

One hundred prisoners and a lightbulb — logic and computation

Hans van Ditmarsch*, Jan van Eijck† and William Wu‡

November 26, 2009

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 different protocols to solve the problem, and we give the expected termination of such protocols when assuming random scheduling.

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 is the only way in which they can communicate). The light is initially switched off. There is no fixed order of interrogation, or fixed 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, 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. 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; [20] cites 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

*Philosophy, University of Seville, Spain, hvd@us.es

†CWI, Amsterdam & OTS, University of Utrecht, Netherlands, jve@cwi.nl

‡Electrical Engineering, Stanford University, USA, willywu@stanford.edu

informal references before 2001. Recent appearances are in the Mathematical Intelligencer [4], Peter Winkler’s ‘Mathematical Puzzles: A Connoisseur’s Collection’ [19], and in the (Dutch language) mathematics teachers journal ‘Nieuwe Wiskrant’ [13].

Of course, the answer to the riddle is: “Yes, they can.” The typical problem solver tries to assign all prisoners the same role. Then, as far as we know, 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.* –1

The riddle and its solution can be modelled in a dynamic epistemic logic, more precisely: in a logic wherein we can model knowledge and 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. In Section 3 we solve variations of the riddle. For example, the riddle can also be solved if it is not known whether the light is initially on, but by a protocol different from the above. If there is a fixed and publicly known interval between interrogations, e.g., if there is exactly one interrogation per day, yet other protocols exist to solve it, even protocols wherein all prisoners play the same role but where on different days the state of the light has a different meaning. In Section 4 we present results for the expected termination of such protocols, on the assumption of uniform scheduling.

Knowledge plays a crucial role both in the formulation of the riddle as in its analysis. Many who are confronted with the riddle incorrectly assume that the solution requirement is ‘it is common knowledge that everybody has been interrogated’ (all prisoners know that everybody has been interrogated, and they know that everybody knows, and so on...) or ‘it is general knowledge that everybody has been interrogated’ (all prisoners know that everybody has been interrogated). But it is only required that (at least) ‘someone knows that everybody has been interrogated’. The *first* cannot be achieved – and we will even show (Section 3.4) that common knowledge of whatever kind cannot meaningfully increase in this setting, although for other versions of the riddle it can; and to achieve the *second* can be expected to take much longer than to achieve the *third*. In this sense epistemic logics meaningfully contribute to the analysis of the problem and its ramifications.

We also think that the analysis of the problem contributes to the area of epistemic logics. Firstly, it is an in-depth case study applying the full functionality of a known logic – we do

not propose a new logic in this contribution. We do not know of a case-study in dynamic epistemics with factual and informational change where the crucial events are non-public (local behaviour); e.g., in the analysis of the Pit game [12] and in the modal logical modelling of the frame problem [8, 14] all events are public. Secondly, our solution involves an implicit way to handle asynchronous behaviour in a dynamic epistemic logic. Such logics are typically restricted to synchronous interaction. We address the matter in Section 3.3. This may be considered of independent interest. Thirdly, we think that our detailed analysis of the prisoners riddle reveals a challenge to the research community, namely to provide a modelling of the riddle employing an arbitrary past operator (Section 5) – such an operator is not currently available in dynamic epistemics, one-step history operators notwithstanding [7]. For the research into dynamic epistemics the role of epistemic puzzles has often been to reveal such challenges, and a clearly formulated problem is then a result in itself.

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 [3, 2, 17, 16, 12, 11, 5, 6, 15]. The next subsection contains an overview of the language and its semantics, that is based on the presentation in [11, 15], and that for all aspects but factual change goes back to [3]. Readers familiar with the logic are suggested to skip that technical subsection and only read the following highlight of its essential constructs in relation to the prisoners’ riddle.

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 commonly know that all prisoners have been interrogated), and $[U, e]\varphi$, 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’). The execution of an update model in an epistemic model is the computation of a restricted modal product, and this resulting structure can be seen as the state of information after the event.

