Verifying One Hundred Prisoners and a Lightbulb

Hans van Ditmarsch, Jan van Eijck and William Wu

[*] *D. Logic*
University of Sevilla, Spain
Email: hvd@us.es

[**] *CWI, Amsterdam & OTS*
University of Utrecht, Netherlands
Email: jve@cwi.nl

[***] *Electrical Engineering*
Stanford University, USA
Email: willywu@stanford.edu

ABSTRACT. *This is a case-study in knowledge representation and dynamic epistemic protocol verification. We analyze the 'one hundred prisoners and a lightbulb' puzzle. In this puzzle it is relevant what the agents (prisoners)* know, *how their knowledge* changes *due to* observations, *and how they affect the state of the world by* changing facts, *i.e., by their actions. These actions depend on the history of previous actions and observations. Part of its interest is that all actions are* local, *i.e. not publicly observable, and part of the problem is therefore how to disseminate local results to other agents, and make them* global. *The various solutions to the puzzle are presented as protocols (iterated functions from agent's local states, and histories of actions, to actions).*
The paper consists of three parts. First, we present different versions of the puzzle, and their solutions. This includes a probabilistic version, and a version assuming synchronicity (the interval between prisoners' interrogations is known). The latter is very informative for the prisoners, and allows different protocols (with faster expected termination). Then, we model the 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. Finally, we verify the basic protocol to solve the problem.
Novel contributions in this paper are: Firstly, Protocol 2 and Protocol 4. Secondly, the modelling in dynamic epistemic logic in its entirety — we do not know of a case study that combines factual and informational dynamics in a setting of non-public events, or of a similar proposal

to handle asynchronous behaviour in a dynamic epistemic logic. Thirdly, our method to verify dynamic epistemic protocols by reasoning over possibly infinite execution sequences of these protocols.
A precursor of the present paper, entitled 'One hundred prisoners and a lightbulb – logic and computation' (van Ditmarsch et al.*, 2010), was presented at KR 2010, Toronto. The differences with the present contribution are as follows: the former contains a section with computational results (expected runtime of different protocols before termination), whereas the focus of the present paper is the verification of one of the presented protocols in the former.*

KEYWORDS: *protocol, verification, dynamic epistemic logic, math puzzle, multi-agent system*

DOI:10.3166/JANCL.v.2–18

1. Protocols

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)?*

1.1. *Origin*

This riddle is known as the 'one hundred prisoners and a lightbulb' problem. We made some investigations on the puzzle's origin, but we did not find references before 2001. On an IBM Research 2002 website (IBM Research, 2002) a 23 prisoner version is given and it is mentioned that "this puzzle has been making the rounds of Hungarian mathematicians' parties". See also (Dehaye *et al.*, 2003; Winkler, 2004; Wu, 2002) and `http://wuriddles.com/`.

1.2. *Knowledge*

Knowledge plays a crucial role in the formulation of the riddle and in its analysis. To solve the riddle it is only required that some prisoner knows that all prisoners have been interrogated, not that all prisoners know that, and certainly not that this is common knowledge. It is impossible to satisfy the latter (and even *any* growth of common knowledge is impossible, see the logical analysis) — unless the interval between interrogations is known in advance.

1.3. *Solution with counter and non-counter*

Of course, the answer to the riddle is: "Yes, they can." The typical problem solver thinks that all prisoners must have the same role. But because the prisoners are all together prior to the execution of a protocol, they can assign themselves different roles. For $n > 2$ prisoners, a protocol to solve the riddle with two different roles for prisoners 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.

1.4. *Non-counters can count too*

A non-counter may learn that all have been interrogated before the counter. Consider the case of three prisoners 0, 1, and 2, where 0 is the counter, and the event sequence (prisoner interrogated, state of light, : $\cdots$)

$$- : 1+ : 0- : 1- : 2+ : 1+ : 0- : \cdots$$

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

PROTOCOL 2. — As protocol 1, plus for the non-counters two cases: (i) if your first interrogation the light is off, then (turn it on according to 1 and) count the number of times you subsequently see the sequence 'light off – light on', and announce that all have been interrogated after observing this sequence $n - 2$ times; (ii) if your first interrogation the light is on, then after being interrogated again when the light is off (turn it on according to 1 and) count the number of times you subsequently see the sequence 'light off – light on', and announce that all have been interrogated after observing this sequence $n - 3$ times.

