

Model Checking Sum and Product

H.P. van Ditmarsch¹ and J. Ruan^{1*} and L.C. Verbrugge^{2**}

¹ University of Otago, New Zealand, {hans,jruan}@cs.otago.ac.nz

² University of Groningen, Netherlands, rineke@ai.rug.nl

Abstract. We model the well-known Sum-and-Product problem in a modal logic, and verify its solution in a model checker. The modal logic is public announcement logic. The riddle is then implemented and its solution verified in the epistemic model checker DEMO.

1 Introduction

The Sum-and-Product problem was first stated—in Dutch—in [1]:

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:

1. *P* says: “I do not know it.”
2. *S* says: “I knew you didn’t.”
3. *P* says: “I now know it.”
4. *S* says: “I now also know it.”

Determine the pair (x, y) .

This problem is, that 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 14 and 16: if they were, their sum would be 30. This is also the sum of 7 and 23. If those were the numbers their product would have been 161 which, as these are prime numbers, *only* is the product of 7 and 23. So Product (*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 an example, to be able to determine the pair of numbers: the unique solution of the problem is the pair (4, 13).

* 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).

Logical approaches to solve the problem are found in [2–5]. As far as we know, we are the first to use an automated model checker to tackle the Sum-and-Product problem.

In Section 2 we model the Sum-and-Product problem in public announcement logic. In Section 3 we implement the Sum-and-Product specification of Section 2 in DEMO and verify its epistemic features.

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 [6]. We give a concise overview of the logic.

Language Given are a set of agents N and a 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)’.

Structures An *epistemic model* $M = \langle W, \sim, V \rangle$ consists of a *domain* W of (factual) *states* (or ‘worlds’), *accessibility* $\sim : N \rightarrow \mathcal{P}(W \times W)$, and a *valuation* $V : Q \rightarrow \mathcal{P}(W)$. 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 .

Semantics Assume an epistemic model $M = \langle W, \sim, V \rangle$.

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

The group accessibility relation \sim_G is the transitive and reflexive closure of the union of all access for the individuals in G : $\sim_G \equiv (\bigcup_{n \in G} \sim_n)^*$. Epistemic model $M|_{\varphi} = \langle W', \sim', V' \rangle$ is defined as

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

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 $M|_{\varphi}, w \models \psi$. Validity and logical consequence are defined in the standard way. For a proof system, see [7].

To 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. 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 $\{x_i \mid (i, j) \in I\} \cup \{y_j \mid (i, j) \in I\}$. The set of agents is $\{S, P\}$; S and P will also be referred to as Sum and Product, respectively.

A proposition such as ‘Sum knows that the numbers are 4 and 13’ is described 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 described as $K_P(x, y) \equiv \bigvee_{(i,j) \in I} K_P(x_i \wedge y_j)$. This is sufficient to formalize the announcements made towards a solution of the problem:

1. P says: “I do not know it”: $\neg K_P(x, y)$
2. S says: “I knew you didn’t”: $K_S \neg K_P(x, y)$
3. P says: “I now know it”: $K_P(x, y)$
4. 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 $V_{y_j} = \{(x, y) \in I \mid y = j\}$. The solution of the problem is represented by the truth of the statement

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

or, properly expressing that (4, 13) is the only solution, by the model validity

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

Note that announcement 1 by Product is superfluous in the analysis. The ‘knew’ in announcement 2, by Sum, refers to the truth of that announcement in the *initial* epistemic state, not in the epistemic state *resulting* from announcement 1, by Product.

3 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 [8] and MCMAS [9] have a temporal epistemic architecture, and exploration of the search space is based on ordered binary decision diagrams. The epistemic model checker DEMO, developed by Jan van Eijck [10], is not based on temporal epistemics. 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

```

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])

```

Fig. 1. The DEMO program `SNP.hs`. Comment lines have been removed.

is written in the functional programming language Haskell. The model checker DEMO implements the dynamic epistemic logic of [7]. For a comparative study of these three model checkers, on a different problem, see [11]. We have specified the ‘Sum and Product riddle’ in DEMO only. The verification of a comparable specification in MCK exceeds its computational power (and this is also to be expected for MCMAS), although clever restriction of variables might well bring such verification with reach.

