Department of Computer Science, University of Otago

Technical Report OUCS-2006-01

Sum and Product in Dynamic Epistemic Logic

Authors:

H. P. van Ditmarsch
Computer Science, University of Otago

J. Ruan
Computer Science, University of Liverpool, United Kingdom

L.C. Verbrugge
Artificial Intelligence, University of Groningen, Netherlands

Status: Journal submission

Department of Computer Science,
University of Otago, PO Box 56, Dunedin, Otago, New Zealand

http://www.cs.otago.ac.nz/research/techreports.html

Sum and Product in Dynamic Epistemic Logic*

H.P. van Ditmarsch†, J. Ruan‡, and L.C. Verbrugge§

Abstract

The Sum-and-Product riddle was first published in [Fre69]. We provide an overview on the history of the dissemination of this riddle through the academic and puzzle-math community. This includes some references to precursors of the riddle, that were previously (as far as we know) unknown.

We then model the Sum-and-Product riddle in a modal logic called public announcement logic. This logic contains operators for knowledge, but also operators for the informational consequences of public announcements. The logic is interpreted on multi-agent Kripke models. The information in the riddle can be represented in the traditional way by number pairs, so that Sum knows their sum and Product their product, but also as an interpreted system, so that Sum and Product at least know their local state. We show that the different representations are isomorphic. We also provide characteristic formulas of the initial epistemic state of the riddle. Finally we analyze one of the announcements towards the solution of the riddle as a so-called unsuccessful update: a formula that become false because it is announced.

The riddle is then implemented and its solution verified in the epistemic model checker DEMO. This can be done, we think, surprisingly elegantly. The results are compared with other work in epistemic model checking.

Keywords: modal logic, puzzle math, dynamic epistemics, characteristic formula, model checking

1 Introduction

The following problem, or riddle, was first stated, in the Dutch language, in [Fre69] and subsequently solved in [Fre70]. A translation of the original formulation is:

*Contact author is Hans van Ditmarsch. Hans and Ji appreciate support from AOARD research grant AOARD-05-4017. Hans and Rineke appreciate support from the Netherlands Organization for Scientific Research (NWO). A shortened version of Sections 7 and 8, including Figure 2, have previously appeared in conference proceedings [vDRV05].

†Computer Science, University of Otago, New Zealand, hans@cs.otago.ac.nz

‡Computer Science, University of Liverpool, United Kingdom, jruan@csc.liv.ac.uk

§Artificial Intelligence, University of Groningen, the Netherlands, rineke@ai.rug.nl

No. 223. A zegt tot S en P: Ik heb twee gehele getallen x, y gekozen met $1 < x < y$ en $x + y \leqslant 100$. Straks deel ik $s = x + y$ aan S alleen mee, en $p = xy$ aan P alleen. Deze mededelingen blijven geheim. Maar jullie moeten je inspannen om het paar (x, y) uit te rekenen.

Hij doet zoals aangekondigd. Nu volgt dit gesprek:

1. P zegt: Ik weet het niet.
2. S zegt: Dat wist ik al.
3. P zegt: Nu weet ik het.
4. S zegt: Nu weet ik het ook.

Bepaal het paar (x, y).

(*H. Freudenthal*).

Figure 1: The original publication

> 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).

This problem is considered a riddle, or puzzle, because the agents' announcements *appear* to be uninformative, as they are about ignorance and knowledge and not about (numerical) facts, whereas *actually* they are very informative: the agents learn facts from the other's announcements. For example, the numbers cannot be 2 and 3, or any other pair of prime numbers, nor for example 2 and 4, because in all those cases Product would immediately have deduced the pair from their product. As a somewhat more complicated example, the numbers cannot be 14 and 16: if they were, their sum would be 30. This is also the sum of the prime numbers 7 and 23. But then, as in the previous example, Product would (P) would have known the numbers, and therefore Sum (S) – if the sum had been 30 – would have considered it possible that Product knew the numbers. But Sum said that he *knew* that Product didn't know the numbers. So the numbers cannot be 14 and 16. Sum and Product learn enough, by eliminations

of which we gave some examples, to be able to determine the pair of numbers: the unique solution of the problem is the pair $(4, 13)$.

The knowledge that agents have about other agents' mental states and, in particular, about the effect of communications, is vital for solving important problems in multi-agent systems, both for cooperative and for competitive groups. Dynamic epistemic logic was developed to study the changes brought about by communication in such higher-order knowledge of other agent's and of group knowledge [BMS98, Ger99]. The Sum-and-Product puzzle presents a complex illustrative case of the strength of specifications in dynamic epistemic logic and of the possibilities of automated model checking, and both can also be used in real multi-agent system applications. As far as we know, we are the first to use an automated model checker to tackle the Sum-and-Product problem.

Section 2 gives an overview of the dissemination of the riddle through the academic community, and suggests some precursors. In Section 3 we introduce public announcement logic. In Section 4 we model the Sum-and-Product problem in public announcement logic. Section 5 models the Sum-and-Product problem, alternatively, as an interpreted system, and Section 6 provides the general setting of unsuccessful updates of which some announcements in the riddle provide examples. In Section 7 we introduce the epistemic model checker DEMO. In Section 8 we implement the Sum-and-Product specification of Section 4 in DEMO, and we verify its epistemic features. Section 9 reports on DEMO implementations of similar problems, and compares the model checking results to our experiences with other epistemic model checkers.

2 History

John McCarthy wrote the earliest full-length treatment of the Sum-and-Product riddle in the years 1978–1981 [McC90]. McCarthy formulates the problem as follows:

> Two numbers m and n are chosen such that $2 \leq m \leq n \leq 99$. Mr. S is told their sum and Mr. P is told their product. The following dialogue ensues:
>
> i. Mr. P: I don't know the numbers.
>
> ii. Mr. S: I knew you didn't know. I don't know either.
>
> iii. Mr. P: Now I know the numbers.
>
> iv. Mr. S: Now I know them too.
>
> In view of the above dialogue, what are the numbers?

In [McC90] the problem is elegantly modeled in modal logic in such a way that it can be processed in the (first-order) logic theorem prover FOL. This includes an – almost off-hand – introduction of what corresponds to the essential concept

of 'common knowledge': what Sum and Product commonly know is crucial to a clear understanding of the problem. Common knowledge had received a very interesting treatment already in Lewis' 1969 book *Convention* [Lew69] (and also appears in other philosophical literature from that period, e.g. in Schiffer's work [Sch72]), but McCarthy seems to have re-invented it in his 1981 article, thereby inspiring research on common knowledge in Artificial Intelligence.

Note that in the second announcement, Sum seems to give some additional information that does not appear in the Freudenthal-version of the dialogue, namely "I don't know either". However, some simple considerations show that this addition is superfluous, because at the current point in the dialogue, it is already common knowledge among the participants that Sum doesn't know either. After all, the only situation in which Sum *does* know the two numbers from the start is the one where the pair of numbers is $(2, 3)$, which has already been ruled out by Product's first announcement. Further, note that McCarthy allows the two numbers to be the same, unlike Freudenthal. This also does not affect the solution.

Many different versions of the puzzle elicited much discussion from the late seventies onwards. The variations are caused by different announcements, different ranges for the numbers, and different choices for what is considered to be common knowledge at the starting-point. For yet another example, for a certain larger range of possible numbers than 2—99 one finds a solution different from $(4, 13)$ but that then after all is in the 2—99 range. Discussions of several variants of the problem can be found in the literature on recreational mathematics, see especially [Gar79, Sal95, Isa95], and a website `www.mathematik.uni-bielefeld.de/~sillke/PUZZLES/logic_sum_product` that contains many other references.