1.5. *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. This is not trivial. 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. One of the non-counters may *never* have been interrogated. In that case, the counter will falsely announce that every prisoner has been interrogated. But if we were to increase the count by 1 in Protocol 1, and count to n instead of to $n - 1$, 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.

That it is not known what the state of the light is, creates an uncertainty in the count. We can overcome this by having each non-counter count more than the amount of uncertainty.

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 $\mathbf{2n - 3}$ 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.

1.6. *Uniform role protocol*

In the protocols so far, different prisoners perform different roles, and that was the key to solving the puzzle. There is a protocol where all prisoners play the same role, but it is probabilistic. It was suggested by Paul-Olivier Dehaye (in a personal communication). This protocol is easier to present in terms of *tokens*:

Imagine each prisoner to hold a token worth a variable number of points, initially one. Turning the light on if it is off, means dropping one point. Leaving the light on if it is on, means not being able to drop one point. (Before, only a non-counter could drop a point.) Turning the light off if it is on, means collecting one point. Leaving the light off if it is off, means not being able to collect one point. (Before, only the counter collects points.) Protocols terminate once a prisoner has n points.

PROTOCOL 4. — Entering the interrogation room, consider the number of points you carry. If the light is on, add one. Let m be this number. Let a function $Pr : \{0, ..., n\} \rightarrow [0, 1]$ be given, with $Pr(0) = Pr(1) = 1$, $0 < Pr(x) < 1$ for $x \neq 0, 1, n$, and $Pr(n) = 0$. You drop your point with probability $Pr(m)$, otherwise you collect it. The protocol terminates once a prisoner has collected n points.

Dropping a point if you do not carry one, means doing nothing: therefore also $Pr(0) = 1$. Under the above conditions, the protocol terminates. Better odds than any non-zero probability give a Pr that is decreasing in the $1 - \lfloor \frac{n}{2} \rfloor$ range and that is zero on the $\lceil \frac{n}{2} \rceil - n$ range.

Let us explain the example of four prisoners a, b, c, d. Choose $Pr(0) = Pr(1) = 1, Pr(2) = 0.5, Pr(3) = 0, Pr(4) = 0$. Consider the following interrogation sequence, where the lower index stands for the number of points *plus* the state of the light, and where the upper index stands, for the case of $Pr(2)$, for outcome drop (1) or collect (0).

$$- : a_1 + : b_2^1 + : c_2^0 - : d_1 + : b_2^0 - : c_2^0 - : c_2^1 + : b_3 - : c_1 + : b_4$$

Prisoner a gets there first, turns on the light (= drops his point), then b comes in, flips a coin, heads, so does *not* turn off the light (= does *not* collect point), then c comes in, flips a coin, tails, so does turn off the light, then d, light on, then b again, who turns the light off this time and now has 3 points. Crucially, at this point b is designated as the 'counter': as $Pr(3) = Pr(4) = 0$, b will never drop a point but only collect them, until termination. Prisoners a and d already play no role anymore: anyone dropping a single point has count 0, whether the light is on of off does not matter now as $Pr(0) = Pr(1) = 1$ and, as already mentioned, dropping a point if you do not carry one, means doing nothing. Prisoner b now has to wait for c to subsequently drop his token consisting of two points, subject to chance. In the sequence above, the transition "$-c_2^1+$" means that the light is off, c throws heads, so drops one of his two points by turning on the light.

It is important to realize that we cannot define $Pr(2) = 0$, because then a situation can be reached where (as in the above sequence) two players 'stick to their points' so that the protocol will never terminate. But we also cannot have $Pr(2) = 1$, because then no prisoner will ever get more than two points, and the protocol will also not terminate. Probability plays an essential role in this protocol.

1.7. *Synchronization*

Assume the prisoners (commonly) know that a single interrogation per day takes place. This is very informative. 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. On this assumption of *synchronization* other protocols can be conceived, of which we present a few.

1.8. *Dynamic counter assignment*

This protocol consists of two stages.

PROTOCOL 5. — 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 saw the light off in stage I do nothing; the remaining non-counters act according to Protocol 1.

