

CONSISTENCY OF THEORIES OF IDEAS

Bob Elschlager
Computer Science Department
Stanford University
Stanford, California 94305

Inconsistency is a problem with first order predicate calculus theories in which ideas are treated as objects. This paper presents general results on inconsistency, including an interesting restriction on the relation of knowing. A theory having a uniform method of expressing ideas, ideas of ideas, and so on, is sketched. It is shown how to restrict some of the axioms of this theory in order to obtain consistency. The method of obtaining consistency seems to have some generality.

1. INTRODUCTION

As has been noted, for example, in [4], one of the problems with first order predicate calculus (PC) theories that objectify ideas is inconsistency. The next section describes a consistent theory that treats ideas as objects. It describes the axioms AX of this theory, indicates the restrictions that make the theory consistent, briefly discusses the notion of constructive idea, and then deals with some of the details of the formulation. After that, it is shown how easily inconsistency can arise in such theories. Then there is stated a theorem which says that in spite of these inconsistency results the theory described here, along with a number of its extensions, is consistent.

2. THE THEORY

The axiom system AX of the theory states that everyone knows the axioms of logic and all that is logically deducible from them. The axiom system also states that whatever anyone knows is true, and that if one knows something then one knows that one knows it. There are also axioms that state what it means for a variable to be *free* in a string, as well as what it means for one string to be *free* for a variable in another string (these standard concepts of logic are used in specifying the logical axioms in the PC).

Most important for consistency considerations are the axioms that specify the relation of denotation. It is desired that every (closed) string should *denote* its value, or in other words, that the string should denote what it *names* (examples of naming are given below). As will be seen in the section on inconsistency, such a desire can easily lead to paradox. Instead we make the weaker assumption (axiom a4) that every (closed) *determinate* string denotes its value. (Determinate strings, defined in the next section, are a very general class of strings, yet they avoid paradox.) By thus weakening the assumption about denote, one can prove that any of a variety of extensions of the axiom system AX are consistent.

The final axioms of the system state that whatever is a logical consequence of any of these axioms and is determinate, is known by everyone in all states (these last

axioms are defined inductively). I do not see that the determinate restriction in these final axioms is necessary, but I have not yet seen how to prove consistency without it.

At this point the notion of *constructive* can be briefly mentioned. One can optionally introduce this notion into the formulation (the consistency results below apply in either case). A constructive idea, together with a fixed body of knowledge, allows a person to identify the object named by the idea. The notion is related to the distinction between "being aware of" and "actually knowing". Thus the idea "number-of-people-in(room1)" makes one aware of a number without actually knowing what the number is. A person may be able deduce facts about a number (without actually knowing the number), but one actually knows the number only after deducing that the idea denotes the same thing as a constructive idea. In the notation illustrated later,

$$\exists U \text{ constructive}(U) \wedge \\ K(p, \neg^*(\text{number-of-people-in}^*(\text{room1}^*), U), s).$$

The notion of "can" is defined not only as knowing, but constructively knowing that an action will achieve a condition.

The remainder of this section sketches some of the details of the theory.

The language L of the theory is a (first order PC) language with sorts. It contains a set of basic symbols, and these can be *starred* any number of times to produce new symbols. For example, the starred versions of, say, symbols "K", ">", or "3" are "K*", ">*", or "3*", while the starred version of these are "K**", ">**", or "3**"; these are nine distinct symbols.

For purposes of explanation, ideas are concretized as strings of L (the word "string" is used to mean either a term or formula of L). There will be no indication of the sorts, although variables that range over strings are in upper case, variables ranging over variables are in gothic, and usually, but not necessarily, lower case is reserved for variables that range over objects other than ideas.