More geared towards an epistemic logical audience are [Pla89, Pan91, vdM94, vdHV02]. Plaza and Panti were students of Rohit Parikh and have both made some interesting contributions to epistemic logic. In [Pla89] the Sum-and-Product problem is modeled in a dynamic epistemic logic that is the precursor of the public announcement logic presented here, namely without an operator for common knowledge. In [Pan91], on the other hand, the common knowledge involved in the Sum-and-Product puzzle is investigated in detail, with an emphasis on the arithmetic involved. For example, for the formulation of the problem where the range of numbers (up to 100) is not considered to be common knowledge at the start, Panti proves that if the sum of the numbers is greater or equal than 7, then this (and its logical consequences) is the *only* fact that is common knowledge among Sum and Product. Finally, Van der Meyden [vdM94] suggests a solution in temporal epistemic logic.

2.1 Looking for the origin of Sum and Product

In both of the two first full-length publications on the Sum and Product riddle [McC90, Gar79], the authors explicitly wondered about but could not give its exact origins. John McCarthy explains in a footnote in his paper [McC90]:

> I have not been able to trace Mr. S and Mr. P back beyond its alleged appearance on a bulletin board at Xerox PARC.

Martin Gardner, in his 1979 "Mathematical Games" column [Gar79], writes:

> This beautiful problem, which I call "impossible" because it seems to lack sufficient information for a solution, began making the rounds of mathematics meetings a year or so ago. I do not know its origin.

After the appearance of [Gar79], the fact that the puzzle had been published already in 1969 by Dutch topologist and specialist on mathematics education Hans Freudenthal, was brought to Gardner's attention by Dutch algebraist Robert van der Waall. Van der Waall was one of the small number of Dutch mathematicians who had sent in a correct solution to the Dutch mathematics journal *Nieuw Archief voor Wiskunde* after the puzzle's first appearance in 1969.

We have tried to fill in two missing pieces in the history of the Sum-and-Product riddle:

i. If [Fre69] is indeed the first published appearance of the problem, then how did the problem migrate from the Dutch mathematics community of the late 1960s and early 1970s to "a bulletin board at Xerox Parc" and "the rounds of mathematics meetings" in the United States in the late 1970s?

ii. Did Freudenthal invent the problem? And if so, has he possibly been inspired by (less complex) precursors?

Despite several requests on international e-mail lists, we have not been able to answer the first question. As to the second question, we received a partial answer from one of the subscribers to *Nieuw Archief voor Wiskunde*, who thought he remembered to have seen the Sum-and-Product riddle in the puzzle column "Breinbrouwsels" (brain brews) in the now defunct Dutch-language weekly *De Katholieke Illustratie* ('Illustrated Catholic Magazine') in the 1950s.

We have visited several libraries and thus managed to read almost all of the 626 "Breinbrouwsels" that G. van Tilburg published from 1954 until 1965 (and of those we did not read, we could infer what they were from their answers, in other issues). This did not turn up the Sum-and-Product puzzle, but we did find four puzzles (published in 1954, 1955, 1957, and 1963, respectively) that can clearly be seen as precursors. Mostly these puzzles involve partial information about a number of persons' ages, where the fact that one of the participants cannot deduce the ages from the interlocutor's hint, but can deduce them after some further dialogue, is crucial information helping the reader to solve the problem. We will describe some of Van Tilburg's interesting problems in a forthcoming publication in Dutch.

Thus, as far as we know now, Freudenthal really invented the Sum-and-Product puzzle, but may have been inspired by Van Tilburg's "Breinbrouwsels". Possibly he also read some even earlier riddles of British origin, to which we turn our attention now.

2.2 Precursors of Sum and Product

David Singmaster's bibliographies on recreational mathematics (see e.g. `www.g4g4.com/MyCD5/SOURCES/singmaterial.htm`) point to some candidate epistemic puzzles that appeared even earlier than Van Tilburg's. The earliest precursor of the Sum-and-Product riddle that we have been able to trace is the following one, probably invented by Williams and Savage and first published in book-form in 1940 in *The Penguin Problems Book* [WS40, p.53]:

> **The church afloat**
> "I'm taking three females on the river to-morrow," said the vicar to his curate; "would you care to join our party?"
>
> "What are their ages?" asked the curate, cautiously. "Far be it from me to disclose a lady's age!" said the vicar, "but I can tell you this – the product of their ages is 840, and the sum is twice the number of years in your own age. You, a mathematician, should be able to find their ages for yourself."
>
> "Sounds like casuistry, Vicar," said the curate; "but, as a matter of fact, I can't find their ages from your data. By the way, is the eldest older than you?"
>
> "No, younger." "Ah, now I know their ages!" said the curate. "Thanks, I will come with pleasure."
>
> What was the curate's age? How old were the ladies? And what can be deduced about the vicar's age?

Here follows Williams' and Savage's answer [WS40, p.135]:

> Sum of ages must be even.
>
> Uncertainty, resolved by the vicar's final statement, must be due to the fact of there being more than one such sum which was twice the curate's age.
>
> Of the possible sets of 3 factors of 840, there are only two cases of the same *even* sum occurring more than once. The sums in these cases are 46 and 30. Now the curate's age could not be 15; therefore he was 23.
>
> The sets of female ages giving a sum of 46 are 35, 8, 3 and 30, 14, 2. Since the vicar's answer excluded one of these, that one must be the former. Therefore the ladies' ages were 30, 14, 2, and the vicar's age must lie between 30 and 35.

Note that some world knowledge is used implicitly here, namely the fact that mathematicians (and curates) are always older than 15 years, and the fact that the curate, being a mathematician, reasons correctly.

Another problem, that was published in 1944 in *The Second Penguin Problems Book* [WS44, p.27], also hinges on the fact that only for some number

combinations there is more than one way to make the same sum. In a way, the next problem is less attractive than the previous one, because the uncertainty is not completely dissolved at the end: readers are asked to derive the sum of the ages only.

> **Domiciliary**
> "I have told you my age," said Mr. Ptolemy to the inspector who had just knocked on his door. "Besides myself, there are three persons living in this house; the product of their ages is one thousand two hundred ninety-six, and the sum of their ages is the number of the house."
>
> "But it is impossible for me to be *sure* of their ages without further information," said the inspector. "Is any one of them the same age as yourself?"
>
> "No," said Mr. Ptolemy.
>
> "Thanks; now I know their ages," said the inspector.
>
> What was the number of Mr. Ptolemy's house?

This time, the explanation is as follows [WS44, p.116]; again, the authors implicitly use some world knowledge:

> There are many ways of splitting 1296 into three factors, but only *possible* ones need be considered. Two of these sets of factors have the same sum, namely 1, 18, 72 and 2, 8, 81, adding up to 91. The other sums are all different.
>
> As the inspector could not be sure of the ages from the fact that they added up to the number of the house (which he, of course, knew), this number must have been 91.
>
> [Mr. Ptolemy's age - also known to the inspector - must have been 72 or 81 (unless it was 18 or 8 - both unlikely), but we have no means of deciding this point.]

The above two puzzles are roughly of the same kind as Van Tilburg's, but still different. In fact, Van Tilburg may have been inspired to create his puzzles after reading the British gentlemen. Essentially the same problem as "Domiciliary", but in a somewhat different guise, was printed in Greenblatt's *Mathematical Entertainments* [Gre68], first published in the United States in 1965. Greenblatt starts with some historical speculation:

> One of the few amusing things to come out of World War II was a new type of brain twister - the "census-taker" problem. (The time and place of origin of a problem are difficult to specify. To the best of the author's knowledge, this problem was born on the M.I.T. campus in one of the war projects.)

As we now know, the type of problem probably stems from at least somewhat before the start of World War II, and from Great Britain instead of the United States. After all, *The Penguin Problems Book*, although published during the War in 1940, was mostly based on earlier puzzles from Williams' and Savage's column "Perplexities" that used to appear in *The Strand Magazine*. For more details, see `www.cs.otago.ac.nz/staffpriv/hans/sumpro/`.