2.2 Technical preliminaries

Epistemic model The models to 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 non-empty 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,
- $R : N \rightarrow \wp(S \times S)$ assigns an accessibility relation to each agent a ,
- $V : P \rightarrow \wp(S)$ 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 \mathcal{L} is a quadruple $U = (E, R, \text{pre}, \text{post})$ where

- E is a finite non-empty set of events,
- $R : N \rightarrow \wp(E \times E)$ assigns an accessibility relation to each agent,
- $\text{pre} : E \rightarrow \mathcal{L}$ assigns a *precondition* to each event,
- $\text{post} : E \rightarrow (P \rightarrow \mathcal{L})$ assigns a *postcondition* to each event for each atom.

Each $\text{post}(e)$ is required to be only finitely different from the identity function $\epsilon(p) = p$. The finite difference is called the *domain* $\text{dom}(\text{post}(e))$ of $\text{post}(e)$. Note that the domain of ϵ is empty, which explains its name. A pair (U, e) with a distinguished actual event $e \in E$ is called an *update*. We will denote

$$\text{pre}(e) = \varphi \text{ and } \text{post}(e)(p_1) = \psi_1, \dots \text{ and } \text{post}(e)(p_n) = \psi_n$$

using the expression

$$\text{for event } e: \text{ if } \varphi, \text{ then } p_1 := \psi_1, \dots, \text{ and } p_n := \psi_n.$$

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 $U = (E, R, \text{pre}, \text{post})$ for a language \mathcal{L} that can be interpreted in M , and an event $e \in E$ with $(M, s) \models \text{pre}(e)$. The result of executing (U, e) in (M, s) is the model $(M \otimes U, (s, e)) = ((S', R', V'), (s, e))$ where

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

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 \mathcal{L} is given by the following BNF:

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

where $p \in P$, $a \in N$, $B \subseteq N$, and (U, e) is a finite update (i.e., the domain of U is finite) for N and \mathcal{L} . We use the usual abbreviations, in particular for \top (true) and \perp (false), we write $\langle U, e \rangle$ for $\neg[U, e]\neg\varphi$, and $[U]\varphi$ stands for $\bigwedge_{f \in U} [U, f]\varphi$.

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

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

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

A formula φ is valid, notation $\models \varphi$, iff, given an epistemic model (for agents N and atoms P), it is true in all its states.

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

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

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

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

2.3 One hundred prisoners in dynamic epistemic logic

To model the prisoners riddle 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 that are possible in that model.

Agents As agents we take the n prisoners: $N = \{0, \dots, n - 1\}$ (where $n \leq 1$). 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 $\bigwedge_{i>0} q_i$ from now on—means that all non-counters have been interrogated, and $K_0 \bigwedge_{i>0} q_i$ 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, \dots, 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.

- e_\emptyset : nothing happens
- $e_i^{\neg p}$: if the light is off, then turn it on in case you have not turned it on before, or else do not change the state of the light.
- e_i^p : if the light is on, do not change the state of the light.
- $e_0^{\neg p}$: if the light is off, do not change the state of the light.
- e_0^p : if the light is on, turn it off.

The formal description of these events is in Figure 1. In the figure, the identity or empty postcondition ϵ stands for ‘do not change the state of the light’ (more precisely: do not change the value of any fact) , and ‘if \top then ϵ ’ represents ‘nothing happens’: the trivial precondition is always satisfied.

The update model \mathbb{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: $e_i^p \not\sim_i e_i^{\neg p}$, and for $x = p, \neg p$ and $e \neq e_i^x$: $e_i^x \not\sim_i e$.

The ‘nothing happens’ event 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

<i>event name</i>	<i>precondition</i>	<i>postcondition</i>
e_\emptyset	if \top	then ϵ
e_i^{-p}	if $\neg p$	then $p := q_i \rightarrow p$ and $q_i := p \rightarrow q_i$
e_i^p	if p	then ϵ
e_0^{-p}	if $\neg p$	then ϵ
e_0^p	if p	then $p := \perp$

Figure 1: Preconditions and postconditions of interrogation events

themselves. For example, suppose that we are in the initial situation and that the counter is not the first to be interrogated. If the ‘nothing happens’ event were not there, then the counter would know after the first interrogation that the light is on, even if he did not know which non-counter turned the light on: if e_0^{-p} did not take place, one of the events $e_1^{-p}, \dots, e_{n-1}^{-p}$ must have taken place, all of which ensure that p becomes true, and therefore $K_0 p$ is true.