For examples of the intended use of the symbols of L and of the notion of naming, consider,

- (1) Ted
- (2) 3
- (3) safel
- (4) Ted^{*}
- (5) 3^{*}
- (6) safel^{*}
- (7) >(5,3)
- (8) >^{*}(5^{*},3^{*})
- (9) >^{*}(5^{*},3^{*})
- (10) >(x,3)
- (11) >^{*}(x^{*},3^{*})
- (12) >^{*}(X^{*},3^{*})
- (13) $\exists x >(x,3)$
- (14) $\exists (x^* > (x^*, 3^*))$
- (15) $\exists (X^* > (x^*, 3^*))$

Strings 1-3 name Ted, 3, and safel, respectively. On the other hand, 4-6 name the strings "Ted", "3", and "safel", respectively.

String 7 says that 5 is greater than 3, and therefore it names true (this is a truth independent of states). Strings 8-9 name the strings ">(5,3)" and ">^{*}(5^{*},3^{*})", respectively. Whether 10 names true or false depends on the value assigned to the variable x.

String 11 names the string ">(x,3)". What 12 names depends on the value assigned to the variable X; if X is assigned "6", "7+3", or "y+6", then 12 would name ">(6,3)", ">(7+3,3)", or ">(y+6,3)", respectively. The string 13 names true since there is a number greater than 3, while 14 names the string " $\exists x >(x,3)$ ". What 15 names depends on the value assigned to X; if X were assigned "x", "y", or "z", then 15 would name " $\exists x >(x,3)$ ", " $\exists y >(x,3)$ ", or " $\exists z >(x,3)$ ", respectively.

One important relation is that of knowing. This relation has three arguments; the first argument is a person, the second argument a string (of the sort that names a truth value), and the third argument a state. The intended interpretation is that in the state the person knows that the string is true. Thus, the first of

- (16) $K(\text{Ted}, >(5,3), \text{state1})$
- (17) $K(\text{Ted}, K^*(\text{Ted}^*, >^*(5^*, 3^*), \text{state2}^*), \text{state1})$

says that in state1 Ted knows that ">(5,3)" is true, or, that 5 is greater than 3; formula 17 says that in state1 Ted knows that in state2 Ted knew (will know) that 5 is greater than 3.

Another special relation is that for denotation, represented by the relation symbol "denote". Examples are,

- (18) $\text{denote}(3^*, 3)$
- (19) $\text{denote}(\exists^*(x^* > (x^*, 0^*)), \text{TRUE})$

The first says that the string "3" denotes 3, the second that the string " $\exists x >(x,0)$ " denotes true.

For the purpose of conveniently specifying axiom schemas, two mappings from strings to strings are defined: these mappings are not functions in the model. If B is a formula, $(cl\ B)$ indicates the closure of B, i.e., B with all its free variables universally quantified. Thus,

- (20) $cl\ (" \exists z (x+y = z) ") = " \forall x y \exists z (x+y = z) "$

A string will be said to be *closed* if it has no free variables; so if B is closed, then $(cl\ B)$ is the same as B.

The mapping *id* replaces all symbols by their starred versions, changing quantifiers into functions.

- (21) $id(">(x,5)") = ">^*(x^*, 5^*)"$
- (22) $id("K(\text{Streleski}, >^*(5^*, 3^*), \text{state1})") = "K^*(\text{Streleski}^*, >^*(5^*, 3^*), \text{state1}^*)"$
- (23) $id(" \exists x >(x,3) ") = " \exists^*(x^* > (x^*, 3^*))"$

Another convention for specifying schemas is that B and C are syntactic variables that range over strings; they are not variables of L.

To illustrate these conventions consider the axiom schema

- (24) $K(p, id(B), s) \supset (cl\ B)$, where B is any formula.

This says that whatever anyone knows is true. By taking B as " $\exists x >(x,y)$ " or " $\exists s1\ \text{red}(\text{block1}, s1)$ ", one gets two of the instances of this schema:

- (25) $K(p, \exists^*(x^* > (x^*, y^*)), s) \supset \forall y \exists x >(x,y)$
- (26) $K(p, \exists^*(s1^* \text{red}^*(\text{block1}^*, s1^*)), s) \supset \exists s1\ \text{red}(\text{block1}, s1)$