2.3 Descendants of Sum and Product

In recent years, variants of the Sum-and-Product riddle keep cropping up. Johan van Benthem has communicated a particularly nice example, dubbed "GSM-puzzle". The conversation between the two participants in the GSM-puzzle follows exactly the same pattern as the one in the Sum-and-Product riddle. However, due to the context in terms of playing cards with points and colors, no arithmetic is needed to solve it. Thus, the epistemic complexity remains, while the arithmetic complexity has been canceled. For a formulation of the problem, see `www.ai.rug.nl/mas/openprojecten.html\#GSM` and/or `www.cs.otago.ac.nz/staffpriv/hans/sumpro/`.

Some of the more recent variants include more than two participants in the clarifying conversation, for example the following one [Liu04], which combines themes from the Muddy-Children puzzle [MDH86] with those from the Sum-and-Product puzzle. We leave this problem as a challenge to the reader.

> Each of Ace, Bea and Cec is wearing a hat on which a positive integer is printed. Each can see only the numbers on the others' hats. They are told that one of the numbers is the sum of the other two. They make the following statements in succession.
>
> i. Ace: I cannot deduce what my number is.
>
> ii. Bea: Knowing that, I still cannot deduce what my number is.
>
> iii. Cec: Knowing that, I still cannot deduce what my number is.
>
> iv. Ace: Now I can deduce that my number is 50.
>
> Assuming that they all use sound reasoning, what are the numbers on the two other hats?

After this detailed overview of the dissemination of the Sum-and-Product riddle, which we hope may prevent some of this information from gradually disappearing into the fog of war on academic battlegrounds, we continue with the more technical core of this paper, that consists of an introduction into public announcement logic, modelling the riddle in this logic, and verifying its properties in a model checker.

3 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 [FHMV95, vdHV02, vDvdHK05]. 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 Q. The language of public announcement logic is inductively defined as

$$\varphi ::= q \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_n\varphi \mid C_G\varphi \mid [\varphi]\psi$$

where $q \in Q$, $n \in N$, and $G \subseteq N$ are arbitrary. For $K_n\varphi$, read 'agent n knows formula φ'. For $C_G\varphi$, read 'group of agents G commonly know formula φ'. For $[\varphi]\psi$, read 'after public announcement of φ, formula ψ (is true)'.

Next, we introduce the structures. An *epistemic model* $M = \langle W, \sim, V\rangle$ consists of a *domain* W of (factual) *states* (or 'worlds'), *accessibility* $\sim : N \to \mathcal{P}(W \times W)$, where each $\sim(n)$ is an equivalence relation, and a *valuation* $V : Q \to \mathcal{P}(W)$. For $w \in W$, (M, w) is an *epistemic state* (also known as a pointed Kripke model). For $\sim(n)$ we write $\sim_n$, and for $V(q)$ we write V_q. So, accessibility $\sim$ can be seen as a set of equivalence relations $\sim_n$, and V as a set of valuations V_q. Given two states w, w' in the domain, $w \sim_n w'$ means that w is indistinguishable from w' for agent n on the basis of its information. For example, at the beginning of the riddle, pairs $(14, 16)$ and $(7, 23)$ are indistinguishable for Sum but not for Product. Therefore, assuming a domain of number pairs, we have that $(14, 16) \sim_S (7, 23)$ but that $(14, 16) \not\sim_P (7, 23)$. The group accessibility relation $\sim_G$ is the transitive and reflexive closure of the union of all accessibility relations for the individuals in G: $\sim_G \equiv (\bigcup_{n\in G} \sim_n)^*$. This relation is used to interpret common knowledge for group G.

Finally, we give the semantics. Assume an epistemic model $M = \langle W, \sim, V\rangle$.

$$\begin{array}{lll}
M, w \models q & \text{iff} & w \in V_q \\
M, w \models \neg\varphi & \text{iff} & M, w \not\models \varphi \\
M, w \models \varphi \wedge \psi & \text{iff} & M, w \models \varphi \text{ and } M, w \models \psi \\
M, w \models K_n\varphi & \text{iff} & \text{for all } v \in W : w \sim_n v \text{ implies } M, v \models \varphi \\
M, w \models C_G\varphi & \text{iff} & \text{for all } v \in W : w \sim_G v \text{ implies } M, v \models \varphi \\
M, w \models [\varphi]\psi & \text{iff} & M, w \models \varphi \text{ implies } M|\varphi, w \models \psi
\end{array}$$

Here, epistemic model $M|\varphi = \langle W', \sim', V'\rangle$ is defined as

$$\begin{array}{lcl}
W' & = & \{w' \in W \mid M, w' \models \varphi\} \\
\sim'_n & = & \sim_n \cap (W' \times W') \\
V'_q & = & V_q \cap W'
\end{array}$$

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 $\langle\varphi\rangle$: $M, w \models \langle\varphi\rangle\psi$ iff $M, w \models \varphi$ and $M|\varphi, w \models \psi$.

Formula φ is valid on model M, notation $M \models \varphi$, if and only if for all states w in the domain of M: $M, w \models \varphi$. Formula φ is valid, notation $\models \varphi$, if and only if for all models M: $M \models \varphi$. Logical consequence $\Psi \models \varphi$ is defined as "for all (M, w), if $M, w \models \psi$ for all $\psi \in \Psi$, then $M, w \models \varphi$." For $\{\psi\} \models \varphi$, write $\psi \models \varphi$.

A proof system for this logic is presented, and shown to be complete, in [BMS98], with precursors – namely for public announcement logic *without* common knowledge – in [Pla89, Ger99]. For a concise completeness proof, see [vDvdHK05]. Some relevant principles of this logic are

i. $[\varphi]\psi \leftrightarrow (\varphi \rightarrow [\varphi]\psi)$

ii. $[\varphi][\psi]\chi \leftrightarrow [\varphi \wedge [\varphi]\psi]\chi$

iii. $[\varphi]K_n\psi \leftrightarrow (\varphi \rightarrow K_n[\varphi]\psi)$

iv. $[C_N\varphi]C_N\varphi$

Item i expresses that the interpretation of the dynamic operator $[\varphi]$ is a *partial* function. Item ii expresses that a sequence of two announcements φ and ψ can be replaced by the single announcement 'φ, and after φ, ψ'. Item iii expresses the preconditions and postconditions of announcements with respect to individual knowledge (for common knowledge, this relation is more complex). Item iv expresses that *public* knowledge (i.e., common knowledge for the entire group of agents) remains true after announcement. Not all formulas remain true after their announcement, in other words, $[\varphi]\varphi$ is *not* a principle of this logic. This matter will be addressed in Section 6. Some announcements towards the solution of the Sum-and-Product problem provide concrete counterexamples, and this will explain why the 'puzzling' conversation of S and P makes sense.

4 Sum and Product in Public Announcement Logic

We give a specification of the Sum-and-Product problem in public announcement logic. First we need to determine the set of atomic propositions and the set of agents. In the formulation of the problem, x, y are two integers such that $1 < x < y$ and $x+y \leq 100$. Define $I \equiv \{(x, y) \in \mathbb{N}^2 \mid 1 < x < y \text{ and } x+y \leq 100\}$. Consider the variable x. If its value is 3, we can represent this information as the (truth of) the atomic proposition '$x = 3$'. Slightly more formally we can think of '$x = 3$' as a propositional letter x_3. Thus we create a (finite) set of atoms $\{x_i \mid (i, j) \in I\} \cup \{y_j \mid (i, j) \in I\}$.

