Model Checking Logic Puzzles

Hans van Ditmarsch[*]   Ji Ruan[†]
hans@cs.otago.ac.nz   J.Ruan@csc.liv.ac.uk

[*]Computer Science, University of Otago, New Zealand

[†]Computer Science, University of Liverpool, United Kingdom

Résumé :
Dans les puzzles épistémiques les annonces d'ignorance, ou des séquences de tels annonces, souvent résultent en connaissances. Nous présentons le puzzle 'Quelle Somme ?', et le modèlisent dans la logique des annonces publiques – un langage logique avec des opérateurs dynamiques et épistémiques. La solution du puzzle est controlée avec la programme de vérification DEMO.

Mots-clés : communications multi-agent, vérification des modèles, logique dynamique épistémique, annonce publique

Abstract:
A common theme in logic puzzles involving knowledge and ignorance is that announcements of ignorance may eventually result in knowledge. We present the 'What Sum' riddle. It is modelled in public announcement logic, a modal logic with both dynamic and epistemic operators. We then solve the riddle in the model checker DEMO.[1]

Keywords: agent communication, model checking, dynamic epistemic logic, public announcement

1   Introduction

The following riddle (transcribed in our terminology) appeared in Math Horizons in 2004, as 'Problem 182' in a regular problem section of the journal, edited by A. Liu [8].

Each of agents Anne, Bill, and Cath has a positive integer on its forehead. They can only see the foreheads of others. One of the numbers is the sum of the other two. All the previous is common knowledge. The agents now successively make the truthful announcements:

i. Anne: "I do not know my number."

ii. Bill: "I do not know my number."

iii. Cath: "I do not know my number."

iv. Anne: "I know my number. It is 50."

What are the other numbers?

You know your own number if and only if you know which of the three numbers is the sum. This 'What is the sum?', from now on 'What Sum', riddle combines features from wisemen or Muddy Children puzzles [12] with features from the Sum and Product riddle [3, 10]. A common feature in such riddles is that we are given a multi-agent interpreted system, and that successive announcements of ignorance finally result in its opposite, typically factual knowledge. In a global state of an interpreted system [2] each agent or processor has a local state, and there is common knowledge that each agent only knows its local state, and what the extent is of the domain. If the domain consists of the full cartesian product of the sets of local state values, it is common knowledge that agents are ignorant about others' local states. In that case an ignorance announcement has no informative value. For ignorance statements to be informative, the domain should be more restrictive than the full cartesian product; and this is the case

in all such riddles. As in Muddy Children, we do not take the 'real' state of the agent (the number on its forehead) as its local state, but instead the information seen on the foreheads of others (the other numbers). This change of perspective is, clearly, inessential. 'Sum and Product'[2] is also about numbers, and even about sums of numbers, and the announcements are similar. But the structure of the background knowledge is very different (which will become clearer after introducing the logic to describe both riddles).

Other epistemic riddles involve cryptography and the verification of information security protocols ('Russian Cards', see [19]), or involve communication protocols with private signals involving diffusion of information in a distributed environment ('100 prisoners and a lightbulb', see [21]).

The understanding of such riddles is facilitated by the availability of suitable specification languages. For 'What Sum' we propose the logic of public announcements, wherein succinct descriptions in the logical language are combined with convenient relational structures on which to interpret them. We also benefit from the availability of verification tools, to aid interpreting such descriptions on such structures. In our case we have used DEMO, an epistemic model checker developed by Van Eijck (see `homepages.cwi.nl/~jve/demo/` and [20]). Some adjustments are required (we need a finite version of the model) to make this model checking work. This results in possibly in-

---

[2]*A says to S and P: I have chosen two integers x, y such that $1 < x < y$ and $x + y \leq 100$. In a moment, I will inform S only of $s = x + y$, and P only of $p = xy$. These announcements remain private. You are required to determine the pair (x, y).*

   He acts as said. The following conversation now takes place:

    i. P says: "I do not know it."

    ii. S says: "I knew you didn't."

    iii. P says: "I now know it."

    iv. S says: "I now also know it."

Determine the pair (x, y).[3, translated]

---

teresting versions of the riddle.

Even though such riddles are often pivotal to the development and spreading of a specialisation area—who doesn't know about the 'Muddy Children' puzzle?—the detailed and rockbottom analysis of their highly proceduralised features is not necessarily considered a serious enough pursuit to increase our understanding of multiagent system dynamics. May our original analysis of 'What Sum' be seen as a worthy contribution.

Section 2 provides an introduction into public announcement logic, and in Section 3 we analyse 'What Sum' in this logic. Section 4 'preprocesses' the riddle for model checking and discusses some versions of the riddle. In Section 5 we introduce DEMO, and in Section 6 we specify and verify a finite version of the riddle in that model checker.

2 Public Announcement Logic

Public announcement logic is a dynamic epistemic logic and is an extension of standard multi-agent epistemic logic. Intuitive explanations of the epistemic part of the semantics can be found in [2, 19]. We give a concise overview of, in that order, the language, the structures on which the language is interpreted, and the semantics.