‘Nothing happens’ is commonly known as a ‘clock tick’: nothing happens except that all agents learn that the time has progressed. But this is a not entirely appropriate description in our setting, because the event that nothing happens is not public. For example, as long as a prisoner has not been interrogated, then even after spending many weeks in his isolation cell, that prisoner is uncertain whether the clock has ticked even once. On the other hand, even when a prisoner is taken out of his isolation cell for interrogation immediately, a few split seconds after having been brought there, he cannot rule out that all prisoners have already been interrogated and that thousands of such clock ticks have taken place. There appears to be a relation to the modelling of asynchronous behaviour, which is discussed in Section 3.3.

Instead of e_i^{-p} we also could have distinguished two events

$$\begin{aligned} e_i^{-p-q} &: \quad \text{if } \neg p \wedge \neg q_i \quad \text{then } p := \top \text{ and } q_i := \top \\ e_i^{-pq} &: \quad \text{if } \neg p \wedge q_i \quad \text{then } \epsilon \end{aligned}$$

They have the same effect as the single event e_i^{-p} . Let us show this now. Assume that p is false. According to $p := q_i \rightarrow p$, if q_i is false, then $q_i \rightarrow p$ is true, so p becomes true; whereas if q_i is already true, $q_i \rightarrow p$ is false so p remains false. According to $q_i := p \rightarrow q_i$, q_i becomes true if it was false, or remains true if it was already true. We prefer the single event e_i^{-p} , because the precondition then corresponds to the (fresh) observation of the non-counter. Either way, this does not matter, as the update effect is the same.

There is an inessential 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 so. We could have made the match exact by defining

$$e_0^p : \quad \text{if } p \quad \text{then } p := p \rightarrow \bigwedge_{i>0} q_i$$

thus ensuring that the light remains on if $\bigwedge_{i>0} q_i$ is true.

Another inessential discrepancy is that we do not model the announcement of the counter. Technically, that would be a so-called *public truthful announcement* of the formula $K_0 \bigwedge_{i>0} q_i$. This is a singleton update model, accessible for all prisoners and the counter, where the precondition and postcondition of the single event are: ‘if $K_0 \bigwedge_{i>0} q_i$ then ϵ .’ The result of that announcement is common knowledge that all have been interrogated: $C_N \bigwedge_{i>0} q_i$ is now true.

Protocol Execution of Protocol 1 consists of iteration of l_n until the termination condition $K_0 \bigwedge_{i>0} q_i$ is satisfied. The correctness of our implementation of this protocol cannot be expressed in the logical language, as the language does not contain an infinitary modal operator expressing arbitrarily finite iteration of single events (and as in the branching temporal structure resulting from iterated execution of l_n we cannot select or indicate the terminating run). But we can formulate this on a metalevel. The initial conditions are that all of p, q_1, \dots, q_n are false, and that this is common knowledge. Given these conditions, we show that after termination of the protocol all prisoners have indeed been interrogated. First note that for each i a pair (p, q_i) can only *once* become true during a run of Protocol 1. The only way for an atom q_i to become true is the execution of event $(l_n, e_i^{\neg p})$, and as the only assignment changing the value of q_i is in that event, namely $q_i := p \rightarrow q_i$, its value remains true once it has become true. Also, because of assignment $p := q_i \rightarrow p$, once q_i is true p remains false. Now consider the statement ψ_k for ‘the counter knows that at least k other prisoners have been interrogated’, for $k < n - 1$. If ψ_k is true, then after execution of event (l_n, e_0^p) statement ψ_{k+1} is true: the observation of p is a model elimination of the states with valuation where exactly k atoms q_i (and thus $\neg p$) are true. For $k = n - 2$, note that ψ_{n-1} equals $\bigwedge_{i>0} q_i$, which as said is the termination condition. When he makes his final announcement the counter indeed knows that all prisoners have been interrogated, as all prisoners but himself have been interrogated and he himself has also been interrogated namely when he observes the light that makes him conclude that. And many times before already...