Concerning the agents, the role of the announcer A is to guarantee that the background knowledge for solving the problem is commonly known among Sum and Product. The announcer need not be introduced as an agent in the logical

modelling of the system. That leaves $\{S, P\}$ as the set of agents. Agents S and P will also be referred to as Sum and Product, respectively.

The proposition 'Sum knows that the numbers are 4 and 13' is represented as $K_S(x_4 \wedge y_{13})$. The proposition 'Sum knows the (pair of) numbers' is described as $K_S(x,y) \equiv \bigvee_{(i,j)\in I} K_S(x_i \wedge y_j)$. Similarly, 'Product knows the numbers' is represented by $K_P(x,y) \equiv \bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$. Furthermore, note that the 'knew' in announcement ii, by Sum, refers to the truth of $K_S\neg K_P(x,y)$ in the *initial* epistemic state, not in the epistemic state *resulting* from announcement i, by Product. Therefore, announcement i by Product is superfluous in the subsequent analysis.[1] This is sufficient to formalize the announcements made towards a solution of the problem:

i. P says: "I do not know it": $\neg K_P(x,y)$

ii. S says: "I knew you didn't": $K_S\neg K_P(x,y)$

iii. P says: "I now know it": $K_P(x,y)$

iv. S says: "I now also know it": $K_S(x,y)$

We can interpret these statements on an epistemic model $\mathcal{SP}_{(x,y)} \equiv \langle I, \sim, V\rangle$ consisting of a domain of all pairs $(x,y) \in I$ (as above), with accessibility relations $\sim_S$ and $\sim_P$ such that for Sum: $(x,y) \sim_S (x',y')$ iff $x+y = x'+y'$, and for Product: $(x,y) \sim_P (x',y')$ iff $xy = x'y'$; and with valuation V such that $V_{x_i} = \{(x,y) \in I \mid x = i\}$ and $V_{y_j} = \{(x,y) \in I \mid y = j\}$.

We can describe the solution of the problem as the truth of the statement

$$\mathcal{SP}_{(x,y)}, (4,13) \models \langle K_S\neg K_P(x,y)\rangle\langle K_P(x,y)\rangle\langle K_S(x,y)\rangle\top$$

This expresses that, if $(4,13)$ is the initial state, then it is possible to publicly announce *ii*, *iii*, and *iv*, in that order. We can also express more properly that $(4,13)$ is the only solution as the model validity

$$\mathcal{SP}_{(x,y)} \models [K_S\neg K_P(x,y)][K_P(x,y)][K_S(x,y)](x_4 \wedge y_{13})$$

5 Sum and Product as an interpreted system

A relevant observation is that a pair of numbers (x,y) with $x < y$ corresponds to exactly one sum-product pair (s,p). In one direction this is trivial, for the other direction: assume $(x+y, xy) = (x'+y', x'y')$. Let without loss of generality x be the smaller of x and x', so that $x' = x+v$. Then from $xy = x'y' = (x+v)(y-v)$ follows that $yv - xv - v^2 = 0$, so that $v = 0$ or $v = y - x$. The second merely reverses the role of x and y; in our terms, it cannot be satisfied, because x was required to be strictly smaller than y. This observation paves the way

[1] In dynamic epistemic logic *with assignment* one can model such past tense epistemic statements explicitly [Koo05].

for a different modelling of the problem than the traditional one with '(smaller number, larger number)' pairs (x, y).

We now let atomic propositions represent the sum and product of the different numbers, instead of representing these numbers themselves. For example, s_7 represents that the sum of the two numbers is 7. We allow a slight abuse of the language: if $i+j = k$ then we also write s_{i+j} for s_k. Similarly, we write p_{ij} for p_l when $ij = l$. Thus we create a set of atoms $\{s_{x+y} \mid (x, y) \in I\} \cup \{p_{xy} \mid (x, y) \in I\}$.

The obvious way to interpret *such* atoms is on an epistemic model $\mathcal{SP}_{(s,p)} \equiv \langle W', \sim', V' \rangle$ with a domain W' consisting of all pairs (s, p) such that $s = x + y$ and $p = xy$ (as in the original formulation of the problem) for all $(x, y) \in I$, i.e., with $1 < x < y$ and $x + y \leq 100$; with *accessibility relations* $\sim'_S$ and $\sim'_P$ such that for Sum: $(s, p) \sim'_S (s', p')$ iff $s = s'$, and for Product: $(s, p) \sim'_P (s', p')$ iff $p = p'$; and with valuation such that $V'_{s_{x+y}} = \{(s, p) \in W' \mid s = x + y\}$ and $V'_{p_{xy}} = \{(s, p) \in W' \mid p = xy\}$.

We have now modelled the problem as an *interpreted system* where agents at least know their local state. Interpreted systems were introduced in theoretical computer science as an abstract architecture for distributed systems [FHMV95]. Sum's local state is the sum of the two numbers, Product's local state is the product of the two numbers. A global state for the problem is a pair of local states, one for Sum and one for Product. The set of global states is a subset of the full cartesian product of local state values: the dependencies between local states enable Sum and Product to communicate their local state to each other without explicitly referring to it.

'Sum knows the (pair of) numbers' can be represented by 'Sum knows the global state of the system', i.e., as $K_S(s, p) \equiv \bigvee_{(x,y)\in I} K_S(s_{x+y} \wedge p_{xy})$, and, similarly, 'Product knows the numbers' by $K_P(s, p) \equiv \bigvee_{(x,y)\in I} K_P(s_{x+y} \wedge p_{xy})$. The formalization of the announcements made towards a solution of the problem is then similar to above:

$$\mathcal{SP}_{(s,p)} \models [K_S \neg K_P(s, p)][K_P(s, p)][K_S(s, p)](s_{4+13} \wedge p_{4\cdot 13})$$

An advantage of this representation is that we can apply known results for interpreted systems, such that agents at least know their local state, and the availability of *characteristic formulas* for modal structures [BM96, vB98] to the specific case of finite interpreted systems [vDvdHK03]. That agent S knows its local state, means that S knows the sum of the two numbers, whatever they are: $\mathcal{SP}_{(s,p)} \models s_{x+y} \rightarrow K_S s_{x+y}$. From this follows that in the models for our problem a requirement $K_S(s_{x+y} \wedge p_{xy})$, that is equivalent to $K_S s_{x+y} \wedge K_S p_{xy}$, is equivalent to $K_S p_{xy}$. Similarly, $p_{xy} \rightarrow K_P p_{xy}$, and therefore, in the models, $K_P(s_{x+y} \wedge p_{xy})$ is equivalent to $K_P s_{x+y}$.

Concerning the characteristic formula describing the initial situation, we can apply results from [vDvdHK03].[2] The characteristic formula $\delta(\mathcal{SP}_{(s,p)})$ is

[2] A characteristic formula of a pointed model (M, w) is a formula $\delta(M, w)$ such that $M, w \models \psi$ iff $\delta(M, w) \models \psi$, in other words, any ψ true in (M, w) is entailed by $\delta(M, w)$. A similar notion equates model validity with entailment by way of $M \models \psi$ iff $\delta(M) \models \psi$. These descriptions exist for finite epistemic models. We also have that $\delta(M, w) \leftrightarrow (\delta(w) \wedge C_N \delta(M))$,

defined as

$$\begin{array}{rcl}\delta(\mathcal{SP}_{(s,p)}) & \equiv & \bigvee_{(x,y)\in I}(s_{x+y} \wedge p_{xy}) \ \wedge \\ & & \bigwedge_{(x,y)\in I}(K_S s_{x+y} \leftrightarrow \neg K_S \neg(s_{x+y} \wedge p_{xy})) \ \wedge \\ & & \bigwedge_{(x,y)\in I}(K_P p_{xy} \leftrightarrow \neg K_P \neg(s_{x+y} \wedge p_{xy}))\end{array}$$

The first conjunct of $\delta(\mathcal{SP}_{(s,p)})$ sums up the valuations of the different states in the domain. The second conjunct says (entails) that S knows its local state if and only if it considers possible any global state with that local state. For example $K_S s_{17} \leftrightarrow \neg K_S \neg(s_{17} \wedge p_{52})$; another conjunct is $K_S s_{17} \leftrightarrow \neg K_S \neg(s_{17} \wedge p_{60})$. From this follows that $K_S s_{17}$ implies $\neg K_S \neg p_{52} \wedge \neg K_S \neg p_{60} \wedge \ldots$: if the sum of the two numbers is 17, S considers it possible that their product is 52, or 60, etc.