Given are a finite set of agents N and a finite or countably infinite set of atoms P. The language of public announcement logic is inductively defined as

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_n\varphi \mid C_B\varphi \mid [\varphi]\psi$$

where $p \in P$, $n \in N$, and $B \subseteq N$ are arbitrary. Other propositional and epistemic operators are introduced by abbreviation. For $K_n\varphi$, read 'agent n knows formula φ'. For example, if Anne knows that her number is 50, we can write $K_a 50_a$, where a stands for Anne and some set of atomic propositions is assumed that contains 50_a

to represent 'Anne has the number 50.' For $C_B\varphi$, read 'group of agents B commonly know formula φ'. For example, we have that $C_{abc}(20_b \rightarrow K_a 20_b)$: it is common knowledge to Anne, Bill, and Cath, that if Bill's number is 20, Anne knows that (because she can see Bill's number on his forehead)—instead of $\{a, b, c\}$ we often write abc. For $[\varphi]\psi$, read 'after public announcement of φ, formula ψ (is true)'. For example, after Anne announces "(I know my number. It is 50.)" it is common knowledge that Bill's number is 20. This is formalised as $[K_a 50_a]C_{abc}20_b$.

The basic structure is the epistemic model. This is a Kripke structure, or model, wherein all accessibility relations are equivalence relations. An *epistemic model* $M = \langle S, \sim, V \rangle$ consists of a *domain* S of (factual) *states* (or 'worlds'), *accessibility* $\sim : N \rightarrow \mathcal{P}(S \times S)$, where each $\sim(n)$ is an equivalence relation, and a *valuation* $V : P \rightarrow \mathcal{P}(S)$. For $s \in S$, (M, s) is an *epistemic state* (also known as a pointed Kripke model). For $\sim(n)$ we write $\sim_n$, and for $V(p)$ we write V_p. Accessibility $\sim$ can be seen as a set of equivalence relations $\sim_n$, and V as a set of valuations V_p. Given two states s, s' in the domain, $s \sim_n s'$ means that s is indistinguishable from s' for agent n on the basis of its information. For example, at the beginning of the riddle, triples $(2, 14, 16)$ and $(30, 14, 16)$ are indistinguishable for Anne but not for Bill nor for Cath. Therefore, assuming a domain of natural number triples, we have that $(2, 14, 16) \sim_a (30, 14, 16)$. The group accessibility relation $\sim_B$ is the transitive and reflexive closure of the union of all accessibility relations for the individuals in B: $\sim_B \equiv (\bigcup_{n \in B} \sim_n)^*$. This relation is used to interpret common knowledge for group B. Instead of '$\sim_B$ equivalence class' ($\sim_n$ equivalence class) we write B-class (n-class).

For the semantics, assuming an epistemic model $M = \langle S, \sim, V \rangle$:

$$
\begin{array}{lll}
M, s \models p & \text{iff} & s \in V_p \\
M, s \models \neg\varphi & \text{iff} & M, s \not\models \varphi \\
M, s \models \varphi \wedge \psi & \text{iff} & M, s \models \varphi \text{ and } M, s \models \psi \\
M, s \models K_n\varphi & \text{iff} & \text{for all } t \in S : \\
 & & s \sim_n t \text{ implies } M, t \models \varphi \\
M, s \models C_B\varphi & \text{iff} & \text{for all } t \in S : \\
 & & s \sim_B t \text{ implies } M, t \models \varphi \\
M, s \models [\varphi]\psi & \text{iff} & M, s \models \varphi \text{ implies} \\
 & & M|\varphi, s \models \psi
\end{array}
$$

where model $M|\varphi = \langle S', \sim', V' \rangle$ is defined as

$$
\begin{array}{lll}
S' & = & \{s' \in S \mid M, s' \models \varphi\} \\
\sim'_n & = & \sim_n \cap (S' \times S') \\
V'_p & = & V_p \cap S'
\end{array}
$$

The dynamic modal operator $[\varphi]$ is interpreted as an epistemic state transformer. Announcements are assumed to be truthful, and this is commonly known by all agents. Therefore, the model $M|\varphi$ is the model M restricted to all the states where φ is true, including access between states. The dual of $[\varphi]$ is $\langle\varphi\rangle$: $M, s \models \langle\varphi\rangle\psi$ iff $M, s \models \varphi$ and $M|\varphi, s \models \psi$. Formula φ is valid on model M, notation $M \models \varphi$, iff for all states s in the domain of M: $M, s \models \varphi$. Formula φ is valid, notation $\models \varphi$, iff for all models M: $M \models \varphi$.

A proof system for this logic is presented, and shown to be complete, in [1], with precursors—namely for public announcement logic *without* common knowledge— in [15, 5]. A concise completeness proof is given in [19]. The logic is decidable both with and without common knowledge [15, 1]. Results on the complexity of both logics can be found in [9]. The original [15] also contains a version of the semantics (no completeness results) with 'know-value'-operators that can be said to formalise infinitary conjunctions (or disjunctions), including announcements of such formulas with corresponding restriction of the domain to those states where the formula is true. To analyse 'What Sum' we need to refer to that extension (that we prefer to leave informal for the sake of the exposition).

In public announcement logic, not all formulas remain true after their announcement, in other words, $[\varphi]\varphi$ is *not* a principle of the logic. Some formulas involving epistemic operators become **false** after being announced! For a simple example, consider that Bill were to tell Anne (truthfully) at the initial setting of the riddle: "Your number is 50 but you don't know that." Interpreting 'but' as a conjunction, this is formalised as $50_a \wedge \neg K_a 50_a$. After the announcement, Anne knows that her number is 50: $K_a 50_a$. Therefore the announced formula, that was true before the announcement, has become false after the announcement. In the somewhat different setting that formulas of form $p \wedge \neg K_n p$ cannot be consistently known this phenomenon is called the Moore-paradox [11, 7]. In the underlying dynamic setting it has been described as an *unsuccessful update* [5, 19]. Similarly, ignorance statements in 'What Sum' such as Anne saying that she does not know her number, may in due time lead to Anne knowing her number, the opposite of her ignorance.