We can also think of validating the results in a model checker. The epistemic model checker DEMO, based on Haskell, has been developed by Jan van Eijck [18]. A minor addition in functionality allows the specification of events also involving factual change. With that, we can model ‘prisoners’ completely in DEMO. Details are found in Appendix 5 on page 18.

2.4 Example: 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 a non-counter i into a single event e_i without precondition and with the postcondition of $e_i^{\neg p}$. This is, because the trivial postcondition in e_i^p has the same effect as ‘ $p := q_i \rightarrow p$ and $q_i := p \rightarrow q_i$ ’: if p is true, then according to $p := q_i \rightarrow p$ the new value of p remains true, and according to $q_i := p \rightarrow q_i$ 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 l_3^0 (an upper index 0 distinguishes the single-agent case event models and epistemic models from their multi-agent counterparts) consists of events:

e_0	if \top	then ϵ
e_1	if \top	then $p := q_1 \rightarrow p$ and $q_1 := p \rightarrow q_1$
e_2	if \top	then $p := q_2 \rightarrow p$ and $q_2 := p \rightarrow q_2$
e_0^p	if p	then $p := \perp$
$e_0^{\neg p}$	if $\neg p$	then ϵ

The update model l_3^0 for the case of three prisoners is depicted in Figure 2, and the execution in initial epistemic model \mathcal{I}_3^0 of Protocol 1, consisting of repeated execution of update l_3^0 until termination, is depicted in Figure 3. We gave it in a concise graph representation

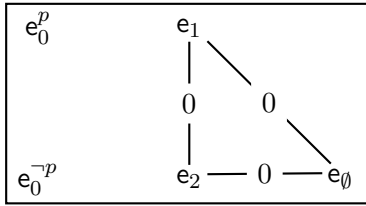


Figure 2: The update model l_3^0 for the interrogation of three prisoners.

corresponding to the tree-model generated by the initial state and the update model, and we identified bisimilar states.

The top state in Figure 3 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 l_3 : because nothing is known about the interval between interrogations, immediately after their departure 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 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 e_0^{-p} , the transition back to the top state of the model.

3 Variations of the riddle

3.1 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 following event sequence (in which we ignore the non-announcement events after every interrogation event):

$$e_1^{-p}, e_0^p, e_1^{-p}, e_2^{-p}, \underline{e_1^p}, e_0^p, e_0^K$$

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 sees 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 2 As protocol 1, plus: After they have initially turned off the light, all non-counters count the number of times 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. \dashv

We can adjust the event models by changing termination condition $K_0 \bigwedge_{i>0} q_i$ into one for all prisoners: $\bigvee_{j=0}^{n-1} K_j \bigwedge_{i>0} q_i$. No other adjustment seems required.¹

¹The correctness requirement is that: given a subsequence π of a run of Protocol 3 (as implemented by event models) that contains event e_i^p before event e_i^{-p} , $\{q_i \wedge \psi_k\} \pi \{\psi_{k+1}\}$; where ψ_k , as before, expresses that

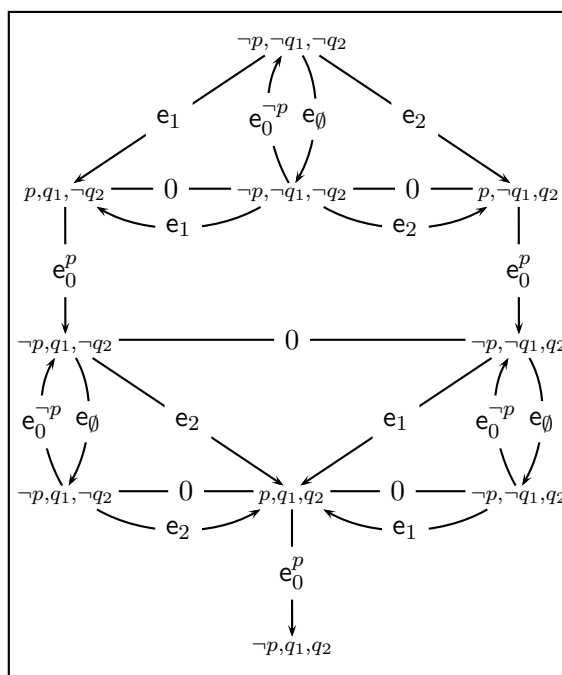


Figure 3: The initial epistemic state (top) and the results of executing the update model for all possible events are pictured for the case of three prisoners trying to escape. The states are indicated by an atomic description. We assume reflexivity and transitivity of access for agent 0. Reflexive arrows for (therefore uninformative) events have not been drawn. For example, in the top-left state events e_1 , e_2 , and 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.