The traditional modelling of Sum and Product relates to the interpreted system modelling in a precise technical sense. Expand the language to one containing atoms for all numbers x, y *and* atoms for all sums and products s, p of those numbers. Extend the models $\mathcal{SP}_{(x,y)}$ and $\mathcal{SP}_{(s,p)}$ to $\mathcal{SP}^+_{(x,y)}$ and $\mathcal{SP}^+_{(s,p)}$, respectively, by adding valuations for all sum and product atoms in the former, and for all smaller and larger number atoms in the latter. For example, to define $\mathcal{SP}^+_{(x,y)}$ we have to add valuations for all atoms s and p such that $(x, y) \in V^+_{s_{x+y}}$ iff $s = x + y$ and $(x, y) \in V^+_{p_{xy}}$ iff $p = xy$. We now have that $\mathcal{SP}^+_{(x,y)}$ and $\mathcal{SP}^+_{(s,p)}$ are isomorphic. (From this then follows that the models are also *bisimilar* [BdRV01] – a slightly weaker notion of 'sameness of models' that still guarantees that the theories describing the models are logically equivalent.) Without going into great detail, it suffices to define the isomorhphism as $\mathfrak{R} : I \to W'$ such that $\mathfrak{R} : (x, y) \mapsto (x + y, xy)$, to observe that this relation is a bijection, that $(x, y) \sim_S (x', y')$ iff $\mathfrak{R}(x, y) \sim_S \mathfrak{R}(x', y')$ iff $(x + y, xy) \sim_S (x' + y', x'y')$, and similarly for Product, and that the valuation of all facts remains the same for any states (x, y) and $(x + y, xy)$. The characteristic formula for the interpreted system $\mathcal{SP}^+_{(s,p)}$ in the expanded logical language is the previous one, $\delta(\mathcal{SP}_{(s,p)})$, in conjunction with

$$\bigwedge_{(i,j)\in I} ((x_i \wedge y_j) \leftrightarrow (s_{i+j} \wedge p_{ij}))$$

This *propositional* equivalence relates a number pair to its unique corresponding sum and product pair.

To conclude, using the interpreted system representation, we can describe the initial situation for the Sum-and-Product puzzle in a very precise way. Moreover, the traditional representation and the interpreted system one are in a sense interchangeable: they have the same logical theory.

---

where $\delta(w)$ is *the description of state* w, for example summing up its valuation, or some other formula only true in w.

6 Unsuccessful updates

Not all formulas remain true after their announcement, in other words, $[\varphi]\varphi$ is *not* a principle of public announcement logic. A poignant example is when I'm telling you that "You don't know that the Highlanders just beat the Lions!". In the standard conversational setting this presumes that the factual information of which you are ignorant is actually the case, i.e., this normally means "The Highlanders just beat the Lions and you don't know that the Highlanders just beat the Lions." It is therefore an announcement of the form $q \wedge \neg K_n q$. After the announcement, you know that the fact in question is true – $K_n q$ – and therefore the formula of the announcement has become false: $K_n q$ entails $\neg q \vee K_n q$, which is equivalent to $\neg(q \wedge \neg K_n q)$, the negation of the announcement. In a somewhat different setting that the formula $q \wedge \neg K_n q$ cannot be consistently known, this phenomenon has been known in philosophical circles for a long time, namely as the Moore-paradox [Moo42, Hin62]. In the underlying dynamic setting it has been described as an unsuccessful update in [Ger99, Ger05]. General terminology is proposed in [vDK05]. Let φ be a formula in the language of public announcement logic:

- *Successful formula*
  φ is successful iff $[\varphi]\varphi$ is valid.
- *Unsuccessful formula*
  φ is unsuccessful iff it is not successful.
- *Successful update*
  φ is successful in epistemic state (M, w) iff $M, w \models \langle\varphi\rangle\varphi$
- *Unsuccessful update*
  φ is unsuccessful in (M, w) iff $M, w \models \langle\varphi\rangle\neg\varphi$.

Note that an unsuccessful formula may be a successful update in one epistemic state and an unsuccessful update in another epistemic state. It can be shown that $[\varphi]\varphi$ is valid iff $[\varphi]C_G\varphi$ is valid iff $\varphi \rightarrow [\varphi]C_G\varphi$ is valid. (See [vDK05], the second equivalence follows directly from the principle $[\varphi]\psi \leftrightarrow (\varphi \rightarrow [\varphi]\psi)$, listed as item i on page 10 in Section 3.) Therefore, the successful formulas capture the notion 'formulas that remain true after their announcement'.

Clearly, also in the course of solving the Sum-and-Product problem the agents appear to learn things that they did not know before. So some reversal of ignorance into knowledge seems to take place. We therefore expect that some of the announcements made towards the solution of the problem are unsuccessful updates. In this section we refer to those four successive announcements as (how they have been enumerated before, namely as) (i) $\neg K_P(x, y)$, (ii) $K_S\neg K_P(x, y)$, (iii) $K_P(x, y)$, and (iv) $K_S(x, y)$.

The case i Remember that announcement i was superfluous in the analysis of the riddle. We therefore do not expect it to have 'surprising informational qualities', and this is indeed the case: i is a successful formula.

Formula i equals $\neg K_P(x,y)$ where $K_P(x,y)$ which was defined as $\bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$. Therefore, it has form $\neg K_n\varphi \wedge \neg K_n\psi \wedge \ldots$, with $\varphi, \psi, \ldots$ booleans. We show that this formula is successful for two conjuncts, i.e., formula $[\neg K_n\varphi \wedge \neg K_n\psi](\neg K_n\varphi \wedge \neg K_n\psi)$ is valid for booleans φ and ψ; the case for the longer finite conjunction follows similarly.

Let M, w be arbitrary. Assume $M, w \models \neg K_n\varphi \wedge \neg K_n\psi$. We have to prove that $M|(\neg K_n\varphi \wedge \neg K_n\psi), w \models \neg K_n\varphi \wedge \neg K_n\psi$. From $M, w \models \neg K_n\varphi \wedge \neg K_n\psi$ follows that there are v and $v' \in \mathcal{D}(M)$ such that $v \sim_n w$ and $M, v \models \neg\varphi$, and $v' \sim_n w$ and $M, v' \models \neg\psi$, respectively. As $\sim_n$ is an equivalence relation, we also have that $v \sim_n v$ and $v \sim_n v'$, we have as well $M, v \models \neg K_n\varphi \wedge \neg K_n\psi$; similarly, $M, v' \models \neg K_n\varphi \wedge \neg K_n\psi$. In other words, both v and v' are in the domain of $M|(\neg K_n\varphi \wedge \neg K_n\psi)$. As the value of boolean propositions only depends on the current factual state, from $M, v \models \neg\varphi$ and $v \in \mathcal{D}(M|(\neg K_n\varphi \wedge \neg K_n\psi))$ follows $M|(\neg K_n\varphi \wedge \neg K_n\psi), v \models \neg\varphi$; and from the last follows $M|(\neg K_n\varphi \wedge \neg K_n\psi), w \models \neg K_n\varphi$. Similarly, $M|(\neg K_n\varphi \wedge \neg K_n\psi), v' \models \neg\psi$; from which follows $M|(\neg K_n\varphi \wedge \neg K_n\psi), w \models \neg K_n\psi$. Therefore $M|(\neg K_n\varphi \wedge \neg K_n\psi), w \models \neg K_n\varphi \wedge \neg K_n\psi$, as required.

