An Inconsistent Theory of Belief

Richard W. Weyhrauch *
Marco Cadoli † Carolyn L. Talcott ‡

March 14, 1996

Prolog

The purpose of this note is to prove that if an agent B believes both that agent A does not know all tautologies, and that A has some simple propositional reasoning capability then B has an inconsistent belief system. More generally if B attributes to A simple reasoning capability in any theory, then this amounts to the assertion that if anyone knows a theorem of the theory, then A must know it.

1 Introduction

It seems clear to us(clt + rww) that an agent may fail to know whether a long formula is a tautology. Thus believe that any agent would be willing to assert the following.

(**NK**) $\forall a \exists w.(Taut(w) \wedge \neg Knows(a, w))$

*IBUKI Inc., PO Box 1627, Los Altos, CA 94022. E-mail: `rww@ibuki.com`

†Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, I-00198 Roma, Italy. E-mail: `cadoli@dis.uniroma1.it`

‡Department of Computer Science, Stanford University, Stanford, CA 94305. E-mail: `clt@cs.stanford.edu`

Now consider the following axioms:

(Ax1) $\quad (W_1 \Rightarrow (W_2 \Rightarrow W_1))$

(Ax2) $\quad (W_1 \Rightarrow W_2) \Rightarrow ((W_1 \Rightarrow (W_2 \Rightarrow W_3)) \Rightarrow (W_1 \Rightarrow W_3))$

(Ax3) $\quad (W_1 \Rightarrow W_2) \Rightarrow ((W_1 \Rightarrow \neg W_2) \Rightarrow \neg W_1)$

(Ax4) $\quad \neg\neg W_1 \Rightarrow W_1$

These four axiom schemata together with modus ponens yield a proof system satisfying

(C) $\quad \forall w.(\mathit{Taut}(w) \Leftrightarrow \mathit{Provable}(w))$

where w ranges over propositional formulas, $\mathit{Taut}(w)$ holds iff w is a tautology of propositional calculus, $\mathit{Provable}(w)$ holds iff w can be proved using the proof system. [1]

We now show that if B believes **NK**, and also believes that A knows the proof system above, then a formal contradiction is straight forward. Namely, we show that for some tautology w, both $\neg \mathit{Knows}(A, w)$, and $\mathit{Knows}(A, w)$ can be proved. Notice that, in this case, it is the agent B, not the agent A, that is inconsistent.

2 Laying out the formal argument

B formalizes this simple propositional reasoning capability of an agent A as follows. The four axiom schemata are expressed by

(KAx1) $\quad \forall w_1\ w_2.\mathit{Knows}(A, w_1 \overset{\circ}{\Rightarrow} (w_2 \overset{\circ}{\Rightarrow} w_1))$

(KAx2) $\quad \forall w_1\ w_2\ w_3.\mathit{Knows}(A, (w_1 \overset{\circ}{\Rightarrow} w_2) \overset{\circ}{\Rightarrow} ((w_1 \overset{\circ}{\Rightarrow} (w_2 \overset{\circ}{\Rightarrow} w_3)) \overset{\circ}{\Rightarrow} w_1 \overset{\circ}{\Rightarrow} w_3))$

(KAx3) $\quad \forall w_1\ w_2.\mathit{Knows}(A, (w_1 \overset{\circ}{\Rightarrow} w_2) \overset{\circ}{\Rightarrow} ((w_1 \overset{\circ}{\Rightarrow} \overset{\circ}{\neg} w_2) \overset{\circ}{\Rightarrow} \overset{\circ}{\neg} w_1))$

(KAx4) $\quad \forall w_1.\mathit{Knows}(A, \overset{\circ}{\neg}\overset{\circ}{\neg} w_1 \overset{\circ}{\Rightarrow} w_1)$

Or, more concisely

(KAx) $\quad \forall w.(\mathit{IsAx}(w) \Rightarrow \mathit{Knows}(A, w))$

Modus ponens corresponds to

(KA) $\forall w_1 w_2 \,.\, Knows(A, w_1) \wedge Knows(A, w_1 \overset{\circ}{\Rightarrow} w_2) \Rightarrow Knows(A, w_2)$

where w_1, w_2, w_3 range over propositional formulas $w_1 \overset{\circ}{\Rightarrow} w_2$ is the formula "w_1 implies w_2". The predicate $IsAx(w)$ holds if w is an instance of one of the axiom schemes. The relation $Knows(A, w)$ expresses the fact that agent A knows that the formula w is true. From this we can prove:

(**O**) $\forall w.(Taut(w) \Rightarrow Knows(A, w))$

This somewhat counterintuitive consequence is a form of *logical omniscience* for A. Notice that it contradicts **NK** and thus B's theory of A's knowledge is inconsistent.