3 Formalisation of 'What Sum'

The set of agents $\{a, b, c\}$ represent Anne, Bill and Cath, respectively. Atomic propositions i_n represent that agent n has natural number i on its forehead. Therefore the set of atoms is $\{i_n \mid i \in \mathbb{N}^+ \text{ and } n \in \{a, b, c\}\}$.

If Anne sees (knows) that Bill has 20 on his forehead and Cath 30, we describe this as $K_a(20_b \wedge 30_c)$. If an upper bound max for all numbers *were* specified in the riddle, the number of states would be finite and "knowing the others' numbers" would be described as $\bigvee_{y,z \leq \mathsf{max}} K_a(y_b \wedge z_c)$. For model checking it is relevant to point out that this expression is equivalent to $\bigwedge_{y,z \leq \mathsf{max}}(y_b \wedge z_c) \rightarrow K_a(y_b \wedge z_c)$, given that different Bill/Cath number pairs are mutually exclusive, and using standard validities for the logic. The latter form is 'cheaper' to model check than the for-

mer, because the truth of the boolean condition in the conjuncts of the latter can be determined in a given state, whereas an epistemic statement requires checks in that agent's entire equivalence class.

For 'What Sum', Anne seeing the numbers of Bill and Cath is therefore described as the *infinitary* $\bigvee_{y,z \in \mathbb{N}^+} K_a(y_b \wedge z_c)$, and Anne saying: "I don't know my number" is similarly described as $\neg \bigvee_{x \in \mathbb{N}^+} K_a x_a$ (or $\bigwedge_{x \in \mathbb{N}^+}(x_a \rightarrow \neg K_a x_a)$). Infinitary descriptions are, unlike infinitely large models, not permitted in this (propositional) logic. Our model checking results will be for a finite version of the riddle.

The epistemic model $\mathcal{T} = \langle S, \sim, V \rangle$ is defined as follows, assuming positive natural numbers x, y, z.

$$S \equiv \{(x, y, z) \mid x = y + z \text{ or } y = x + z \text{ or } z = x + y\}$$

$$\begin{aligned}
(x, y, z) &\sim_a (x', y', z') && \text{iff} && y = y' \text{ and } z = z' \\
(x, y, z) &\sim_b (x', y', z') && \text{iff} && x = x' \text{ and } z = z' \\
(x, y, z) &\sim_c (x', y', z') && \text{iff} && x = x' \text{ and } y = y' \\
(x, y, z) &\in V_{x_a} \\
(x, y, z) &\in V_{y_b} \\
(x, y, z) &\in V_{z_c}
\end{aligned}$$

The fine-structure of the epistemic model $\mathcal{T}$ is not apparent from its formal definition. A relevant question is what the background knowledge is that is available to the agents, i.e., what the abc-classes in the model are (an abc-class, or $\{a, b, c\}$ equivalence class, of a state s in the model consists of all states t such that $s \sim_{\{a,b,c\}} t$, where $\sim_{\{a,b,c\}} = (\sim_a \cup \sim_b \cup \sim_c)^*$, as above). Such a computation was performed by Panti [14] for 'Sum and Product' (see footnote 2), which revealed three classes: either (in two of the three classes) the solution of the problem is already common knowledge in the initial state, or the agents commonly know that the sum of the numbers is at least 7. This means that in 'Sum and Product' not very much is commonly known. In contrast, a model $\mathcal{T}$ for 'What Sum' has a very different structure, with many more common knowledge classes. It is therefore quite informative to

know what they are, and we will describe them in detail.

An *abc*-class in $\mathcal{T}$ can be visualised as an infinite binary tree. The depth of the tree reflects the following order on number triples in the domain of $\mathcal{T}$: $(x, y, z) > (u, v, w)$ iff ($x > u$ and $y = v$ and $z = w$) or ($x = u$ and $y > v$ and $z = w$) or ($x = u$ and $y = v$ and $z > w$). If $(x, y, z) > (u, v, w)$ according to this definition, (x, y, z) is a child of (u, v, w) in that tree. Every node except the root has one predecessor and two successors, as in Figure 1.
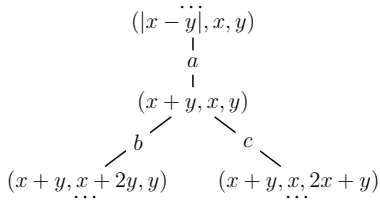

Figure 1: Modulo agent symmetry, all parts of the model $\mathcal{T}$ branch as here. Arcs connecting nodes are labelled with the agent who cannot distinguish those nodes.

The root of each tree has label $(2x, x, x)$ or $(x, 2x, x)$ or $(x, x, 2x)$. Differently said, given three natural numbers such that one is the *sum* of the other two, replace that sum by the *difference* of the other two; one of those other two has now become the sum; if you repeat the procedure, you *always* end up with two equal numbers and their sum. An agent who sees two equal numbers, immediately infers that its own number must be their sum (twice the number that is seen), because otherwise it would have to be their difference 0 which is not a positive natural number. It will be obvious that: the structure truly is a forest (a set of trees), because each node only has a single parent; all nodes except roots are triples of three *different* numbers; and all trees are infinite. All *abc*-trees are isomorphic modulo (i) a multiplication factor for the numbers occurring in the arguments of the node labels, and modulo (ii) a permutation of arguments and a cor-

responding swap of agents, i.e., swap of arc labels. For example, the numbers occurring in the tree with root $(6, 3, 3)$ are thrice the corresponding numbers in the tree with root $(2, 1, 1)$; the tree with root $(2, 1, 1)$ is like the tree for root $(1, 2, 1)$ by applying permutation (213) to arguments and (alphabetically ordered) agent labels alike. The left side of Figure 3 shows the trees with roots $(2, 1, 1)$, $(1, 2, 1)$, and $(1, 1, 2)$. For simplicity, we write 211 instead of $(2, 1, 1)$, etc. In the left tree, for Bill $(2, 1, 1)$ is indistinguishable from $(2, 3, 1)$ wherein his number is the sum of the other two instead of their difference; for Anne triple $(2, 3, 1)$ is indistinguishable from $(4, 3, 1)$, etc.