1.9. *Head counter and assistant counters*

A more involved scenario from (Wu, 2002) employs a head counter and assistant counters. Again the protocol consists of two stages, both finite. These are repeated until termination. We describe them informally.

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. (A further refinement is possible, communicating the results of a stage I+II cycle to the next iteration.)

1.10. *Binary tokens protocol*

The binary tokens scheme generalizes the example in the previous paragraph. It was originally presented in (Wu, 2002; Dehaye *et al.*, 2003). We can give different roles to different prisoners, and we can give different meanings to turning on or off the light on different days. We can think of the prisoners exchanging 'tokens' with variable point values, as in Protocol 4. All prisoners start with a token worth one point. In the head/assistant counter scenario, counter and assistants all collect 10 (their own plus 9) in Stage I, and in Stage II the assistants deposit their 10-point tokens into the room by turning on the light and the master counter collects these bigger tokens.

PROTOCOL 6 (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. Every P_k must be a nonnegative power of 2. There is one role for all prisoners:*

– *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:*
  a) *If the light is on, set $T := T + P_{k-1}$, and turn it off.*
  b) *If $T \geq n$, make the announcement.*
  c) *If $T_m = 1$, turn the light on, and set $T := T - P_k$.*
  d) *Else, if $T_m = 0$, leave the light off (i.e., do nothing).*

The protocol is defined for n a power of 2, but it can be adjusted to any number of prisoners. 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, where the values of tokens are dictated by the sequence (P_k). In the computation section we present a suitable sequence (P_k).

2. Logic

The riddle and its solution can be modelled in a dynamic epistemic logic wherein we can model knowledge and also factual and epistemic change. We need all three. The counter will make his announcement when he *knows* that all prisoners have been interrogated. That is only true after it is true that all prisoners have been interrogated. Switching the light changes the truth value of the proposition 'the light is on'. This is *factual 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*.

2.1. *Epistemic logic with epistemic and factual change*

Dynamic epistemic logics involving both epistemic and factual change have been proposed in (Baltag, 2002; van Ditmarsch *et al.*, 2005; van Ditmarsch, 2006; van Benthem *et al.*, 2006; Herzig *et al.*, 2006; Kooi, 2007; van Ditmarsch *et al.*, 2008). We base our summary presentation on (van Ditmarsch *et al.*, 2008).

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 the dynamic modal construct $[\mathsf{U}, \mathsf{e}]\varphi$, 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 off the light.' This allows us to formalize expressions as 'after the event (if the light is on, counter a turns off the light), a knows that at least four prisoners have been interrogated.' 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 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 distinct events 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. For example, above, the precondition is p for 'the light is on' and the postcondition is $p := \bot$ for 'it becomes false that the light is on', i.e., 'counter a turns off the light'. 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. The following subsections contain details of the logic.

2.2. *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,
– $R : N \to \wp(S \times S)$ 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*.

2.3. *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,
- $\mathsf{post} : \mathsf{E} \to (P \to \mathscr{L})$ assigns a *postcondition* to each event for each atom.

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

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

using the expression

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

2.4. *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

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

2.5. *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.

2.6. *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:

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

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 $\mathscr{L}$. 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 $\bigwedge_{\mathsf{f} \in \mathsf{U}}[\mathsf{U}, \mathsf{f}]\varphi$.

2.7. *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 $\varphi, \psi \in \mathscr{L}$.

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

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.

2.8. *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

– $\mathsf{E}'' = \mathsf{E} \times \mathsf{E}'$,
– $\mathsf{R}''(a) = \{((\mathsf{f}, \mathsf{f}'), (\mathsf{g}, \mathsf{g}')) \mid (\mathsf{f}, \mathsf{g}) \in \mathsf{R}(a)$ 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 $\mathsf{post}''(\mathsf{f}, \mathsf{f}')(p) = \mathsf{post}(\mathsf{f})(p)$ if $p \notin \mathrm{dom}(\mathsf{post}'(\mathsf{f}'))$, and $[\mathsf{U}, \mathsf{f}]\mathsf{post}'(\mathsf{f}')(p)$ otherwise.

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

3. One hundred prisoners in dynamic epistemic logic

To model the solution of 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. We focus on the setting of Protocol 1. In that setting, only a single agent needs to be modelled, the counter.