We observe that, by **(C)**, to establish **(O)** it is sufficient to establish

(**O1**) $\forall a\ w.(Provable(w) \Rightarrow Knows(a, w))$.

We prove **(O1)** by induction on the proof of w. In more detail, assume that w is a tautology. Thus by **(C)** we may assume that we have a proof of w. Such a proof is either an axiom or an application of modus ponens to two smaller proofs.

- The base case is when the proof of w is an instance of an axiom schemata. By **(KAx)** $Knows(A, w)$ holds.

- The inductive case is when the proof of w is an application of modus ponens to proofs of formulas w_1 and $w_1 \overset{\circ}{\Rightarrow} w$ for some formula w_1. By the induction hypothesis $Knows(A, w_1)$ and $Knows(A, w_1 \overset{\circ}{\Rightarrow} w)$ hold. Hence, by axiom (**KA**), $Knows(A, w)$ holds, and we are done.

A formalization of this argument in an implementation of FOL is given below.

3 Laying Blame

In this note we showed the simple result that if an agent, B, believes about agent A that

(1) there is some tautology that A does not know, **(NK)**

(2) A has a simple propositional reasoning capability, **(KAx,KA)**.

then B has an inconsistent belief system. With an inconsistent belief system B will believe everything!!!! Simply put, if we imagine an autonomous agent could have the beliefs (1) and (2) it could not be expected to survive long.

Since we have stated that we consider (1) to be self-evident, this leaves (2) to blame. On the other hand, it seem like the attribution of simple propositional knowlege to others causes us to be inconsistent!!! Why?
Challenge: An adaquate representation theory must account resolve this dilemma. We give the beginning of one possible explanation in [2].

References

[1] A. Church. *Introduction to Mathematical Logic, Vol. I.* Princeton University Press, 1956.

[2] R. W. Weyhrauch, M. Cadoli, and C. L. Talcott. Using abstract resources to control reasoning, 1995. submitted for publication.

Appendix: The formal argument in FOL.