Processing Announcements   The result of an announcement (whether described infinitary or not) is the restriction of the model to all states where the announcement is true. We can also apply this to the ignorance announcements of agents in 'What Sum'. Consider an *abc*-tree T in $\mathcal{T}$. Let n be an arbitrary agent. Either the root of T is a singleton n-class, or all its n-classes consist of two elements: a two-element class represents the agent's uncertainty about its own number. An ignorance announcement by agent n in this riddle corresponds to removal of all singleton n-classes from the model $\mathcal{T}$. This means that *some* of the model's trees are split into two subtrees (with both children of the original root now roots of infinite trees).

An ignorance announcement may have very different effects on *abc*-classes that are the same modulo agent permutations. For example, given *abc*-classes in $\mathcal{T}$ with roots 121, 112, and 211, the effect of Anne saying that she does not know her number *only* results in elimination of 211, as only the first *abc*-class contains an *a*-singleton. Given 211, Anne knows that she has number 2 (as 0 is excluded). But triple 112 she cannot distinguish from 312, and 121 not from 321. Thus one proceeds with all three announcements. See also Figure 2.
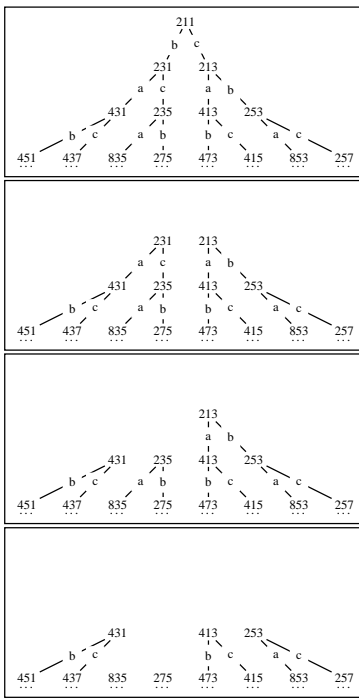
Figure 2 (left column):

Figure 2: The results of three ignorance announcements on the abc-class with root $(2, 1, 1)$.

Figure 3 (right column):

Figure 3: On the left, abc-classes of the model $\mathcal{T}$ with root 211, 121, and 112. Any other abc-class is isomorphic to one of these, modulo a multiplication factor. The results of the (combined) three ignorance announcements on those abc-classes are on the right. The triples in bold are those where Anne knows her number.

Solving the riddle We have now sufficient background to solve the riddle. We apply the successive ignorance announcements to the three classes with roots $(2, 1, 1)$, $(1, 2, 1)$, and $(1, 1, 2)$, determine the triples wherein Anne knows the numbers, and from those, wherein Anne's number divides 50. See Figure 3—note that in triple $(8, 3, 5)$ Anne also knows her number: the alternative $(2, 3, 5)$ wherein her number is 2 has been eliminated by Cath's, last, ignorance announcement. The *unique* triple wherein Anne's number divides 50 is $(5, 2, 3)$. In other words, the unique abc-tree in the *entire* model $\mathcal{T}$ where Anne knows that she has 50 after the three ignorance announcements, is the one with root $(10, 20, 10)$. The solution to the riddle is therefore that Bill has 20 and Cath has 30. After the three announcements in the abc-class with root $(10, 20, 10)$, the triple $(50, 20, 30)$ remains wherein Anne knows that Bill has 20 and Cath 30.
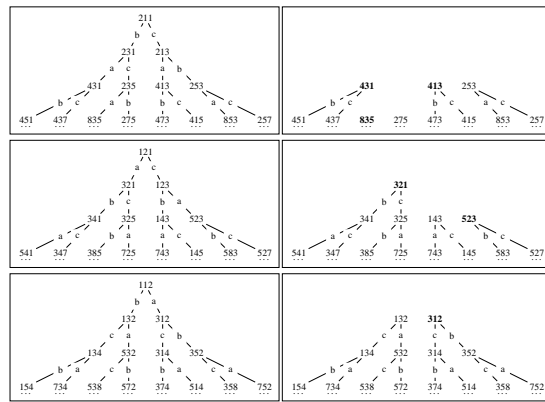
The original riddle could have more restrictive: in the quoted version [8] it is *not* required to determine who holds which other number, but as we have seen this can also be determined. It also occurred to us that the original riddle could have been posed differently (and we tend to think, far more elegantly) as follows:

Each of agents Anne, Bill, and Cath has a positive integer on its forehead. They can only see the foreheads of others. One of the numbers is the sum of the other two. All the previous is common knowledge. The agents now successively make the truthful announcements:

  i. *Anne: "I do not know my number."*

  ii. *Bill: "I do not know my number."*

  iii. *Cath: "I do not know my number."*

What are the numbers, if Anne now knows her number and if all numbers are prime?

Consulting Figure 3, it will be obvious that the answer should be: '5, 2, and 3'.

