One hundred prisoners and a lightbulb — logic and computation

Hans van Ditmarsch
D. Logic, F. Philosophy
University of Sevilla, Spain
Email: hvd@us.es

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

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

Abstract

This is a case-study in knowledge representation. 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 computational aspect is about average runtime termination under conditions of random ('fair') scheduling.

The paper consists of three parts. First, we present different versions of the puzzle, and their solution. 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 give the expected termination results of several protocols when assuming random scheduling.

This paper integrates the literature and presents novel contributions. Novel 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 computational results on Protocol 2 and results from manuscript (Wu 2002).

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)?

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 informal references before 2001. On `http://domino.watson.ibm.com/Comm/wwwr_ponder.nsf/challenges/July2002.html` it is mentioned that "this puzzle has been making the rounds of Hungarian mathematicians' parties" in a 23 prisoner version. See also (Dehaye, Ford, and Segerman 2003; Winkler 2004) and `http://wuriddles.com/`.

Knowledge Knowledge plays a crucial role in the formulation of the riddle as 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.

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. ⊣

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 (state of light, prisoner interrogated, …)

$$-1+0-1-2+1+0-$$

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.* ⊣

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 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 $2\cdot3-2=4$. 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.

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 $Pr : \{0,...,n\} \to [0,1]$ 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.

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. The game now becomes one of minimizing expected termination given random scheduling: for 100 prisoners, the expected termination of Protocol 1 is just under 30 years, but this can be reduced to about 10 (see later).

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 only saw the light off in stage I do nothing; the remaining non-counters act according to Protocol 1.* ⊣

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.)

Binary tokens protocol The binary tokens scheme generalizes the example in the previous paragraph. It was originally presented in (Wu 2002; Dehaye, Ford, and Segerman 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 mth 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).* ⊣

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).

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*.

Epistemic logic with epistemic and factual change Dynamic epistemic logics involving both epistemic and factual change have been proposed in (Baltag 2002; van Ditmarsch, van der Hoek, and Kooi 2005; van Ditmarsch 2006; van Benthem, van Eijck, and Kooi 2006; Herzig and Lima 2006; Kooi 2007; van Ditmarsch and Kooi 2008). We base our summary presentation on (van Ditmarsch and Kooi 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 $[U, e]\varphi$, 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 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. (Figures 1, 2, 3 illustrate event descriptions, an update model and its execution in the setting of an example.) The Appendix "Technical preliminaries" contains technical details of the logic.

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.

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

event	precond.	postcondition
$e_\emptyset$	if $\top$	then ϵ
$e_i^{\neg p}$	if $\neg p$	then $p := q_i \rightarrow p$ and $q_i := p \rightarrow q_i$
e_i^{p}	if p	then ϵ
$e_0^{\neg p}$	if $\neg p$	then ϵ
e_0^{p}	if p	then $p := \bot$

Figure 1: Pre- and postconditions of interrogation events

terrogated, 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.

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

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

- $e_\emptyset$: 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 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 $e \neq e_i^x$: $e_i^x \not\sim_i e$.

Nothing happens? The 'nothing happens' event $e_\emptyset$ ensures that the prisoners remain uncertain of the state of the light, even when they are not interrogated 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 $e_0^{\neg p}$ did not take place, one of the events $e_1^{\neg p}, \ldots, e_{n-1}^{\neg p}$ 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. This is not an 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.

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

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

They have the same effect as the single event $e_i^{\neg p}$. Let us show this now. Assume that p is false. According to $p := q_i \to p$, if q_i is false, then $q_i \to p$ is true, so p becomes true; whereas if q_i is already true, $q_i \to p$ is false so p remains false. According to $q_i := p \to q_i$, q_i becomes true if it was false, or remains true if it was already true. We prefer the single event $e_i^{\neg p}$, 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 event e_0^p as 'if p then $p := p \to \bigwedge_{i>0} q_i$', thus ensuring that the light remains on if $\bigwedge_{i>0} q_i$ is true.