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 seems very low. This can in principle be computed using the techniques for expected duration before termination.

3.2 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.

It creates an uncertainty in the count, that it is not known what the state of the light is. We can overcome the uncertainty of a single count by having each non-counter count more than that. 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 3 *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:

$e_i^{\neg p}$	if $\neg p$	then $p := (q_i \wedge r_i) \rightarrow p, q_i := p \rightarrow q_i$ and $r_i := (p \vee \neg q_i) \rightarrow r_i$
e_i^p	if p	then ϵ

at least k prisoners have been interrogated, for $k < n - 1$. Note that q_i is an extra precondition: prisoner i must already have been interrogated.

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

$$\begin{aligned} e_i^{\neg p \neg q} &: & \text{if } \neg p \wedge \neg q_i & & \text{then } p := \top \text{ and } q_i := \top \\ e_i^{\neg p q \neg r} &: & \text{if } \neg p \wedge q_i \wedge \neg r_i & & \text{then } p := \top \text{ and } r_i := \top \\ e_i^{\neg p q r} &: & \text{if } \neg p \wedge q_i \wedge r_i & & \text{then } \epsilon \end{aligned}$$

A further simplification for the single-agent epistemic case also proceeds as before. A correctness proof for these event model implementations of Protocol 3 requires a correctness statement over subsequences of protocol runs as in the previous subsection.

3.3 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 [9]. In the update model I_n of the previous section, simply remove the ‘nothing happens’ event e_\emptyset . (Note that Protocol 1 is not adjusted, and that correctness follows as before, as we merely changed a detail of the implementation that was not used in the correctness proof.) 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 $e_0^p, e_1^p, \dots, e_{n-1}^p$ if the light is on, or between n executable events $e_0^{\neg p}, e_1^{\neg p}, \dots, e_{n-1}^{\neg p}$ if the light is off, respectively. There are no other logical issues here.

On the assumption of synchronization, various other protocols can also be conceived, and the expected duration before termination then becomes of great interest, however, not of *logical* interest. We will discuss computational issues in detail in Section 4.

If this is the synchronized version, the modelling in Section 2.3 should apparently be seen as an asynchronous version, and the ‘nothing happens’ event e_\emptyset makes the difference. We think that this is of general interest for dynamic epistemics, and that asynchronous behaviour in multi-agent systems can be simulated in dynamic epistemics by adding such a ‘nothing happens’ event to an event model and making it indistinguishable from other events. The reason why e_\emptyset has a temporal effect is that histories of events of different length are indistinguishable if they are the same except for occurrences of e_\emptyset events. For example, sequences $e_1^p, e_\emptyset, e_1^p, e_\emptyset, e_\emptyset$ and e_1^p, e_1^p are indistinguishable for prisoner 1, which makes him uncertain, so to speak, whether the clock has ticked only twice or five times: he does not know what time it is.

3.4 Growth of common knowledge

If it is initially common knowledge that the synchronous version is being played, the prisoners have much more information at their disposition. Under these conditions, common knowledge can strictly increase during the execution of the protocol. For example, as already noticed above, after one day it is common knowledge that the light is on.

To the contrary, for the original setting of the riddle we can prove that *factual common knowledge cannot strictly increase*. Let φ be a boolean proposition of interest that is not initially commonly known. Examples are p – the light is on; but it may be anything, such

as ψ_k – at least k prisoners have been interrogated. The proof is trivial: Assume that φ is not yet commonly known. Execute any of the events $e_i^p, e_i^{\neg p}, e_0^p, e_0^{\neg p}$. If one of the first two is executed, prisoner 0 (and, anyway, all prisoners except i) considers it possible that nothing happened (in model $(1, e_i^p)$ the event e_0 is indistinguishable for 0), and therefore still considers it possible that φ is not commonly known (extend the length of the path to a $\neg\varphi$ state with one step, for 0). If one of the last two happened, all non-counting prisoners consider it possible that nothing happens.