4 Towards Model Checking

To be able to use a model checker we need a finite approximation of the model. Suppose we use an upper bound max for the numbers. Let $\mathcal{T}^{\mathsf{max}}$ be the corresponding epistemic model. An abc-tree is now cut at the depth where nodes (x, y, z) occur such that the sum of two of the arguments x, y, z exceeds max. This finite approximation may not seem a big deal but it makes the problem completely different: abc-classes will not just have *roots* wherein the agent may know his number (because the other numbers are equal) but will also have *leaves* wherein the agent may know his number (because the sum of the other two numbers exceeds max). In other words, we have far more singleton equivalence classes. Let max $= 10$. Node $(2, 5, 7)$ in the abc-class with root $(2, 1, 1)$ has only a b-child $(2, 9, 7)$ and a c-parent $(2, 5, 3)$, and not an a-child, as $5 + 7 = 12 >$ max. So Anne immediately knows that her number is 2. All roots $(2x, x, x)$ with $3x >$ max form singleton abc-classes in $\mathcal{T}^{\mathsf{max}}$, for the same reason.

In such models it is no longer the case that all equivalence classes are isomorphic modulo a multiplication factor and swapping of agent labels. For a given upper bound max we still have that, if $x > y$, the abc-class T with root $(2x, x, x)$ is a prefix (in a partially ordered sense) of the abc-class T' with root $(2y, y, y)$, which implies that $T \subseteq T'$ (modulo a factor $\frac{y}{x}$ for numbers occurring in T). For different upper bounds max, max$'$ we have that (literally) $\mathcal{T}^{\mathsf{max}} \subseteq \mathcal{T}^{\mathsf{max}'}$ iff max $\leq$ max$'$.

Under these circumstances it is less clear what constitutes an exhaustive search of 'all possibilities that remain after an announcement'. Fortunately, we are now talking about *formal* announcements in the language of public announcement logic. The following non-trivial result is essential. Let T, T' be different epistemic models $\mathcal{T}$ for 'What Sum' (i.e., for different upper bounds max) or, modulo a multiplication factor, different abc-classes in a given $\mathcal{T}$ model.

If $T \subseteq T'$ and $\vec{\varphi}$ is a sequence of ignorance announcements executable in both T and T', then $T|\vec{\varphi} \subseteq T'|\vec{\varphi}$.

The proof is simple, and by induction on the number of such announcements. Consider a next ignorance announcement ψ being made, by agent n. As said, it removes singleton equivalence classes for that agent. If $T \subseteq T'$ it may be that some singleton n-classes in T were two-state n-classes in T'. These will therefore be omitted when executing the announcement of ψ in T, whereas they would have been preserved when executing the same announcement in T'. There are no other differences in execution: all n-classes that were singleton in both T and T' will be omitted anyway as a result of the ψ-announcement. Therefore, we still have that $T|\psi \subseteq T'|\psi$.

This may seem obvious. But it is far from that: for arbitrary $M' \subseteq M$ and arbitrary φ we do *not* have that $M'|\varphi \subseteq M|\varphi$. Let us give a counterexample. Given agents a, b and state variables p, q (in 10 p is true and q is false) consider the (two-state) model $M' = 11|a|10$, which is a restriction of the (three state) model $M = 11|a|10|b|01$. Consider $\varphi = K_b q \vee K_b \neg q$, for 'Bill knows whether q.' Then $M'|\varphi = M'$, whereas $M|\varphi$ is the singleton model consisting of state 11 wherein a and b have common knowledge of p and q. Therefore $M' \subseteq M$ but $M'|\varphi \nsubseteq M|\varphi$.

Apart from having an upper bound we discuss one other, less essential, change: suppose we start counting from 0 instead of 1. In that case each abc-equivalence class with root $(2x, x, x)$ is extended with one more node: the new root $(0, x, x)$ is indistinguishable from $(2x, x, x)$ for Anne. An agent who sees a 0, infers that his number

must be the other number that (s)he sees. If there is a 0, two of the three agents see that. Therefore, the root has just one child $(2x, x, x)$; if the triple is $(0, x, x)$ Bill and Cath know that their number is x.[3]

The abc-class with root 011 from the epistemic model $\mathcal{T}_0^{10}$ (upper bound 10, lower bound 0) is displayed on the left in Figure 4. The result of the three ignorance announcements is displayed on the right. We can now investigate different versions of the problem. The model checker is then helpful because some versions are hard to verify with pencil and paper, or mere mental computation. For example, we considered the version: If $0 \leq x, y, z \leq$ max, for which values of max does Anne *always* know the numbers after the three announcements? This range is $8 \leq$ max ≤ 13 (so, for 7 not all three announcements can be made truthfully, and for 14 it may be that Anne does not know her number) and this includes max $= 10$. Figure 4 shows that from abc-class with root 011 the triples 211 and 213 remain. In both cases Anne knows her number. Similar computations show that from the abc-classes with root 101 and 110 no triples remain. In other words, the announcements could not all three have been made (truthfully) if the number triple occurs in either of those two classes. Using the properties of inclusion for different abc-classes, we have now ruled out all classes of type $x0x$ and $xx0$ and only have to check other classes of type $0xx$. From class 022, the triples 242 and 246 remain after the three announcements (and the ones with root 033 and beyond are empty again). Therefore, whatever the numbers, Anne now