Figure 1 contains the specification of the Sum and Product riddle in DEMO. 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]`. 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`.

The initial model `msnp` of the Sum-and-Product riddle (see Figure 1) is a multi-pointed epistemic model, that consists—this is the line `msnp = (Pmod [0..numpairs-1] val acc [0..numpairs-1])` in the program—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 our previous definition of a valuation V , but the correspondence $q \in \text{val}(w)$ 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 .

The function `acc` specifies the accessibility relations. Agent `a` represents Sum and agent `b` represents Product. For $(w, (x_1, y_1))$ and $(v, (x_2, y_2))$ in `ipairs`, if their sum is the same: $x_1 + y_1 = x_2 + y_2$, then they cannot be distinguished by Sum: (a, w, v) in `acc`; and if their product is the same: $x_1 * y_1 = x_2 * y_2$, 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 structures called ‘singleton action models’, generated by the announced formula (precondition) φ and an operation `public`. For our purposes it is sufficient to focus on that precondition.

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 $(x_i \wedge y_j) \rightarrow \neg K_P(x_i \wedge y_j)$. 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 $\bigwedge_{(i,j) \in I} ((x_i \wedge y_j) \rightarrow K_P(x_i \wedge y_j))$. Using these observations, and writing an implication $\varphi \rightarrow \psi$ as $\neg\varphi \vee \psi$, the three problem announcements 2, 3, and 4 listed on page 3 are checked in DEMO in by the formulas `fmsr1e`, `fmrp2e`, and `fmsr3e`, respectively, as listed in Figure 1. The corresponding singleton action models are obtained by applying the function `public`, for example, `amrs1e = public (fmsr1e)`.

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 with, on successive lines, point `[0]`, domain `[0]`—after each update, states are renumbered starting from 0—, valuation `(0, [p4, q13])`—representing the facts `P 4` and `Q 13`, i.e., the solution pair `(4, 13)`—, and accessibility relations `(a, [[0]])` and `(b, [[0]])`—Sum and Product have full knowledge, as their access is the identity. Intermediate results of the computation can also be given. For the complete output of such interaction, see www.cs.otago.ac.nz/staffpriv/hans/sumpro/.

References

1. Freudenthal, H.: (formulation of the sum-and-product problem). *Nieuw Archief voor Wiskunde* **3(17)** (1969) 152
2. McCarthy, J.: Formalization of two puzzles involving knowledge. In Lifschitz, V., ed.: *Formalizing Common Sense : Papers by John McCarthy*. Ablex series in artificial intelligence. Ablex Publishing Corporation, Norwood, N.J. (1990) original manuscript dated 1978–1981.
3. Plaza, J.: Logics of public communications. In Emrich, M., Pfeifer, M., Hadzikadic, M., Ras, Z., eds.: *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*. (1989) 201–216
4. Panti, G.: Solution of a number theoretic problem involving knowledge. *International Journal of Foundations of Computer Science* **2(4)** (1991) 419–424
5. van der Meyden, R.: Mutual belief revision. In Doyle, J., Sandewall, E., Torasso, P., eds.: *Proceedings of the 4th international conference on principles of knowledge representation and reasoning (KR)*, Morgan Kaufmann (1994) 595–606
6. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: *Reasoning about Knowledge*. MIT Press, Cambridge MA (1995)
7. Baltag, A., Moss, L., Solecki, S.: The logic of public announcements, common knowledge, and private suspicions. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam (1999) CWI Report SEN-R9922.
8. Gammie, P., van der Meyden, R.: MCK: Model checking the logic of knowledge. In Alur, R., Peled, D., eds.: *Proceedings of the 16th International conference on Computer Aided Verification (CAV 2004)*, Springer (2004) 479–483
9. Raimondi, F., Lomuscio, A.: Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In: *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, IEEE Computer Society (2004) 630–637
10. van Eijck, J.: Dynamic epistemic modelling. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam (2004) CWI Report SEN-E0424.
11. van Ditmarsch, H., van der Hoek, W., van der Meyden, R., Ruan, J.: Model checking russian cards. *Electronic Notes in Theoretical Computer Science* (2005) To appear; presented at MoChArt 05 (Model Checking in Artificial Intelligence).