The result can be extended from boolean to modal formulas, but we do not know if it holds for all modal formulas.

4 Computation

If a single interrogation per day takes place, when can the prisoners expect to be set free from prison? This question was asked and answered in the unpublished manuscript [20], see also Wu’s mathematical puzzle site <http://wuriddles.com>, 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 (namely 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.

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:

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

This amounts to approximately 28.5 years.

Dynamic counter assignment There was a bit of grumbling among the prisoners when it became known how long they could expect to have to wait until their release. Should it not be easier to stage a break-out now, instead of waiting almost 30 years? Wait, said some of the smarter 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 $n - m$ times only and announces the $n - (m - 1)$ 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. -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 $100 - (m - 1)$, for example if $m = 2$ we get the original solution where the counter has to count all 99 other prisoners. The expected number of days before a prisoner enters the room twice is 13.² This means that this prisoner knows that 11 other prisoners have already been interrogated. In phase *II*, 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.³

Head counter and assistant counters This is a more involved scenario from [20]. It employs a head counter and assistant counters. 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.

²Let K be a random variable for the number of days before a prisoner enters the room twice, let $\mathbf{E}[\mathbb{I} K]$ be the expectation of its value. Then $P(K = k) = 1 \cdot \frac{99}{100} \cdot \frac{98}{100} \dots \frac{100-k+2}{100} \cdot \frac{k-1}{100} = \frac{100!(k-1)}{100^k \cdot (100-k+1)!}$ such that $\mathbf{E}[\mathbb{I} K] = \sum_{k=1}^{101} k \frac{100!(k-1)}{100^k \cdot (100-k+1)!} = 13.21$ which is about 13 days.

³What 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. This is therefore the result of $\sum_{i=1}^{88} (\frac{100}{i} + \frac{100}{1})$ instead of, as before, $\sum_{i=1}^{99} (\frac{100}{i} + \frac{100}{1})$. Plus the 13 days to be expected for stage *I*...

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 [20]). 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 the informal [20] and in [4] (in our presentation we correct some minor errors in the latter), and emerged gradually in the forum maintained by the author of this contribution. 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) of finite length 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 using a coupon collection analysis⁴ that the following sequence has an average runtime of $O(n(\ln n)^2)$.

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

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 (P_k) sequence restarts on itself. That is, we can see it as an infinite sequence of (P_k) sequences. So we can think of the protocol as going through cycles, where each cycle has $\log_2 n$ stages.

5 Further research

For our modelling purposes, our logic has several restrictions: we cannot refer to past events in preconditions, we can simulate but not really express asynchronous behaviour, we cannot express arbitrary finite execution ('Kleene-star') of events, and we cannot select single runs of a protocol. It is worthwhile to live with such restrictions, as the underlying logic is axiomatizable, as there are model checking tools for verification, etc. Let us explore what is needed to lift these restrictions.

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 would not use such auxiliary variables q_i . If a dynamic epistemic logic with (arbitrary) past operators were to exist...: then we could 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. We hope that the research community will take up this challenge and develop such a dynamic epistemic logic. It would need reference to previous events (as the 'yesterday' operator in Sack's [7], other relevant publications here are [21, 1]) and a Kleene-star operation to refer to the arbitrarily distant past.

In a linear temporal modal logic (LTL) fair scheduling of prisoners and correctness of the protocol can be expressed directly, unlike in dynamic epistemic logic. Temporal logics are also suited to express asynchronous behaviour. An alternative modelling in temporal epistemic logics seems therefore worthwhile to investigate. A relation between dynamic epistemic logics and branching time temporal logics is by way of tree models (forests) à la [10] that are induced by repeated update model execution.

Concerning computational matters, as said it is not known what the minimum expected termination time is (for 100 prisoners) of protocols to solve the riddle. This has already been investigated with extensive simulations and trials without a conclusive answer.

⁴The proof is found in the appendix Coupon Collection Analysis on page 21.

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. Hans van Ditmarsch is also affiliated to Computer Science, University of Otago, New Zealand.