---

[3]Suppose there is no upper bound but 0 is still allowed—every audience being presented with this riddle for positive integers contains at least one person asking if 0 is allowed. This is an interesting variation. Anne will still learn her own number if it is 50 from the three ignorance announcements, but the reader ('problem solver') can now no longer deduce Bill's and Cath's number in that case: these can now also be 25 and 25. The reader should be able to determine this easily by contemplating Figure 3. From the models resulting from the three ignorance announcements, only *one* now looks different. Which one?
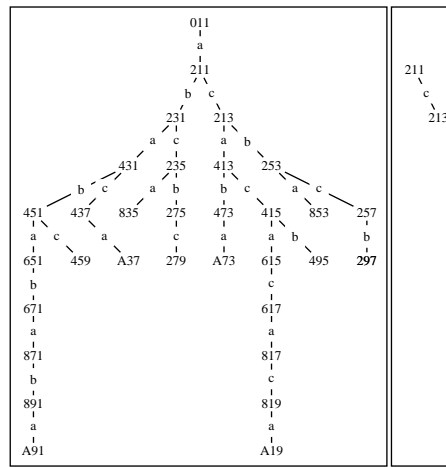

Figure 4: The abc-class with root 011 in model $\mathcal{T}_0^{10}$, and the result of three ignorance announcements. The horizontal order of branches has no meaning. Symbol A represents 10.

knows her number. But the problem solver cannot determine what that number is (it may be 1, or it may be 2) and also cannot determine what the other numbers are, not even if it is also known what Anne's number is (if it is 1, the other numbers may be 2 and 1, or 2 and 3; and similarly if it is 2).

5 Model Checker DEMO

Epistemic model checkers with dynamic facilities have been developed to verify properties of interpreted systems, knowledge-based protocols, and various other multi-agent systems. Examples are MCK [4], MCMAS [16], and recent work by Su [17]. All those model checkers use the interpreted systems architecture, and exploration of the search space is based on ordered binary decision diagrams. Their dynamics are expressed in temporal or temporal epistemic (linear and/or branching time) logics.

A different model checker, not based on a temporal epistemic architecture, is DEMO. It has been developed by Van Eijck [20]. DEMO is short for Dynamic

Epistemic MOdelling. It allows modelling epistemic updates, graphical display of Kripke structures involved, and formula evaluation in epistemic states. This general purpose model checker has also many other facilities. DEMO is written in the functional programming language Haskell.

The model checker DEMO implements the dynamic epistemic logic of [1]. In this 'action model logic' the global state of a multi-agent system is represented by an epistemic model. But more epistemic actions are allowed than just public announcements, and each epistemic action is represented by an *action model*. Just like an epistemic model, an action model is also based on a multi-agent Kripke frame, but instead of carrying a valuation it has a precondition function that assigns a precondition to each point in the action model. A point in the action model domain stands for an atomic action.

The epistemic state change in the system is via a general operation called the *update product*: this is a way to produce a single structure (the next epistemic model) from two given structures (the current epistemic model and the current action model). We do not give details, as we restrict our attention to very simple action models, namely those corresponding to public announcements. Such action models have a singleton domain, and the precondition of that point is the announced formula. The next epistemic model is produced from the current epistemic model and the singleton action model for the announcement by the model restriction introduced in Section 2.

The recursive definition of formulas in DEMO includes (we omitted the clause for updates) `Form = Top | Prop Prop | Neg Form | Conj [Form] | Disj [Form] | K Agent Form | CK [Agent] Form`. Formula `Top` stands for $\top$, `Prop Prop` for atomic propositional letters (the first occurrence of `Prop` means that the datatype is 'propositional atom', whereas the second occurrence of `Prop` is the placeholder for an actual proposition letter, such as `P 3`), `Neg` for negation, `Conj [Form]` stands for the conjunction of a list of formulas of type `Form`, similarly for `Disj`, `K Agent` stands for the individual knowledge operator for agent `Agent`, and `CK [Agent]` for the common knowledge operator for the group of agents listed in `[Agent]`.

The pointed and singleton action model for a public announcement is created by a function `public` with a precondition (the announced formula) as argument. The update operation is specified as `upd :: EpistM -> PoAM -> EpistM` ; here `EpistM` is an epistemic state and `PoAM` is a pointed action model, and the update generates a new epistemic state. If the input epistemic state `EpistM` corresponds to some (M, s), then in case of the truthful public announcement of φ the resulting `EpistM` has the form $(M|\varphi, s)$. We can also update with a list of pointed action models: `upds :: EpistM -> [PoAM] -> EpistM` .

Complexity Each model restriction $M|\varphi$ requires determining the set $\{s \in \mathcal{D}(M) \mid M, s \models \varphi\}$. Given a model M, a state s, and a formula φ, checking whether $M, s \models \varphi$ can be solved in time $O(|M| \times |\varphi|)$, where $|M|$ is the size of the model as measured in the size of its domain plus the number of pairs in its accessibility relations, and where $|\varphi|$ is the length of the formula φ. This result has been established by the well-known labelling method [6, 2]. This method is based on dividing φ into subformulas. One then orders all these subformulas, of which there are at most $|\varphi|$, by increasing length. For each subformula, all states are labelled with either the formula or its negation, according to the valuation of the model and based on the results of previous steps. This is a bottom-up approach, in the sense that the labelling starts from the smallest subformulas. So

it ensures that each subformula is checked only once in each state.

In DEMO (v1.02) the algorithm to check whether $M, s \models \varphi$ does not employ this bottom-up approach. Instead, it uses a top-down approach, starting with the formula φ and recursively checking its largest subformulas. For example, to check whether $M, s \models K_a\psi$, the algorithm checks whether $M, s' \models \psi$ for all s' such that $s \sim_a s'$, and then recursively checks the subformulas of ψ. This algorithm is $O(|M|^{|\varphi|})$, since each subformula may need to be checked $|M|$ times, and there are at most $|\varphi|$ subformulas of φ. So, theoretically, DEMO's algorithm is quite expensive.