3.1. *Agents, atoms, formulas*

The set of agents consists of prisoner 0, the counter: $N = \{0\}$. The other prisoners are called non-counters. We do not model their knowledge. 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 $\bigwedge_{i>0} q_i$—means that all non-counters have been interrogated, and $K_0 \bigwedge_{i>0} q_i$ therefore means that the counter knows that all non-counters have been interrogated. To observe the light, the counter must be under interrogation, 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.

3.2. *Initial epistemic model*

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

3.3. *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 $1 \leq i \leq n-1$.

event	*precond.*	*postcondition*
e_i	if $\top$	then $p := q_i \rightarrow p$ and $q_i := p \rightarrow q_i$
$\mathsf{e}_0^{\neg p}$	if $\neg p$	then ϵ
e_0^{p}	if p	then $p := \bot$

– e_i: 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; if the light is on, do not change the state of the light.
– $\mathsf{e}_0^{\neg p}$: 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 update model I is non-deterministic choice between all these events, with the obvious partitions for the prisoners between the events: the counter 0 can distinguish events involving himself from each other and from any other event: $\mathsf{e}_0^{p} \not\sim_0 \mathsf{e}_0^{\neg p}$, and $\mathsf{e}_i \not\sim_0 \mathsf{e}_j$ for $i \neq j, i, j > 0$.

It is possible to view $\mathsf{e}_0^{\neg p}$ together with e_0^{p} as a single event e_0, because we can consider it as a multi-pointed update model I with two points $\mathsf{e}_0^{\neg p}$ and e_0^{p}.

3.4. *Protocol*

Execution of Protocol 1 consists of iteration of I 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 I we cannot select or indicate the terminating run). But we can formulate this on a metalevel. Section 4 verifies Protocol 1 in detail.

3.5. *DEMO*

We can think of validating the results in a model checker. The epistemic model checker DEMO, written in Haskell, has been developed by Jan van Eijck (van Eijck, 2007). A minor addition in functionality allows the specification of events also involving factual change. With that, we can model 'prisoners' completely in DEMO. The scripting language of the model checker matches the logic we present closely. It should therefore not be seen as an independent way to determine the correctness of the protocol. DEMO serves our purposes well because it allows us to determine the truth of a given formula after a given event sequence very quickly, prior to thinking systematically about a protocol with that formula as a postcondition.

3.6. *Logic of other protocols*

The logic of other protocols require the modelling or more than one agent's knowledge. For example, in Protocol 2 all the prisoners count. We can adjust the event models by changing termination condition $K_0 \bigwedge_{i>0} q_i$ into one for all prisoners:

$$\bigvee_{j=0}^{n-1} K_j \bigwedge_{i>0} q_i.$$

For details on logical aspects of other protocols see (van Ditmarsch *et al.*, 2010).

4. Verification

To formally prove that Protocol 1 is correct, we need to do several things. In the first place, we have to move from the original informal statement of a solution to a formal version. Above we have used dynamic epistemic logic for this. In this section we will enrich dynamic epistemic logic with a few additional features that effectively turn dynamic epistemic logic into a formalism for describing how knowledge-based programs (in the sense of (Fagin *et al.*, 1997)) get executed.

Next we show formally that the formal version of the protocol (the dynamic epistemic logic program) matches the informal version step-by-step.

Finally we can apply program verification techniques to prove that the dynamic epistemic logic program does what it is supposed to do. It then follows from this that the informal protocol does what it is supposed to do.

Take the case of n prisoners. For $i \in \{0, \dots, n-1\}$, let e_i be the event of the interrogation of prisoner i. Let p express that the light is on. We will slightly streamline the events e_i by extending dynamic epistemic logic with two simple programming constructs: *registers* and *successor*. In particular, we assume a register c for storing natural numbers (the counting register), we allow the factual change operation $c := c+1$, and we assume that it is possible to check statements of the form $c = n$, where n is a natural number. Assuming 0 to be the counter, this allows us to express the event e_0 as the following 'program':

```
case
  light:     light := false, c := c + 1;
  not light: skip
endcase
```

An interrogation sequence for n prisoners numbered $0, \dots, n-1$ is an infinite list of natural numbers, with each number less than n. An example is:

$$\sigma = 0 : 1 : 2 : 3 : 4 : 5 : \sigma$$

Another way to write the same σ is:

$$\sigma = [0{,}1{,}2{,}3{,}4{,}5] +\!\!+ \sigma$$

where $+\!\!+$ is the operation that concats a finite list and a (finite or infinite) list. σ_i is i-th member of σ, counting from 0.