Some examples of the axioms and axiom schemas AX of the theory are

- (a1) $\forall U \forall V p s [K(p, >^*(U, >^*(V, U)), s)]$
- (a2) $\forall V W [K(p, V, s) \wedge K(p, >^*(V, W), s) \supset K(p, W, s)]$
- (a3) $\forall X V [K(p, V, s) \supset K(p, V^*(X, V), s)]$
- (a4) $\text{denote}(id(B), B)$, for every closed determinate string B
- (a5) $\forall U x y [\text{denote}(U, x) \wedge \text{denote}(U, y) \supset (x = y)]$

Axiom a1 states that in all states all people know that the string $(S1 \supset (S2 \supset S1))$ is true for all formulas S1 and S2. Axiom a2 states that in any state if someone knows S1 and $(S1 \supset S2)$, then in that state they know S2, where S1 and S2 are any formulas.

Axiom schema a4 states that every closed determinate string denotes its value, while axiom a5 says that a string can denote at most one thing.

3. DETERMINATE STRINGS

We define *determinate* string inductively as follows. A single variable or constant is determinate. The string $F(x1, \dots, xn)$ is determinate for any variables $x1, \dots, xn$ and any function (or relation) symbol *F* other than "K" or "denote". If $x1, \dots, xn$ are some of the free variables in a determinate string $B(x1, \dots, xn)$, and if $B1, \dots, Bn$ are determinate strings, then $B(B1, \dots, Bn)$ is determinate. If B is determinate, then so are $(V x B)$ and $(\exists x B)$. Finally, if B is determinate, and if *u, p, and s* are variables (of the appropriate sort), then $(\text{denote}(id(B), u))$ and $(K(p, id(B), s))$ are determinate.

Notice that any string that does not contain the symbols "K" or "denote" is determinate (the string may contain starred symbols, including K^* and denote^* , since these are distinct from "K" and "denote"). In general, a string is determinate if every first argument of a "denote" in the string, as well as every second argument of a "K" in the string, is an *id* of some determinate string. Some examples of determinate strings are

- (e1) $\forall x (x > y)$
- (e2) Ted
- (e3) $\text{denote}(\text{Ted}^*, x) \wedge K(p, >^*(x^*, y^*), s)$
- (e4) $\exists p K(p, \text{denote}^*(\text{Ted}^*, x^*), s)$
- (e5) $\text{denote}(\text{denote}^*(\text{Ted}^*, x^*), \text{denote}(W, y))$

Some examples of strings that are not determinate are

- (e6) $K(p, >^*(X, y^*), s)$
- (e7) $\text{denote}(V, x)$
- (e8) $\text{denote}(\text{denote}^*(V^*, x^*), u)$

4. INCONSISTENCY

The results in this section show how inconsistency considerations restrict both the "denote" and the "K" relation. These results are fairly general because only the few conditions explicitly stated in this section will be used: the axioms in the other sections are *not* assumed. In the following, $\text{Sub}(u,v,w)$ and $\text{Tr}(u)$ are formulas. For reference purposes, we list,

- (sb1) $\forall u \forall v \forall w_1 \forall w_2 (\text{Sub}(u,v,w_1) \wedge \text{Sub}(u,v,w_2) \supset w_1 = w_2)$
- (sb2) $\text{Sub}(\text{id}(B), \text{id}(C(v)), \text{id}(C(\text{id}(B))))$, for all strings B and C
- (tr1) $\text{Tr}(\text{id}(B)) \equiv (B \equiv \text{TRUE})$, for all closed formulas B
- (dn1) $\text{denote}(\text{id}(B), B)$, for all closed formulas B
- (k1) $\forall p \forall s [K(p, \text{id}(B), s) \supset c(B)]$, for all formulas B
- (k2) $c(B) \supset \exists p \forall s [K(p, \text{id}(B), s)]$, for every formula B .

One may think of sb1 and sb2 as saying that the formula $\text{Sub}(u,v,w)$ expresses substitution; of tr1 as saying that the formula $\text{Tr}(u)$ expresses truth; of k1 as saying that whatever anyone knows is true; of k2 as saying that whatever is true is known in at least one state by at least one person; of dn1 as being an unrestricted version of the "denote" relation. (In the following, when we say that one of these schemas is provable or is true, we mean that every instance of it is provable or is true.)