In practice it is less expensive, because the Haskell language and its compiler and interpreter support a cache mechanism: after evaluating a function, it caches some results in memory for reuse (see e.g. [13]). Since it is hard to predict what results will be cached and for how long, we cannot give an estimate how much the cache mechanism influences the computational results for DEMO. See also [18]. Computational results for the experiments in the next section are given in footnote 5.

6 'What Sum' in DEMO

The DEMO program SUMXYZ.hs, displayed in Figure 5, implements the 'What Sum' problem for upper bound max $=$ 10.[4] The list triples = triplesx ++ triplesy ++ triplesz (this is a union (++) of three lists) corresponds to the set of possible triples (x, y, z) for the given bound 10—note that in Haskell we are required to define such sets as lists. The next part of the program constructs the domain based on that list: this merely means that each member of the list must be associated with a state name. State names must

---

[4]The program is original but should be considered a version of the DEMO program for 'Sum and Product' in [18].

```
module SUMXYZ
where
import DEMO
upb = 10
-- constrained triples (x,y,z) with x,y,z <= upb
triplesx  = [(x,y,z)|x<-[0..upb], y<-[0..upb],
                     z<-[0..upb], x==y+z]
triplesy  = [(x,y,z)|x<-[0..upb], y<-[0..upb],
                     z<-[0..upb], y==x+z]
triplesz  = [(x,y,z)|x<-[0..upb], y<-[0..upb],
                     z<-[0..upb], z==x+y]
triples = triplesx ++ triplesy ++ triplesz
-- associating states with number triples
numtriples = llength(triples)
llength [] =0
llength (x:xs) = 1+ llength xs
itriples = zip [0..numtriples-1] triples
-- initial multi-pointed epistemic model
three :: EpistM
three =
(Pmod [0..numtriples-1] val acc [0..numtriples-1])
 where
 val = [(w,[P x,Q y,R z])|(w,(x,y,z))<-itriples]
 acc = [(a,w,v)| (w,(x1,y1,z1))<-itriples,
        (v,(x2,y2,z2))<-itriples,y1==y2,z1==z2]++
       [(b,w,v)| (w,(x1,y1,z1))<-itriples,
        (v,(x2,y2,z2))<-itriples,x1==x2,z1==z2]++
       [(c,w,v)| (w,(x1,y1,z1))<-itriples,
        (v,(x2,y2,z2))<-itriples, x1==x2, y1==y2]
-- agents a,b,c say: I do not know my number
fagxnot = Conj [(Disj[Neg (Prop (P x)),
         Neg (K a (Prop (P x))) ])| x <-[0..upb]]
aagxnot = public (fagxnot)
fagynot = Conj [(Disj[Neg (Prop (Q y)),
         Neg (K b (Prop (Q y))) ])| y <-[0..upb]]
aagynot = public (fagynot)
fagznot = Conj [(Disj[Neg (Prop (R z)),
         Neg (K c (Prop (R z))) ])| z <-[0..upb]]
aagznot = public (fagznot)
-- model restriction from announcements
result =
 showM (upds three [aagxnot, aagynot, aagznot])
```

Figure 5: The DEMO program SUMXYZ.hs

be consecutive numbers, counting from 0. The association is explicit in the list itriples that consists of pairs of which the first argument is a number (from the list [0..numtriples-1]) and the second argument is one of the triples (x, y, z) in the list triples. The initial model $\mathcal{T}_0^{10}$ is then represented as three in the program. The expression (Pmod [0..numtriples-1] val acc [0..numtriples-1]) defines three as an epistemic model (Pmod), with domain [0..numtriples-1], valuation val, a set (list) of accessibility relations acc (and [0..numtriples-1] points—left unexplained here). In val we find for example (67,[p6, q8, r2]) which says that state num-

ber 67 corresponds to triple $(6, 8, 2)$. Given `(43,[p10, q8, r2])` we now find `(a,43,67)` in `acc`.

Anne's announcement that she does not know her number is represented as the action model `aagxnot` constructed from the announcement formula `fagxnot` by the function `public`. The formula `fagxnot` is defined as `Conj [(Disj[Neg (Prop (P x)), Neg (K a (Prop (P x)))  ])|x <-[0..upb]]` . This specifies that whatever x is (`x <-[0..upb]`), if Anne has it she does not know it (`Disj[Neg (Prop (P x)), Neg (K a (Prop (P x))) ]`). The last corresponds to $\neg x_a \vee \neg K_a x_a$, which is equivalent to $x_a \rightarrow \neg K_a x_a$. Therefore, the whole expression corresponds to $\bigwedge_{0 \leq x \leq 10} x_a \rightarrow \neg K_a x_a$. This is the computationally cheaper version also formalised as $\neg \bigvee_{0 \leq x \leq 10} K_a x_a$, see Section 3.