Public announcement 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 I_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 I_n we cannot select or indicate the terminating run). But we can formulate this on a metalevel. The initial conditions are that all of $p, q_1, \ldots, q_n$ 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 $(I_n, e_i^{\neg p})$, and as the only assignment changing the value of q_i occurs in that event, namely $q_i := p \to q_i$, its value remains true once it has become true. Also, because of assignment $p := q_i \to p$, 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 (I_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 $K_0 \bigwedge_{i>0} q_i$, which is the termination condition.

DEMO 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 (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. For details, see the Appendix on DEMO.

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

$$
e_i : \text{if } \top \text{ then } p := q_i \to p \text{ and } q_i := p \to q_i
$$

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 '$p := q_i \to p$ and $q_i := p \to q_i$': if p is true, then according to $p := q_i \to p$ the new value of p remains true, and according to $q_i := p \to q_i$ the new value of q_i remains the old value of q_i.

The update model I_3^0 for the case of three prisoners is depicted is Figure 2, and the execution in initial epistemic model $\mathcal{I}_3^0$ of Protocol 1, consisting of repeated execution of update I_3^0 until termination, is depicted in Figure 3. We gave it in a concise graph representation corresponding to the tree-model generated by the initial state and the update
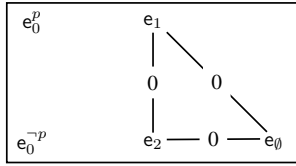
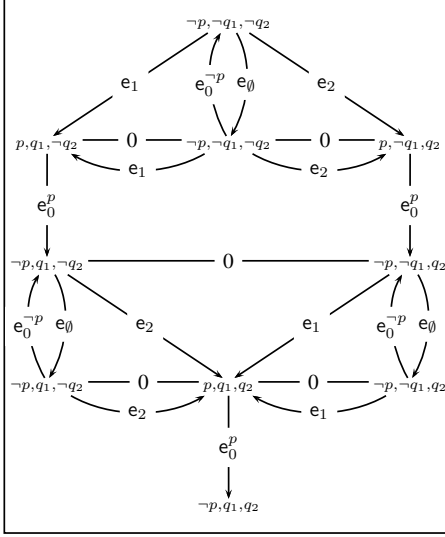Figure 2: Update model for the interrogation of three prisoners

Figure 3: Depicting the execution of actions for three prisoners

model, and we identified bisimilar states. The states are indicated by an atomic description. The initial epistemic state is the top state. 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.

The crucial 'nothing happens' event $e_\emptyset$ ensures that uncertainty about the state of the light arises straight after the prisoners leave the dining area, by way 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 $e_0^{\neg p}$, the transition back to the top state of the model.

Logic of other protocols 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: $\bigvee_{j=0}^{n-1} K_j \bigwedge_{i>0} q_i$.

In Protocol 3 the initial state of the light is unknown. 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

Asynchronous behaviour For the synchronized setting where the interval between interrogations is known, the logical modelling of the problem becomes simpler, because dynamic epistemic logic assumes synchronization (van Benthem et al. 2009). In the update model I_n, simply remove the 'nothing happens' event $e_\emptyset$. This non-deterministic update model can be said to represent random scheduling, namely between n executable events $e_0^p, e_1^p, ..., e_{n-1}^p$ if the light is on, or between n executable events $e_0^{\neg p}, e_1^{\neg p}, ..., e_{n-1}^{\neg p}$ if the light is off, respectively.

This suggests that the standard solution is an asynchronous version, and that the 'nothing happens' event $e_\emptyset$ makes the difference. We think that this is indeed the case, and that this is of general interest for dynamic epistemics. 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. He therefore cannot tell, so to speak, if the clock has ticked five or two times (or any other amount): he does not know what time it is.