References

- [1] G. Aucher and A. Herzig. From DEL to EDL : Exploring the power of converse events. In K. Mellouli, editor, *ECSQARU*, volume 4724 of *Lecture Notes in Computer Science*, pages 199–209. Springer, 2007.
- [2] A. Baltag. A logic for suspicious players: Epistemic actions and belief updates in games. *Bulletin of Economic Research*, 54(1):1–45, 2002.
- [3] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, 1999. CWI Report SEN-R9922.
- [4] P. Dehaye, D. Ford, and H. Segerman. One hundred prisoners and a lightbulb. *Mathematical Intelligencer*, 25(4):53–61, 2003.
- [5] A. Herzig and T. De Lima. Epistemic actions and ontic actions: A unified logical framework. In J.S. Sichman et al., editors, *IBERAMIA-SBIA 2006, LNAI 4140*, pages 409–418. Springer, 2006.
- [6] B.P. Kooi. Expressivity and completeness for public update logics via reduction axioms. *Journal of Applied Non-Classical Logics*, 17(2):231–254, 2007.
- [7] Y. Sack. *Adding Temporal Logic to Dynamic Epistemic Logic*. PhD thesis, Indiana University, Bloomington, USA, 2007.
- [8] R. Scherl and H. Levesque. Knowledge, action and the frame problem. *Artificial Intelligence*, 144(1–2):1–39, 2003.
- [9] J.F.A.K. van Benthem, J.D. Gerbrandy, T. Hoshi, and E. Pacuit. Merging frameworks for interaction. *Journal of Philosophical Logic*, 38:491–526, 2009.
- [10] J.F.A.K. van Benthem, J.D. Gerbrandy, and E. Pacuit. Merging frameworks for interaction: DEL and ETL. In D. Samet, editor, *Proceedings of TARK 2007*, pages 72–81, 2007.
- [11] J.F.A.K. van Benthem, J. van Eijck, and B.P. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.
- [12] H.P. van Ditmarsch. The logic of Pit. *Knowledge, Rationality & Action (Synthese)*, 149(2):343–375, 2006.
- [13] H.P. van Ditmarsch. Honderd gevangenen en een gloeilamp. *Nieuwe Wiskrant*, 27(1):15–18, 2007. (in Dutch).

- [14] H.P. van Ditmarsch, A. Herzig, and T. De Lima. Optimal regression for reasoning about knowledge and actions. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 1070–1075. AAAI Press, 2007.
- [15] H.P. van Ditmarsch and B.P. Kooi. Semantic results for ontic and epistemic change. In G. Bonanno, W. van der Hoek, and M. Wooldridge, editors, *Logic and the Foundations of Game and Decision Theory (LOFT 7)*, Texts in Logic and Games 3, pages 87–117. Amsterdam University Press, Amsterdam, 2008.
- [16] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. Dynamic epistemic logic with assignment. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 05)*, pages 141–148, New York, 2005. ACM Inc.
- [17] J. van Eijck. Guarded actions. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, 2004. CWI Report SEN-E0425.
- [18] J. van Eijck. DEMO — a demo of epistemic modelling. In J.F.A.K. van Benthem, D. Gabbay, and B. Löwe, editors, *Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop*, number 1 in Texts in Logic and Games, pages 305–363. Amsterdam University Press, 2007.
- [19] P. Winkler. *Mathematical Puzzles: A Connoisseur’s Collection*. AK Peters, 2004.
- [20] W. Wu. 100 prisoners and a lightbulb. Work in progress available at <http://www.ocf.berkeley.edu/~wwu/papers/100prisonersLightBulb.pdf>, 2002.
- [21] A. Yap. Product update and looking backward. Technical report, University of Amsterdam, 2006. ILLC Research Report PP-2006-39.

Appendix: DEMO implementation

The epistemic model checker DEMO, based on Haskell, has been developed by Jan van Eijck [18], 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 `xx**removed for anonymization**xx`.