The final line in the program asks to display the results of the three ignorance announcements. Its output is

```
==> [0,1,2,3]
[0,1,2,3]
(0,[p2,q1,r1])(1,[p1,q3,r2])
 (2,[p1,q3,r4])(3,[p2,q1,r3])
(a,[[0],[1],[2],[3]])
(b,[[0],[1],[2],[3]])
(c,[[0,3],[1,2]])
```

States are sequentially renumbered starting from 0 after each update. The four remaining triples 211, 132, 134, and 213 are clearly shown, see also Figure 4. Anne always knows her number, as her partition on the set of four states is the identity (and so does Bill, but not Cath).[5]

---

[5]We did experiments in a PC configured as Windows XP, AMD CPU 1.8Ghz, with 1G RAM. We use the Glasgow Haskell Compiler Interactive (GHCi) version 6.4.1, enabling the option ":set +s" to display information after evaluating each expression, including the elapsed time and number of bytes allocated. The results for time and space consumption of the crucial `upds msnp [aagxnot,aagynot,aagznot]` are as follows: for `upb=10`, time: 1.59 seconds, and space: 29,075,432 bytes; to give an impression of how this scales up: for `upb=20`, time: 30.31 seconds, and space: 334,474,032 bytes; for `upb=30`, time: 193.20 seconds, and space: 1,706,593,672 bytes.

We hope that this rather summary overview of DEMO nevertheless reveals its enormous versatility as a model checker. E.g., to check which states remain when a different upper bound is chosen, one merely has to replace the line `upb = 10` in the program by that other upper bound. In general, the enormous advantage of this model checker is that it allows for a separate specification of the initial model and the subsequent dynamic features, as in the original riddle (and, typically, as in the specification of the dynamics of a multiagent system to be formally modelled).

7 Conclusions

We presented an original analysis of an epistemic riddle, and formalised a finite version of the riddle with the use of public announcement logic and epistemic model checking. Crucial in the analysis was to model the riddle as an interpreted system, and to focus on the description of the background knowledge, i.e., abc-equivalence classes of the epistemic model. We introduced the model checker DEMO and the specification of the riddle in DEMO.

We think that detailed analysis of logic puzzles contributes to the understanding of logical tools and formalisms, and how to apply them to model multiagent system dynamics. In particular, the specification of security protocols in DEMO is, we think, promising. In our experiences with specifying such protocols, DEMO compares favourably to other state-of-the-art model checkers MCK and MCMAS—of course we would not dare to suggest that DEMO is 'better': when specifying a problem in which public announcements are essential, it is not surprising that a tool specially developed for such dynamics functions well.

Future development of DEMO may involve (Jan van Eijck, personal communi-

cation) facilities to model not merely information change, such as incoming new information, but also factual change. This would expand the use of this tool to model planning protocols, security protocols that include key exchange, etc. We are much looking forward to that development.

References

[1] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In I. Gilboa, editor, *Proceedings of TARK VII*, pages 43–56, 1998.

[2] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge MA, 1995.

[3] H. Freudenthal. (formulation of the sum-and-product problem). *Nieuw Archief voor Wiskunde*, 3(17):152, 1969.

[4] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In R. Alur and D. Peled, editors, *Proceedings of CAV 04*, pages 479–483. Springer, 2004.

[5] J.D. Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, University of Amsterdam, 1999. ILLC Dissertation Series DS-1999-01.

[6] J.Y. Halpern and M.Y. Vardi. Model checking vs. theorem proving: a manifesto. In V. Lifschitz, editor, *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy*, pages 151–176, San Diego, CA, USA, 1991. Academic Press Professional, Inc.

[7] J. Hintikka. *Knowledge and Belief*. Cornell University Press, Ithaca, NY, 1962.

[8] A. Liu. Problem section: Problem 182. *Math Horizons*, 11:324, 2004.

[9] C. Lutz. Complexity and succinctness of public announcement logic. In *Proceedings of AAMAS 06*, pages 137–144, 2006.

[10] J. McCarthy. Formalization of two puzzles involving knowledge. In V. Lifschitz, editor, *Formalizing Common Sense : Papers by John McCarthy*. Ablex Publishing Corporation, Norwood, N.J., 1990. original manuscript dated 1978–1981.

[11] G.E. Moore. A reply to my critics. In P.A. Schilpp, editor, *The Philosophy of G.E. Moore*, pages 535–677. Northwestern University, Evanston IL, 1942. The Library of Living Philosophers (volume 4).

[12] Y.O. Moses, D. Dolev, and J.Y. Halpern. Cheating husbands and other stories: a case study in knowledge, action, and communication. *Distributed Computing*, 1(3):167–176, 1986.

[13] N. Nethercote and A. Mycroft. The cache behaviour of large lazy functional programs on stock hardware. *SIGPLAN Notices*, 38(2 supplement):44–55, 2003.

[14] G. Panti. Solution of a number theoretic problem involving knowledge. *International Journal of Foundations of Computer Science*, 2(4):419–424, 1991.

[15] J.A. Plaza. Logics of public communications. In M.L. Emrich *et al.*, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216. Oak Ridge National Laboratory, 1989.

[16] F. Raimondi and A.R. Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In *Proceedings of AAMAS 04*, pages 630–637. IEEE Computer Society, 2004.

[17] K. Su. Model checking temporal logics of knowledge in distributed systems. In D. L. McGuinness and G. Ferguson, editors, *Proceedings of AAAI 04*, pages 98–103. AAAI Press / The MIT Press, 2004.

[18] H.P. van Ditmarsch, J. Ruan, and R. Verbrugge. Sum and product in dynamic epistemic logic. *Journal of Logic and Computation*, 2007. To appear.

[19] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2007.

[20] J. van Eijck. Dynamic epistemic modelling. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, 2004. CWI Report SEN-E0424.

[21] W. Wu. 100 prisoners and a lightbulb. `www.ocf.berkeley.edu/~wwu/papers/100prisonersLightBulb.pdf`, 2001.