsf{R} : N \to \wp(\mathsf{E} \times \mathsf{E})$ assigns an accessibility relation to each agent,
- $\mathsf{pre} : \mathsf{E} \to \mathscr{L}$ assigns a precondition to each event,
- $\mathsf{post} : \mathsf{E} \to (P \rightharpoonup \mathscr{L})$ assigns a partial substitution with a finite domain to each event. This is called the *postcondition* of that event.

A pair (U, e) with a distinguished actual event $\mathsf{e} \in \mathsf{E}$ is called an *update*. Instead of

$\mathsf{pre}(\mathsf{e}) = \varphi$ *and* $\mathsf{post}(\mathsf{e})(p_1) = \psi_1$, ... *and* $\mathsf{post}(\mathsf{e})(p_n) = \psi_n$

we also write

for event e*: if* φ*, then* $p_1 := \psi_1$*, ..., and* $p_n := \psi_n$*.*

Execution of update model in epistemic model The effects of these information changing events on an information state are as follows. Given are an epistemic model $M = (S, R, V)$, a state $s \in S$, an update model $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{post})$ for a language $\mathscr{L}$ that can be interpreted in M, and an event $\mathsf{e} \in \mathsf{E}$ with $(M, s) \models \mathsf{pre}(\mathsf{e})$. The result of executing (U, e) in (M, s) is the model $(M \otimes \mathsf{U}, (s, \mathsf{e})) = ((S', R', V'), (s, \mathsf{e}))$ where

- $S' = \{(t, \mathsf{f}) \mid (M, t) \models \mathsf{pre}(\mathsf{f})\}$,
- $R'(a) = \{((t, \mathsf{f}), (u, \mathsf{g})) \mid (t, u) \in R(a) \text{ and } (\mathsf{f}, \mathsf{g}) \in \mathsf{R}(a)\}$,
- $V'(p) = \{(t, \mathsf{f}) \mid (M, t) \models \mathsf{post}(\mathsf{f})(p)\}$.

Dynamic epistemic logic Event models can be used to define a logic for reasoning about information change. An update is associated with a dynamic operator in a modal language, based on epistemic logic. The updates are now part of the language: an update (U, e) is an inductive construct of type α that should be seen as built from simpler constructs of type φ, namely the preconditions and postconditions for the events of which the update consists.

Language Let a finite set of agents N and a countable set of propositional variables P be given. The language $\mathscr{L}$ is given by the following BNF:

$$\varphi \ ::= \ p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_a\varphi \mid C_B\varphi \mid [\mathsf{U}, \mathsf{e}]\varphi$$

where U is a finite update model for N and $\mathscr{L}$. We use the usual abbreviations, in particular for $\top$ (true) and $\bot$ (false), we write $\langle \mathsf{U}, \mathsf{e} \rangle$ for $\neg[\mathsf{U}, \mathsf{e}]\neg\varphi$, and $[\mathsf{U}]\varphi$ stands for $\bigwedge_{\mathsf{f} \in \mathsf{U}} [\mathsf{U}, \mathsf{f}]\varphi$.

Semantics The semantics of this language is standard for epistemic logic and based on the product construction for the execution of update models from the previous section. Below, $R(B)$ is the reflexive transitive closure of the union of all accessibility relations $R(a)$ for agents $a \in B$.

Let a model (M, s) with $M = (S, R, V)$ be given. Let $a \in N$, $B \subseteq N$, and $\varphi, \psi \in \mathscr{L}$.

$(M, s) \models p$ iff $s \in V(p)$
$(M, s) \models \neg\varphi$ iff $(M, s) \not\models \varphi$
$(M, s) \models \varphi \wedge \psi$ iff $(M, s) \models \varphi$ and $(M, s) \models \psi$
$(M, s) \models K_a\varphi$ iff $(M, t) \models \varphi$ for all t such that $(s, t) \in R(a)$
$(M, s) \models C_B\varphi$ iff $(M, t) \models \varphi$ for all t such that $(s, t) \in R(B)$
$(M, s) \models [\mathsf{U}, \mathsf{e}]\varphi$ iff $(M, s) \models \mathsf{pre}(\mathsf{e})$ implies $(M \otimes \mathsf{U}, (s, \mathsf{e})) \models \varphi$

Composition Given two update models, their composition is another update model. Let update models $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{post})$ and $\mathsf{U}' = (\mathsf{E}', \mathsf{R}', \mathsf{pre}', \mathsf{post}')$ and events $\mathsf{e} \in \mathsf{E}$ and $\mathsf{e}' \in \mathsf{E}'$ be given. The *composition* $(\mathsf{U}, \mathsf{e}) \circ (\mathsf{U}', \mathsf{e}')$ of these update models is $(\mathsf{U}'', \mathsf{e}'')$ where $\mathsf{U}'' = (\mathsf{E}'', \mathsf{R}'', \mathsf{pre}'', \mathsf{post}'')$ is defined as follows

- $\mathsf{E}'' = \mathsf{E} \times \mathsf{E}'$,
- $\mathsf{R}''(a) = \{((\mathsf{f}, \mathsf{f}'), (\mathsf{g}, \mathsf{g}')) \mid (\mathsf{f}, \mathsf{g}) \in \mathsf{R}(a) \text{ and } (\mathsf{f}', \mathsf{g}') \in \mathsf{R}'(a)\}$,
- $\mathsf{pre}''(\mathsf{f}, \mathsf{f}') = \mathsf{pre}(\mathsf{f}) \wedge [\mathsf{U}, \mathsf{f}]\mathsf{pre}'(\mathsf{f}')$,
- $\mathrm{dom}(\mathsf{post}''(\mathsf{f}, \mathsf{f}')) = \mathrm{dom}(\mathsf{post}(\mathsf{f})) \cup \mathrm{dom}(\mathsf{post}'(\mathsf{f}'))$ and if $p \in \mathrm{dom}(\mathsf{post}''(\mathsf{f}, \mathsf{f}'))$, then

$$\mathsf{post}''(\mathsf{f}, \mathsf{f}')(p) = \begin{cases} \mathsf{post}(\mathsf{f})(p) & \text{if } p \notin \mathrm{dom}(\mathsf{post}'(\mathsf{f}')), \\ [\mathsf{U}, \mathsf{f}]\mathsf{post}'(\mathsf{f}')(p) & \text{otherwise.} \end{cases}$$

We can either sequentially execute two update models, or compute their composition and execute that: $\models [\mathsf{U}, \mathsf{e}][\mathsf{U}', \mathsf{e}']\varphi \leftrightarrow [(\mathsf{U}, \mathsf{e}) \circ (\mathsf{U}', \mathsf{e}')]\varphi$.

See (van Benthem et al., 2006; van Ditmarsch and Kooi, 2008) for more details.