Growth of common knowledge? For the original setting of the riddle, *common knowledge of factual propositions cannot grow*, until the moment the announcement is made that everybody has been interrogated. The proof is simple. Let φ be a boolean proposition of interest that is not initially commonly known. (Examples are p—the light is on, and ψ_k—at least k prisoners have been interrogated.) Assume that φ is not yet commonly known. Therefore ($S5$ property) all prisoners consider it possible that φ is not 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 considers it possible that nothing happened, and therefore still considers it possible that φ is not commonly known. If one of the last two happened, take a non-counting prisoner, e.g. prisoner 1, then 1 considers it possible that nothing happened, and therefore still considers it possible that φ is not commonly known. The result can be extended from boolean to modal formulas, but we do not know if it holds for all modal formulas.

If the interval between interrogations is known, common knowledge can grow. For example, after one day it is common knowledge that the light is on. And it can shrink: on day two this knowledge has been lost again. Some common knowledge is preserved after being obtained: after one day it is common knowledge that at least one prisoner has been interrogated, and that remains common knowledge forever.

Computation

For each of the presented protocols, we can ask what the time complexity is of expected termination, given a scheduling policy of interrogation. For the original number of 100 prisoners and for the scheduling policy where prisoners are randomly selected for interrogation, this question has been addressed. (This policy is fair: Consider an interrogation sequence produced with it. With probability 1, this sequence has the property that at any time, every prisoner will be interrogated again—fairness.) It is unknown what the minimum is of expected termination.

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 $1, 0, 2, 0, 3, 0, ...99, 0$; and the expected duration for all prisoners to be interrogated once, given random scheduling, is roughly $100 \ln 100 = 460$ days. For a prisoner to learn that all prisoners have been interrogated, takes much longer.

Expected termination for protocol 1   Let $n = 100$ and consider again Protocol 1. A single interrogation per day takes place. For the light to be turned on, a non-counter has to be interrogated. Assuming 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, probability $\frac{98}{100}$, etc. The expectations of those events are $\frac{100}{99}, \frac{100}{1}, \frac{100}{98}, \ldots$. Their sum is easily computed:

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

This amounts to approximately 28.5 years.

Expected termination for protocol 3   In Protocol 3 the non-counters also keep count, namely of sequences off/on, just in case they can announce success before the counter. Let us call a non-counter *lucky* if he announces success before the counter. For the case of three prisoners, keeping count makes sense. We have prisoners 0, 1, and 2. Distinguish the case that 1 is interrogated ('called') before 2 from the case that 2 is interrogated before 1. In the first case, 1 gets lucky if, after 0, he is called *before* 2, even odds, and after that is called *before* 0, even odds again: $0.5 \cdot 0.5 = 0.25$. In case 2 is initially called before 1, 1 has to be called before 0, even odds, and after 0 has then been called, 1 has to be called before 2, even odds again, so this is also a probability of 0.25. To this we have to add the equal probability that 2 is lucky. Together this gives a 50% chance that 1 or 2 announces success before 0. For larger n it is extremely rare that a non-counter gets lucky. A long computation (omitted) demonstrates that an *upper* bound for that probability, for $n = 100$, is 5.63×10^{-72}.

Expected termination for protocol 5   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. This is the proof:

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

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 5 is then about 25 years, which we can informally show as follows:

What counts most is the obligation for the counter to be interrogated again, and again, each time with an expectancy of 100 days. So 11 days less, means 1100 days off the previous estimate, plus a bit extra given the non-counters that have no job to perform. This shaves off nearly four years from prison. This is therefore the result of $\sum_{i=1}^{88}\left(\frac{100}{i} + \frac{100}{1}\right)$ instead of, as before, $\sum_{i=1}^{99}\left(\frac{100}{i} + \frac{100}{1}\right)$, plus of course the 13 days to be expected for stage I, but that is negligable.