σ is a *fair* interrogation sequence for n prisoners if

– for each i, $0 \leq s_i < n$ (σ is a sequence for n prisoners), and
– for each $i \in n$ and each $j \in \{0, \ldots, n-1\}$ there is a $k \in n$ with $\sigma_{i+k} = j$ (at each point i, each prisoner j will be interrogated at some future point $i + k$).

Input-output can be modelled as follows:

Input: for the case where there are n prisoners: an infinite stream over $\{0, \ldots, n-1\}$, i.e., a member of the set $\{0, \ldots, n-1\}^\infty$.

Output: a natural number (or the protocol runs forever).

The property we have to check is the following *correctness statement*: If σ is a fair interrogation sequence for n prisoners, then protocol PROT$_n$ will output a natural number k with the property that

$$\{0, \ldots, n-1\} \subseteq \{\sigma_i \mid i < k\}.$$

This is a formal version of the informal statement that after the k-th interrogation, all of the n prisoners have been interrogated.

Let $\mathcal{I}$ be the initial epistemic model given before. Let $\mathbf{M}$ be the set of all Kripke models with valuations over the signature. Let E be the set $\{(\mathsf{I}, \mathsf{e}_i) \mid 0 \leq i < n\}$. From now on we will write $(\mathsf{I}, \mathsf{e}_i)$ as e_i. Let U be the function $\mathbf{M} \to E^\infty \to \mathbf{M}^\infty$ given by:

$$U\ M\ (\mathsf{e} : \mathsf{es}) = M \circ \mathsf{e}\ :\ U\ (M \circ \mathsf{e})\ (\mathsf{es}).$$

Then if the sequence of events starts $\mathsf{e}, \mathsf{e}', \mathsf{e}'', \ldots$, the image of $U_{\mathcal{I}}$ starts

$$\mathcal{I} \circ \mathsf{e}_0, \mathcal{I} \circ \mathsf{e}_0 \circ \mathsf{e}_1, \mathcal{I} \circ \mathsf{e}_0 \circ \mathsf{e}_1 \circ \mathsf{e}_2, \ldots$$

Let T be the function $\mathbf{M}^\infty \to n \cup \{\infty\}$ given by

$$\begin{aligned} T(M : \mathrm{ms}) &= T_0(M : \mathrm{ms}) \\ T_i(M : \mathrm{ms}) &= \begin{cases} i & \text{if } M \models K_0(p \wedge c = n-2), \\ T_{i+1}(\mathrm{ms}) & \text{otherwise.} \end{cases} \end{aligned}$$

Now consider the following diagram:

$$\begin{tikzcd}
\{0,\ldots,n-1\}^{\infty} \arrow[r, "\mathrm{PROT}_n"] \arrow[d, "\mathrm{map}\ (\lambda i \mapsto \mathsf{e}_i)"'] & n \cup \{\infty\} \\
\{\mathsf{e}_0,\ldots,\mathsf{e}_{n-1}\}^{\infty} \arrow[r, "U_{\mathcal{I}}"'] & \mathbf{M}^{\infty} \arrow[u, "T"']
\end{tikzcd}$$

THEOREM 7 (CORRECTNESS). — *For all fair streams σ the diagram commutes on a natural number.*

Proof: Induction on the number of prisoners n.

Case $n = 2$: PROT_2 ends after the first occurrence of 10 in the input stream. By fairness, 10 must occur in the stream. After $\mathsf{e}_1\mathsf{e}_0$ occurs in the event stream, $K_0(p \wedge c = 0)$ is true in the resulting epistemic model, and T halts at the position of that model.