The case ii An agent can become ignorant from professing his own knowledge, and announcement *ii* is a typical example. This may sound strange[3], but it can easily be observed to be true for announcement *ii*: after *ii*, Product knows the numbers (formula *iii*), so it can no longer be true that Sum knows that Product does not know the numbers: in other words, formula *ii* is now false. Ergo, *ii* is an unsuccessful update.

The cases iii and iv The last two announcements *iii* and *iv* are successful formulas: this is because they are *preserved* formulas: they are truth preserving under submodel restrictions, an inductively defined fragment with – among other clauses – inductive clauses that atomic propositions are always preserved, and that if φ and ψ are preserved, then also $\varphi \wedge \psi$, $\varphi \vee \psi$, and $K_n\varphi$ [vB02]. The announcements *iii* and *iv* are disjunctions of formulas of the form $K_n(x_i \wedge y_j)$, and are therefore preserved. All preserved formulas are successful [vDK05]. And all successful formulas induce successful updates in all epistemic states.

No inductive definition of the successful formulas is known – in particular, if φ and ψ are both successful, $[\varphi]\psi$ may be unsuccessful. Having said that, it is remarkable that the sequence of the three announcements *ii* ; *iii* ; *iv* is an unsuccessful update, or, put in a single formula: $ii \wedge [ii]iii \wedge [ii \wedge [ii]iii]iv$ is unsuccessful in the initial epistemic state. This formula becomes false after its announcement: after that, just like after *ii*, Sum knows that Product knows the numbers, so it is now false that Sum knows that Product does not know the numbers: *ii* has become false, and therefore the entire conjunction corresponding to the sequence *ii* ; *iii* ; *iv*. The first announcement *i* can also be added to

[3]Stranger even, is that an agent can also become knowledgeable from professing his own ignorance, for which there are other examples.

the conjunction, so that $i \wedge ii \wedge [ii]iii \wedge [ii \wedge [ii]iii]iv$ is also unsuccessful in the initial epistemic state.

This last observation captures, we think, more than anything else our intuition that the Sum-and-Product problem is puzzling.

7 The Epistemic Model Checker DEMO

Recently, epistemic model checkers have been developed to verify properties of interpreted systems, knowledge-based protocols, and various other multi-agent systems. The model checkers MCK [GvdM04] and MCMAS [RL04] use the interpreted system architecture; MCK does this in a setting of linear and branching time temporal logic. The exploration of the search space in both MCK and MCMAS is based on ordered binary decision diagrams.

A different model checker, not based on a temporal epistemic architecture, is DEMO. It has been developed by Jan van Eijck [vE04]. DEMO is short for Dynamic Epistemic MOdelling. It allows modelling epistemic updates, graphical display of Kripke structures involved, and formula evaluation in epistemic states. DEMO is written in the functional programming language Haskell.

The model checker DEMO implements the dynamic epistemic logic of [BM04]. In this 'action model logic' the global state of a multi-agent system is represented by an epistemic model as in Section 3. 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 an operation called the *update product*. This is a restricted modal product. In this submission we restrict our attention to action models for public announcements. Such action models have a singleton domain, and the precondition of that point is the announced formula. We refrain from details and proceed with (a relevant part of – recursive clauses describing the effect of updates have been omitted) the recursive definition of formulas in DEMO.

```
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, w), then in case of the truthful public announcement of φ the resulting `EpistM` has the form $(M|\varphi, w)$. We can also update with a list of pointed action models:

```
upds ::  EpistM -> [PoAM] -> EpistM
```

An example is the sequence of three announcements in the Sum-and-Product problem.

8 Sum and Product in DEMO

We implement the Sum-and-Product riddle in DEMO and show how the implementation finds the unique solution $(4, 13)$. Figure 2 contains the implementation.

A list is a standard data structure in Haskell, unlike a set. The set $I \equiv \{(x, y) \in \mathbb{N}^2 \mid 1 < x < y \text{ and } x + y \leq 100\}$ is realized in DEMO as the list

```
pairs = [(x,y)| x<-[2..100], y<-[2..100], x<y, x+y<=100]
```

Thus, { and } are replaced by [and], $\in$ is replaced by `<-`, and instead of I we name it `pairs`. A pair such as $(4, 18)$ is not a proper name for a domain element. In DEMO, natural numbers are such proper names. Therefore, we associate each element in `pairs` with a natural number and make a new list.

```
ipairs = zip [0..numpairs-1] pairs
```

Here, `numpairs` is the number of elements in `pairs`, and the function `zip` pairs the i-th element in `[0..numpairs-1]` with the i-th element in `pairs`, and makes that the i-th element of `ipairs`. For example, the first element in `ipairs` is `(0,(2,3))`.

The initial model of the Sum-and-Product riddle is represented as

```
msnp :: EpistM
msnp = (Pmod [0..numpairs-1] val acc [0..numpairs-1])
  where
  val = [(w,[P x, Q y]) | (w,(x,y))<- ipairs]
  acc = [(a,w,v)| (w,(x1,y1))<-ipairs, (v,(x2,y2))<-ipairs, x1+y1==x2+y2 ]++
        [(b,w,v)| (w,(x1,y1))<-ipairs, (v,(x2,y2))<-ipairs, x1*y1==x2*y2 ]
```

```
module SNP
where
import DEMO

pairs  = [(x,y)|x<-[2..100], y<-[2..100], x<y, x+y<=100]
numpairs = llength(pairs)
llength [] =0
llength (x:xs) = 1+ llength xs
ipairs = zip [0..numpairs-1] pairs

msnp :: EpistM
msnp = (Pmod [0..numpairs-1] val acc [0..numpairs-1])
  where
  val = [(w,[P x, Q y]) | (w,(x,y))<- ipairs]
  acc = [(a,w,v)| (w,(x1,y1))<-ipairs, (v,(x2,y2))<-ipairs, x1+y1==x2+y2 ]++
        [(b,w,v)| (w,(x1,y1))<-ipairs, (v,(x2,y2))<-ipairs, x1*y1==x2*y2 ]

fmrs1e = K a (Conj [Disj[Neg (Conj [Prop (P x),Prop (Q y)]),
         Neg (K b (Conj [Prop (P x),Prop (Q y)]))]| (x,y)<-pairs])
amrs1e = public (fmrs1e)
fmrp2e = Conj [(Disj[Neg (Conj [Prop (P x),Prop (Q y)]),
     K b (Conj [Prop (P x),Prop (Q y)]) ] )|(x,y)<-pairs]
amrp2e = public (fmrp2e)
fmrs3e = Conj [(Disj[Neg (Conj [Prop (P x),Prop (Q y)]),
     K a (Conj [Prop (P x),Prop (Q y)]) ] )|(x,y)<-pairs]
amrs3e = public (fmrs3e)