The expected duration of protocols for 100 prisoner can be much further reduced, down to about 10 years (see (Wu 2002)). This is e.g. the case for binary tokens protocol with the point sequence as below. It is unknown whether much less than 10 years is possible.

Expected termination for protocol 6   Given Protocol 6 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. We now prove using a coupon collection analysis that the following sequence has an average runtime of $O(n(\ln n)^2)$.

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

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

Proof of 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 ith 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 mth bit of a prisoner's mental count is 1, then he can only clear this bit by turning on the light bulb in the mth 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 kth 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 jth coupon at the kth stage, where $j \in \{1, \ldots, n/2^k\}$, 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 kth 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:

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

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

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

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

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

This finishes the proof.

Restarting the (P_k) **sequence** 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.

Further research

Puzzle We keep discovering and designing more versions of the riddle. Protocol 4 was a recent addition. The authors of (Dehaye, Ford, and Segerman 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.

Logic For our modelling purposes, the logic we presented has restrictions: we cannot refer to past events in preconditions, we can simulate but not really express asynchronous behaviour (although this largely depends on the meaning of the word 'really'), 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. Works reporting progress in this area are (Sack 2007; Aucher and Herzig 2007; Renne, Sack, and Yap 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. 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 (van Benthem et al. 2009) that are induced by repeated update model execution.

Computation 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.

Acknowledgements

References

Aucher, G., and Herzig, A. 2007. From DEL to EDL : Exploring the power of converse events. In Mellouli, K., ed., *ECSQARU, LNCS 4724*, 199–209. Springer.

Baltag, A. 2002. A logic for suspicious players: Epistemic actions and belief updates in games. *Bulletin of Economic Research* 54(1):1–45.

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

Herzig, A., and Lima, T. D. 2006. Epistemic actions and ontic actions: A unified logical framework. In Sichman, J., et al., eds., *IBERAMIA-SBIA 2006, LNAI 4140*, 409–418. Springer.

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

Renne, B.; Sack, J.; and Yap, A. 2009. Dynamic epistemic temporal logic. In He, X.; Horty, J.; and Pacuit, E., eds., *Logic, Rationality, and Interaction. Proceedings of LORI 2009*, 263–277. Springer. LNCS 5834.

Sack, Y. 2007. *Adding Temporal Logic to Dynamic Epistemic Logic*. Ph.D. Dissertation, Indiana University, Bloomington, USA.

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

van Benthem, J.; van Eijck, J.; and Kooi, B. 2006. Logics of communication and change. *Information and Computation* 204(11):1620–1662.

van Ditmarsch, H., and Kooi, B. 2008. Semantic results for ontic and epistemic change. In Bonanno, G.; van der Hoek, W.; and Wooldridge, M., eds., *Logic and the Foundations of Game and Decision Theory (LOFT 7)*, Texts in Logic and Games 3. Amsterdam University Press. 87–117.

van Ditmarsch, H.; van der Hoek, W.; and Kooi, B. 2005. Dynamic epistemic logic with assignment. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 05)*, 141–148. New York: ACM Inc.

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

van Eijck, J. 2007. DEMO — a demo of epistemic modelling. In van Benthem, J.; Gabbay, D.; and Löwe, B., eds., *Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop*. Amsterdam University Press. 305–363. Texts in Logic and Games 1.

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

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

Appendix: 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 $\mathscr{L}$ is a quadruple $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{post})$ where

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

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

using the expression

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

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

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

- $V'(p) = \{(t, f) \mid (M, t) \models \mathsf{post}(f)(p)\}$.

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 $\mathscr{L}$ is given by the following BNF:

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

where $p \in P$, $a \in N$, $B \subseteq N$, 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 $\bigwedge_{f \in \mathsf{U}}[\mathsf{U}, f]\varphi$.

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

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

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.

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

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

Appendix: DEMO implementation

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

```
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,Neg Top)]))` is as 'If p then $p := \bot$' 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.