Induction step: assume the diagram commutes for all fair streams σ for PROT_n. We have to show that it also commutes for all fair streams for PROT_{n+1}. Let n be the last prisoner that has not been counted (rename prisoners if necessary). From the induction hypothesis we get that there is some k with

$$M_k \models K_0(p \wedge c = n - 2).$$

Now recall that we are considering PROT_{n+1}, so instead of quitting, the counter switches off the light and increments c. Since σ is fair, the pattern $n \cdots 0$ has to occur after position k. Execution of e_n followed by $\cdots$ followed by execution of e_0 will create a model M with $M \models K_0(p \wedge c = n - 1)$. $\square$

5. Further research

5.1. *Puzzle*

We keep discovering and designing more versions of the riddle. Protocol 4 was a recent addition. The authors of (Dehaye *et al.*, 2003) mention generalizations to the computation of any Turing-computable function with n arguments by n prisoners communicating this way—termination is when a prisoner declares the output of the computation.

5.2. *Logic*

For our modelling purposes, the logic we presented has restrictions: we cannot refer to past events in preconditions, we cannot really express asynchronous behaviour, we cannot express arbitrary finite execution ('Kleene-star') of events, and we cannot select single runs of a protocol. As a consequence, we cannot formulate the correctness statement for the protocols we have considered inside our logic. On the positive side, the logic we have presented 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 the light *unless he has done it before*, such that a single atom suffices to model the entire riddle. Works reporting progress in this area are (Sack, 2007; Aucher *et al.*, 2007; Renne *et al.*, 2009).

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. We are investigating alternative modellings in temporal epistemic logic. A relation between dynamic epistemic logics and branching time temporal logics is by way of tree models (forests) à la (van Benthem *et al.*, 2009) that are induced by repeated update model execution. See also (Wang, 2010).

5.3. *Computation*

For each of the presented protocols, we can ask what the time complexity is of expected termination, given a scheduling policy of interrogation. A fair scheduling policy is where prisoners are randomly selected for interrogation. A sequence produced with fair scheduling has the property that at any time, every prisoner will be interrogated again. It is unknown what the minimum is of expected termination for this policy—it is 3500 interrogations more or less. This has already been investigated with extensive simulations and trials, without a conclusive answer. Details are found in (Wu, 2002; Dehaye *et al.*, 2003; van Ditmarsch *et al.*, 2010).

6. 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.

7. References

Aucher G., Herzig A., "From DEL to EDL : Exploring the Power of Converse Events", *in* K. Mellouli (ed.), *ECSQARU, LNCS 4724*, Springer, pp. 199–209, 2007.

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

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

Fagin R., Halpern J. Y., Moses Y., Vardi M. Y., "Knowledge-based programs", *Distrib. Comput.*, vol. 10, num. 4, pp. 199–225, 1997.

Herzig A., Lima T. D., "Epistemic Actions and Ontic Actions: A Unified Logical Framework", *in* J. Sichman et al. (eds), *IBERAMIA-SBIA 2006, LNAI 4140*, Springer, pp. 409–418, 2006.

IBM Research, "Ponder this challenge", `http://domino.watson.ibm.com/Comm/wwwr_ponder.nsf/challenges/July2002.html`, 2002.

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

Renne B., Sack J., Yap A., "Dynamic Epistemic Temporal Logic", *in* X. He, J. Horty, E. Pacuit (eds), *Logic, Rationality, and Interaction. Proceedings of LORI 2009*, Springer, pp. 263–277, 2009. LNCS 5834.

Sack Y., Adding Temporal Logic to Dynamic Epistemic Logic, PhD thesis, Indiana University, Bloomington, USA, 2007.

van Benthem J., Gerbrandy J., Hoshi T., Pacuit E., "Merging frameworks for interaction", *Journal of Philosophical Logic*, vol. 38, pp. 491–526, 2009.

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

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

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

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

van Ditmarsch H., van Eijck J., Wu W., "One hundred prisoners and a lightbulb — logic and computation", *in* F. Lin, U. Sattler (eds), *Proceedings of KR 2010 Toronto*, 2010.

van Eijck J., "DEMO — A Demo of Epistemic Modelling", *in* J. van Benthem, D. Gabbay, B. Löwe (eds), *Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop*, Amsterdam University Press, pp. 305–363, 2007. Texts in Logic and Games 1.

Wang Y., Epistemic Modelling and Protocol Dynamics, PhD thesis, Universiteit van Amsterdam, 2010.

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

Wu W., "100 prisoners and a lightbulb", 2002, Manuscript.