solution = showM (upds msnp [amrs1e, amrp2e, amrs3e])
```

Figure 2: The DEMO program `SNP.hs`. Comment lines have been removed.

Here, `msnp` is a multi-pointed epistemic model, that consists of a domain `[0..numpairs-1]`, a valuation function `val`, an accessibility relation function `acc`, and `[0..numpairs-1]` points. As the points of the model are the entire domain, we may think of this initial epistemic state as the (not-pointed) epistemic model underlying it.

The valuation function `val` maps each state in the domain to the subset of atoms that are true in that state. This is different from Section 3, where the valuation V was defined as a function mapping each atom to the set of states where it is true. The correspondence $q \in \texttt{val}(w)$ iff $w \in V(q)$ is elementary. An element `(w,[P x, Q y])` in `val` means that in state `w`, atoms `P x` and `Q y` are true. For example, given that `(0,(2,3))` is in `ipairs`, `P 2` and `Q 3` are true in state `0`, where `P 2` stands for 'the smaller number is 2' and `Q 3` stands for 'the larger number is 3'. These same facts were described in the previous section by x_2 and y_3, respectively, as that gave the closest match with the original problem formulation. In DEMO, names of atoms *must* start with capital P, Q, R, but the correspondence between names will be obvious.

The function `acc` specifies the accessibility relations. Agent `a` represents Sum and agent `b` represents Product. For `(w,(x1,y1))` and `(v,(x2,y2))` in `ipairs`, if their sum is the same: `x1+y1==x2+y2`, then they cannot be distinguished by Sum: `(a,w,v)` in `acc`; and if their product is the same: `x1*y1==x2*y2`, then they cannot be distinguished by Product: `(b,w,v)` in `acc`. Function `++` is an operation merging two lists.

Sum and Product's announcements are modelled as singleton action models, generated by the announced formula (precondition) φ and the operation `public`. Consider $K_S\neg\bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$, expressing that Sum says: "I knew you didn't." This is equivalent to $K_S\bigwedge_{(i,j)\in I}\neg K_P(x_i \wedge y_j)$. A conjunct $\neg K_P(x_i \wedge y_j)$ in that expression, for 'Product does not know that the pair is (i,j)', is equivalent to $(x_i \wedge y_j) \rightarrow \neg K_P(x_i \wedge y_j)$.[4] The latter is computationally cheaper to check in the model, than the former: in all states but (i,j) of the model, the latter requires a check on two booleans only, whereas the former requires a check *in each of those states* of Product's ignorance, that relates to his equivalence class for that state, and that typically consists of several states.

This explains that the check on $\bigwedge_{(i,j)\in I}\neg K_P(x_i \wedge y_j)$ can be replaced by one on $\bigwedge_{(i,j)\in I}((x_i \wedge y_j) \rightarrow \neg K_P(x_i \wedge y_j))$. Using a model validity, the check on $\bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$ (Product knows the numbers) can also be replaced, namely by a check $\bigwedge_{(i,j)\in I}((x_i \wedge y_j) \rightarrow K_P(x_i \wedge y_j))$.[5] Using these observations, and writing an implication $\varphi \rightarrow \psi$ as $\neg\varphi \vee \psi$, the three problem announcements ii, iii, and iv listed on page 11 are checked in DEMO by the formulas `fmrs1e`, `fmrp2e`, and `fmrs3e`, respectively, as listed in Figure 2. The corresponding singleton action models are obtained by applying the function `public`, namely as `amrs1e = public (fmrs1e)`, `amrp2e = public (fmrp2e)`, and `amrs3e = public (fmrs3e)`. This is also shown in the figure.

Finally, we show a relevant part of DEMO interaction with this implementation. The complete (three-page) output of this interaction can be found on `www.cs.otago.ac.nz/staffpriv/hans/sumpro/`.

The riddle is solved by updating the initial model `msnp` with the action models corresponding to the three successive announcements:

```
*SNP> showM (upds msnp [amrs1e, amrp2e, amrs3e])
==> [0]
[0]
(0,[p4,q13])
(a,[[0]])
(b,[[0]])
```

This function `showM` displays a pointed epistemic model as:

```
==> [<points>]
```

[4] We use the $S5$-validity $\neg K\varphi \leftrightarrow (\varphi \rightarrow \neg K\varphi)$, that can be shown as follows: $\neg K\varphi$ iff $(\varphi \vee \neg\varphi) \rightarrow \neg K\varphi$ iff $(\varphi \rightarrow \neg K\varphi) \wedge (\neg\varphi \rightarrow \neg K\varphi)$ iff $(\varphi \rightarrow \neg K\varphi) \wedge (K\varphi \rightarrow \varphi)$ iff (in $S5$!) $(\varphi \rightarrow \neg K\varphi)$.

[5] We now use that $\varphi\overline{\vee}\psi$ – where $\overline{\vee}$ is exclusive disjunction – entails that ($K\varphi \vee K\psi$ iff $(\varphi \rightarrow K\varphi) \wedge (\psi \rightarrow K\psi)$).

```
[<domain>]
[<valuation>]
[<accessibility relations represented as equivalence classes>]
```

The list `[p4,q13]` represents the facts `P 4` and `Q 13`, i.e., the solution pair $(4, 13)$. Sum and Product have full knowledge (their access is the identity) on this singleton domain consisting of state `0`. That this state is named `0` is not a coincidence: after each update, states are renumbered starting from 0.

For another example, `(upds msnp [amrs1e,amrp2e])` represents the model that results from Product's announcement (*iii*) "Now I know the numbers." Part of the `showM` results for that model are