This presentation is carried out in GETFOL [?] an implementation of the FOL ideas.

```
comment #  We begin with B's propositional meta theory #

comment #  Propositional formulas as  data structures #

declare sort Propvar Wff;
moregeneral Wff <Propvar> ;
declare funconst mknot (Wff) = Wff;
declare funconst mkimp (Wff, Wff) = Wff;
declare funconst notWffof impHyp impConc (Wff) = Wff;
declare indvar w w1 w2 w3 [Wff];
theory WffAdt:
  forall w.notWffof(mknot(w)) = w
  forall w1 w2.impHyp(mkimp(w1,w2)) = w1
  forall w1 w2.impConc(mkimp(w1,w2)) = w2;

comment # Functions for manipulating proofs #

declare indvar d d1 d2;
declare funconst mkax 1;
declare funconst mkmp 2;
declare funconst Concl 1;
theory Concl:
  forall w.Concl(mkax(w)) = w
  forall d1 d2.Concl(mkmp(d1,d2)) = impConc(Concl(d2));

comment # Well-formed Proofs #

declare predconst Isax Proof 1;
axiom Proof1: forall w.(Isax(w) imp Proof(mkax(w)));
axiom Proof2:
forall w1 w2 d1 d2.
  ((Proof(d1) and  Concl(d1) = w1 and
    Proof(d2) and  Concl(d2) = mkimp(w1,w2))
```

```
    imp Proof(mkmp(d1,d2)));

declare predpar Q 1;
axiom ProofInd:
  (forall w.(Isax(w) imp Q(mkax(w))) and
   forall w1 w2 d1 d2.
    (Q(d1) and Q(d2) and Concl(d1) = w1 and Concl(d2) = mkimp(w1,w2)
        imp Q(mkmp(d1,d2))))
   imp forall d.(Proof(d) imp Q(d));

Comment # We state property C as an axiom #

declare predconst Taut 1;
axiom C:
  forall w.(Taut(w) imp exists d.(Proof(d) and Concl(d) = w));

Comment # Agents and attribution of knowledge #

declare sort Agent;
declare predconst Knows 2;
declare indvar a [Agent];
declare indconst A [Agent];
axiom NK:
  forall a.exists w.(Taut(w) and not(Knows(a,w)));
axiom AX:
  forall w.(Isax(w) imp Knows(A,w)) ;
axiom KA:
  forall w1 w2.(Knows(A,w1) and Knows(A,mkimp(w1,w2)) imp Knows(A,w2));

comment #
  We prove  (*) forall  w.(Taut(w) imp Knows(A,w))
  and then conclude FALSE using NK

  First prove a lemma:   forall d . (Proof(d) imp  Knows(A,Concl(d)))
  proof is by ProofInd (lines 1-17)

  Taking Q(d)  to be  Knows(A,Concl(d)) we must show

  (ax) forall w.(Isax(w) imp  Knows(A,Concl(mkax(w))))
```

```
    by Isax(w) imp Concl(mkax(w)) = w  and AX

   (lines 1-4)

  and

  (mp) forall w1 w2 d1 d2.
     (Knows(A,Concl(d1)) and Knows(A,Concl(d2)) and
      Concl(d1) = w1 and Concl(d2) = mkimp(w1,w2)
       imp
      Knows(A,Concl(mkmp(d1,d2))))

   (lines 5-16)

  Use C to conclude (*) (lines 18-25)

and here it is #

alle AX w;
1   Isax(w) imp Knows(A, w)

alle Concl1 w;
2   Concl(mkax(w)) = w

subst ^2 ^1 occ 2 right;
3   Isax(w) imp Knows(A, Concl(mkax(w)))

alli ^1 w;
4   forall w. (Isax(w) imp Knows(A, Concl(mkax(w))))

label fact ax=4;

assume Knows(A, Concl(d1)) and Knows(A, Concl(d2))
       and Concl(d1) = w1 and Concl(d2) = mkimp(w1, w2);
5   ((Knows(A, Concl(d1)) and Knows(A, Concl(d2))) and
     (Concl(d1) = w1)) and (Concl(d2) = mkimp(w1, w2))     (5)

label fact  IH = 5;
```

```
alle WffAdt3 w1 w2;
6   impConc(mkimp(w1, w2)) = w2

tauteq Concl(d2) = mkimp(w1, w2) by IH;
7   Concl(d2) = mkimp(w1, w2)     (5)

subst ^2 ^1  right;
8   impConc(Concl(d2)) = w2     (5)

alle Concl2 d1 d2;
9   Concl(mkmp(d1, d2)) = impConc(Concl(d2))

subst ^1 ^2;
10   Concl(mkmp(d1, d2)) = w2     (5)

alle KA w1 w2;
11   (Knows(A, w1) and Knows(A, mkimp(w1, w2))) imp Knows(A, w2)

tauteq Knows(A, w2) by IH ^1;
12   Knows(A, w2)     (5)

subst ^1 ^3 right;
13   Knows(A, Concl(mkmp(d1, d2)))     (5)

impi IH ^1;
14   (((Knows(A, Concl(d1)) and Knows(A, Concl(d2))) and
       (Concl(d1) = w1)) and (Concl(d2) = mkimp(w1, w2)))
      imp Knows(A, Concl(mkmp(d1, d2)))

alli ^1 w1 w2 d1 d2;
15   forall w1 w2 d1 d2. (
       (((Knows(A, Concl(d1)) and Knows(A, Concl(d2))) and
         (Concl(d1) = w1)) and (Concl(d2) = mkimp(w1, w2)))
        imp Knows(A, Concl(mkmp(d1, d2))))

andi ax ^1;
16   forall w. (Isax(w) imp Knows(A, Concl(mkax(w)))) and
     forall w1 w2 d1 d2. (
       (((Knows(A, Concl(d1)) and Knows(A, Concl(d2))) and
```

```
         (Concl(d1) = w1)) and (Concl(d2) = mkimp(w1, w2)))
        imp Knows(A, Concl(mkmp(d1, d2))))

impe ^1 ProofInd Q: lambda d. Knows(A, Concl(d));
17   forall d. (Proof(d) imp Knows(A, Concl(d)))

label fact Lemma = 17;

assume Taut(w);
18   Taut(w)     (18)

label fact Thyp=18;

alle C w;
19   Taut(w) imp exists d. (Proof(d) and (Concl(d) = w))

impe ^1 ^2;
20   exists d. (Proof(d) and (Concl(d) = w))     (18)

existe ^1 d;
21   Proof(d) and (Concl(d) = w)     (21)

alle Lemma d;
22   Proof(d) imp Knows(A, Concl(d))

tauteq Knows(A, w) by ^2 ^1;
23   Knows(A, w)     (18)

impi Thyp ^1;
24   Taut(w) imp Knows(A, w)

alli ^1 w;
25   forall w. (Taut(w) imp Knows(A, w))

alle NK A;
26   exists w. (Taut(w) and (not Knows(A, w)))

monad FALSE by ^1 ^2;
27   FALSE
```