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 l_3^0 for three prisoners. First we declare the lightbulb module and import some relevant DEMO files :

```

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, e_0^p and $e_0^{\neg p}$. Events 0, 1, and 2 in `interrog 2` (and also in `interrog 3`) correspond to e_1 , e_2 , and e_\emptyset , respectively. For example, $(0, (p, [(P 0, \text{Neg Top}])))$ is as ‘If p then $p := \perp$ ’ for event 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 e_1 , with precondition \top , the postcondition is that $p := q_1 \rightarrow p$ and $q_1 := p \rightarrow q_1$. 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.

Appendix: Coupon Collection Analysis

Let C be a random variable denoting the number of cycles till victory is declared. A cycle of the binary tokens protocol is said to succeed if victory is declared in that cycle, and is deemed a failure otherwise. Each cycle lasts for at most $\tau = (\log_2 n)(n \ln n + n \ln \ln n)$ days, so we can estimate the average runtime of the protocol by computing $\mathbf{E}[C]$. If p_i be the probability that the i th cycle succeeds, then the (p_i) sequence is non-decreasing. This is the case since mental counts are saved from one cycle to the next, which only reduces the competition amongst prisoners to flip the bulb to the ON state. Thus, if C' is the number of cycles when mental counts are erased, then $C \leq C'$, and we have the upper bound $\mathbf{E}[C] \leq \mathbf{E}[C']$. Since C' is a geometric random variable, it only remains to bound the probability that one cycle of the protocol succeeds.

To analyze one cycle, observe that if the m th bit of a prisoner's mental count is 1, then he can only clear this bit by turning on the light bulb in the m th stage, since that is the only stage in which the bulb is worth 2^m points. Thus, the protocol succeeds in one cycle

if and only if at every stage, every prisoner with a nonzero mental count is chosen at least once, where the number of prisoners in Stage k with is $n/2^k$. Hence, each stage reduces to a coupon collection problem. In the k^{th} stage, we collect $n/2^k$ coupons (tokens), and we have $n \ln n + n \ln \ln n$ days to do it. If $\mathbf{P} \left[F_j^{(k)} \right]$ is the probability of failing to collect the j^{th} coupon at the k^{th} stage, where $j \in \{1, \dots, n/2^k\}$, then

$$\mathbf{P} \left[F_j^{(k)} \right] = \left(1 - \frac{1}{n/2^k} \right)^{n \ln n + n \ln \ln n} \leq \left(e^{-2^k} \right)^{\ln n + \ln \ln n} = (n \ln n)^{-2^k} \leq \frac{1}{n \ln n}.$$

Invoking the union bound, $\mathbf{P} \left[F^{(k)} \right]$, the probability of the k^{th} stage failing, is

$$\mathbf{P} \left[F^{(k)} \right] = \mathbf{P} \left[\bigcup_{j=1}^{n/2^k} F_j^{(k)} \right] \leq \sum_{j=1}^{n/2^k} \mathbf{P} \left[F_j^{(k)} \right] \leq \frac{1}{2^k} \frac{1}{\ln n}.$$

Let F denote the event that the first cycle fails. Since failure occurs if and only if at least one of the $\log_2 n$ stages fails, we can again invoke the union bound:

$$\mathbf{P} [F] = \mathbf{P} \left[\bigcup_{k=0}^{(\log_2 n)-1} F^{(k)} \right] \leq \sum_{k=0}^{(\log_2 n)-1} \mathbf{P} \left[F^{(k)} \right] \leq \sum_{k=0}^{(\log_2 n)-1} \frac{1}{2^k} \frac{1}{\ln n} \leq \frac{2}{\ln n}.$$

Let S be the event that the first cycle succeeds. Then $\mathbf{P} [S] = 1 - \mathbf{P} [F] \geq 1 - \frac{2}{\ln n}$. Thus,

$$\mathbf{E} [C] \leq \mathbf{E} [C'] = \frac{1}{\mathbf{P} [S]} \leq \frac{1}{1 - \frac{2}{\ln n}} \longrightarrow 1$$

Conclusively, the expected number of days till the prisoners escape is upper bounded by

$$\mathbf{E} [C'] \cdot \tau = \left(\frac{1}{1 - \frac{2}{\ln n}} \right) \cdot (\log_2 n)(n \ln n + n \ln \ln n) \longrightarrow O(n(\ln n)^2).$$