```
*SNP> showM (upds msnp [amrs1e,amrp2e])
==> [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,
(...)
(0,[p2,q9])(1,[p2,q25])(2,[p2,q27])(3,[p3,q8])(4,[p3,q32])
(5,[p3,q38])(6,[p4,q7])(7,[p4,q13])(8,[p4,q19])(9,[p4,q23])
(...)
(a,[[0,3,6],[1,9,14,23,27,32,37,44,50],[2,10,17,24,28,38,45,46,51],[4
,11,18,29,33,39,47,55,60,65],[5,12,25,35,41,48,52,56,57,62,67,70,73],
[7],[8,22,36],[13,20,26,42,53,58,63,68,71,74,76,79,81],[15,19,30,34,4
0,61,66],[16,21,31,43,49,54,59,64,69,72,75,77,78,80,82,83,84,85]])
(b,[[0],[1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12],[13],[14],
(...)
```

After two announcements 86 pairs (x, y) remain possible. All remaining states are renumbered, from 0 to 85, of which part is shown. Product's (`b`) access consists of singleton sets only, of which part is shown. That should be obvious, as he just announced that he knew the number pair. Sum's (`b`) equivalence class `[0,3,6]` is that for sum 11: note that `(0,[p2,q9])`, `(3,[p3,q8])`, and `(6,[p4,q7])` occur in the shown part of the valuation. Sum's access has one singleton equivalence class, namely `[7]`. That corresponds to the state for pair $(4, 13)$: see `(7,[p4,q13])` in the valuation. Therefore, Sum can now truthfully announce to know the pair of numbers, after which the singleton final epistemic state (that was already displayed) results.

9 Other model checkers and DEMO programs

As mentioned in the introduction to the previous section, other model checkers around are MCK [GvdM04], and MCMAS [RL04]. The question is whether we could also implement this problem in those model checkers. For the latest versions of these model checkers in both cases the answer appears to be 'no'.

The current version of MCK is 0.2.0. In MCK, a state of the environment is an assignment to a set of variables declared in the environment section. These variables are usually assumed to be partially accessible to the individual agents, and agents could share some variables. The change of the state of the multi-agent system is either made by agents or the environment, in the form

of changing these variables. There are two ways to make such changes. One is to send signals to the environment using the action construct by agents in conjunction with the transitions construct by the environment, which provides a way to describe how the environment variables are updated. The other is a specialized form for actions from the perspective that environment variables are shared variables, by providing read and write operations on those shared variables. In both cases, we need guarded statements to make the change. For example, a simple deterministic statement has the form:

$$\text{if } \mathit{cond} \rightarrow C \text{ [otherwise} \rightarrow C_o \text{] fi}$$

where command C is eligible for execution only if the corresponding condition *cond* evaluates to true in the current state. Otherwise, the command C_o will be executed. If we would like to model the Sum-and-Product problem in MCK, the effect of a public announcement should be recorded in a variable which is accessible to all agents. Suppose the effect of P's public announcement : "I now know it" $(K_P(x, y))$ is recorded in variable v. Then in a state just after this announcement, the variable v will be set to $True$ if $K_P(x, y)$ holds in the previous state, and otherwise to $False$. Clearly, we need that statement in the above *epistemic* form, with *cond* involving knowledge checking. Unfortunately, even though in MCK we can check epistemic postconditions, the current version of MCK does not support checking epistemic formulas as preconditions, as in *cond*. This might possibly be related to inherent difficulties to incorporate knowledge in *cond*, but an extension seems called for.

The latest MCMAS is version 0.7. The underlying theory has been developed by Alessio Lomuscio. It can be seen as a continuation of his PhD work on hypercube systems, which are a special class of interpreted systems [Lom99]. Similarly to MCK, MCMAS also does not support actions with knowledge-based preconditions to transit from one global state to another global state.

Apart from the Sum-and-Product riddle we have implemented some of the other riddles discussed in this paper in DEMO, such as the 'Domiciliary' problem in Section 2.2. All these programs are mere variations of the one presented in this paper, because the communications are always similar public announcements of knowledge and ignorance in a two-agent system, whereas only the 'starting conditions' – the number (or symbol) pairs initially allowed – vary from problem to problem. For these programs, including full explanations, we refer to the website `www.cs.otago.ac.nz/staffpriv/hans/sumpro/`.

10 Conclusions

We have modelled the Sum-and-Product problem in public announcement logic and verified its properties in the epistemic model checker DEMO. The problem can be represented in the traditional way by number pairs, so that Sum knows their sum and Product their product, but also as an interpreted system with (sum,product) pairs. Subject to the union of languages, the representations are

bisimilar, and even isomorphic. We also analyzed which announcements made towards a solution of the problem were unsuccessful updates – formulas that become false because they are announced.

A final word on model checking such problems: originally, an analysis involving elementary number theory and combinatorics was necessary to solve the problem. Indeed, that was the whole fun of the problem. Solving it in a model checker instead, wherein one can, in a way, simply state the problem in its original epistemic formulation, hides all that combinatorial structure and makes it appear almost trivial. Far from trying to show that the problem is therefore actually trivial or uninteresting, this rather shows how powerful model checking tools may be, when knowledge specifications are clear and simple but their structural ramifications complex.

References

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, Cambridge, 2001. Cambridge Tracts in Theoretical Computer Science 53.

[BM96] J. Barwise and L.S. Moss. *Vicious Circles*. CSLI Publications, Stanford, 1996.

[BM04] A. Baltag and L.S. Moss. Logics for epistemic programs. *Synthese*, 139:165–224, 2004. Knowledge, Rationality & Action 1–60.

[BMS98] A. Baltag, L.S. Moss, and S. Solecki. The logic of common knowledge, public announcements, and private suspicions. In I. Gilboa, editor, *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 98)*, pages 43–56, 1998.

[FHMV95] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge MA, 1995.

[Fre69] H. Freudenthal. (formulation of the sum-and-product problem). *Nieuw Archief voor Wiskunde*, 3(17):152, 1969.

[Fre70] H. Freudenthal. (solution of the sum-and-product problem). *Nieuw Archief voor Wiskunde*, 3(18):102–106, 1970.

[Gar79] M. Gardner. Mathematical games. *Scientific American*, 241 (December):20–24, 1979. Also addressed in the March (page 24) and May (pages 20–21) issues of volume 242, 1980.

[Ger99] J.D. Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, University of Amsterdam, 1999. ILLC Dissertation Series DS-1999-01.

[Ger05] J.D. Gerbrandy. The surprise examination in dynamic epistemic logic. *Synthese*, 2005. To appear.

[Gre68] M. H. Greenblatt. *Mathematical Entertainments: A Collection of Illuminating Puzzles New and Old*. George Allen and Unwin Ltd, London, 1968.

[GvdM04] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In R. Alur and D. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification (CAV 2004)*, pages 479–483. Springer, 2004.

[Hin62] J. Hintikka. *Knowledge and Belief*. Cornell University Press, Ithaca, NY, 1962.

[Isa95] I.M. Isaacs. The impossible problem revisited again. *The Mathematical Intelligencer*, 17(4):4–6, 1995.

[Koo05] B.P. Kooi. On public update logics. Under submission, 2005.

[Lew69] D.K. Lewis. *Convention, a Philosophical Study*. Harvard University Press, Cambridge (MA), 1969.

[Liu04] A. Liu. Problem section: Problem 182. *Math Horizons*, 11:324, 2004.

[Lom99] A.R. Lomuscio. *Knowledge Sharing among Ideal Agents*. PhD thesis, University of Birmingham, Birmingham, UK, 1999.

[McC90] J. McCarthy. Formalization of two puzzles involving knowledge. In Vladimir Lifschitz, editor, *Formalizing Common Sense : Papers by John McCarthy*, Ablex Series in Artificial Intelligence. Ablex Publishing Corporation, Norwood, N.J., 1990. original manuscript dated 1978–1981.

[MDH86] 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.

[Moo42] 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).

[Pan91] G. Panti. Solution of a number theoretic problem involving knowledge. *International Journal of Foundations of Computer Science*, 2(4):419–424, 1991.

[Pla89] J.A. Plaza. Logics of public communications. In M.L. Emrich, M.S. Pfeifer, M. Hadzikadic, and Z.W. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, 1989.

[RL04] Franco Raimondi and Alessio Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 04)*, pages 630–637. IEEE Computer Society, 2004.

[Sal95] L. Sallows. The impossible problem. *The Mathematical Intelligencer*, 17(1):27–33, 1995.

[Sch72] Stephen Schiffer. *Meaning*. Oxford University Press, Oxford, 1972.

[vB98] J.F.A.K. van Benthem. Dynamic odds and ends. Technical report, ILLC, University of Amsterdam, 1998. Report ML-1998-08.

[vB02] J.F.A.K. van Benthem. One is a lonely number: on the logic of communication. Technical report, ILLC, University of Amsterdam, 2002. Report PP-2002-27 (material presented at the Logic Colloquium 2002).

[vdHV02] W. van der Hoek and L.C. Verbrugge. Epistemic logic: a survey. In L.A. Petrosjan and V.V. Mazalov, editors, *Game theory and Applications*, volume 8, pages 53–94, 2002.

[vDK05] H.P. van Ditmarsch and B.P. Kooi. The secret of my success. *Synthese*, 2005. To appear.

[vdM94] R. van der Meyden. Mutual belief revision. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 595–606. Morgan Kaufmann, 1994.

[vDRV05] H.P. van Ditmarsch, J. Ruan, and L.C. Verbrugge. Model checking sum and product. In *Proceedings of the 18th Australian Joint Conference on Artificial Intelligence (AI 2005)*, pages 790–795. Springer Verlag, 2005. LNAI 3809.

[vDvdHK03] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. Descriptions of game states. In G. Mints and R. Muskens, editors, *Logic, Games, and Constructive Sets*, pages 43–58, Stanford, 2003. CSLI Publications. CSLI Lecture Notes No. 161.

[vDvdHK05] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. Dynamic epistemic logic. Manuscript, 2005.

[vE04] J. van Eijck. Dynamic epistemic modelling. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, 2004. CWI Report SEN-E0424.

[WS40] W. T. Williams and G. H. Savage. *The Penguin Problems Book: A Modern Anthology of Perplexities and Tantalizers*. Penguin Books (Allen Lane), Harmondsworth, 1940.

[WS44] W. T. Williams and G. H. Savage. *The Second Penguin Problems Book*. Penguin Books, Harmondsworth, 1944.