Theorem. Let A be a set of formulas. Suppose that there is a formula $\text{Sub}(u,v,w)$ such that sb1 and sb2 are provable from A . Furthermore, suppose either that dn1 is provable from A or that there is a formula $\text{Tr}(u)$ such that tr1 is provable from A . Then A is inconsistent.

This is Tarski's theorem (see for instance pg 85 of [1]) converted to a sort system. One might rephrase this theorem by saying that if substitution is definable in A , then one can allow neither the unrestricted version dn1 of the denotation relation (cf. axiom a4) nor an unrestricted version tr1 of the truth relation.

Inconsistency results also place restrictions on the relation of knowing.

Theorem. Let A be a set of formulas. Suppose that $\text{Sub}(u,v,w)$ is a formula such that sb1 and sb2 are provable from A . Suppose also that k1 and k2 are provable from A . Then A is inconsistent.

This theorem has an interesting formulation in terms of models and never known truths.

Theorem. Let $\text{Sub}(u,v,w)$ be a formula and let M be a model of sb1, sb2, and k1 (i.e., sb1, sb2, and k1 are true of M). Then in M , there is a "never known truth", which is to say, there is a formula C such that $(c(C))$ is true of M and such that $(\forall p \forall s \neg K(p, \text{id}(C), s))$ is true of M . In other words, if in M substitution is expressible, and if in M whatever anyone knows is true, then there is a truth that no one in any state ever knows.

There are variations and generalizations of these four theorems. Instead of using substitution as given in sb1 and sb2, one can use Peano axioms together with a certain relation between strings. There are likely many such variations. The last theorem can be generalized in a further direction: the set of never known truths in M is infinite in number, and cannot be "delimited in certain ways".

5. CONSISTENCY

On account of the previous inconsistency results, the consideration of consistency is an important one. Let S be a set of formulas that satisfies either c1 or c2.

(c1) S is consistent, and no part of any formula in S refers to an idea, which is to say, S speaks only of objects that are not ideas.

(c2) No formula in S contains the symbols "denote" or "K" (though they may contain such symbols as "denote*" and "K*"), and there is a model of S (i.e., a structure in which all the formulas of S are true) such that the ideas in the model are the strings of L , and such that the connection between the starred symbols and the model is as illustrated by the explanations of examples 1-15.

Theorem. The union of S together with the axioms AX of the theory described earlier is consistent.

For example, the axioms AX together with the Peano axioms and axioms expressing a variety of string substitutions and relations is consistent.

6. CONCLUSION

The axiom system AX was motivated by the desire to have a powerful set of axioms so that the consistency results would be stronger; no overall attention was paid to efficiency of proofs [6].

The results presented here are fairly general. The inconsistency results depend on few assumptions, and those could be modified in many ways. The proof of the consistency theorem is more complicated. A model of the axioms is created in an infinite sequence of stages, with "denote" and "K" appropriately extended at each stage by what has a fixed value (is "determinate") no matter how the rest of the stages are completed. This method seems general, going beyond the particular theory above or its formulation.

REFERENCES

- [1] Lyndon, Roger C. (1966) *Notes on Logic*, Van Nostrand, New York.
- [2] McCarthy, J. and Hayes, P. J. (1969) Some Philosophical Problems from the Standpoint of Artificial Intelligence, in B. Meltzer and D. Michie (eds.) *Machine Intelligence 4*, 463-502, Edinburgh: Edinburgh University Press.
- [3] McCarthy, J. (1977) Epistemological Problems of Artificial Intelligence, 1977 *IJCAI Proceedings*.
- [4] McCarthy, J. (1979) *First Order Theories of Individual Concepts and Propositions*, Stanford AI Lab, AIM-325.
- [5] Moore, Robert C. (1977) Reasoning about Knowledge and Action, 1977 *IJCAI Proceedings*.
- [6] Moore, Robert C. (1979) *Reasoning about Knowledge and Action*, Ph.D. Dissertation, Comp.Sci.Dept., MIT.