EFFICIENT THEOREM-PROVING IN SET THEORY

by

JUSSI KETONEN

Department of Computer Science and
Institute for Mathematical Studies in the Social Sciences
Stanford University

1. THE FOL PROOF-CHECKING PROGRAM

MY INTENT is to suggest a system of set theory that is particularly amenable to mechanical proof-checking. It is based on John McCarthy's concept of a "heavy-duty set theory" (McCarthy, 1980).

I have tested it using the FOL (First Order Logic) system at the Stanford Artificial Intelligence Laboratory. As an application, I was able to prove Ramsey's well-known theorem on partitions of unordered pairs on infinite sets in approximately 100 lines. Ramsey's theorem has been used as a benchmark in mechanical proof-checking; for example, Juan Bulnes (1979) gave a 400-line proof of it using the pure FOL system.

There is nothing new in any of the ideas presented here. The differences, if any, lie in our view of what a good, mathematically oriented proof-checker should look like: Such a program should be viewed as an expert symbol manipulator that can keep track of the correctness of inferences. The emphasis is on the ease of formalization—a factor that makes the program particularly amenable to mathematics as a discipline that thrives on highly abbreviated symbolic manipulations.

1.1 Application of a Proof-checker to Logic

The role of logic is deemphasized in a mathematically oriented proof-checker. However, one should be able to do formal logic on it just as one would do any other kind of mathematics. The use of logic in a proof-checker of this type is a relatively straightforward matter requiring a procedure to check the correctness of simple inferences. For this purpose a program has been written that tests the validity of deductions in a fragment of predicate

The research reported in this article was partially supported by National Science Foundation Grant SED 77-09698 to Stanford University.

calculus designed to capture the notion of trivial inferences (Ketonen, 1981). This procedure should be adequate for its intended purposes.

1.2 *Components of an Efficient Proof-checker*

The capacity to do efficient symbolic manipulations has been notably lacking in most proof-checking systems. The tendency has been to expect a heuristically guided resolution theorem prover to handle the enormously complicated formulas that result after all definitions have unwound. A few capabilities that are needed for a good symbolic processor are listed below.

First, a powerful syntactic simplifier is needed, that is, a facility for rewriting formulas. This has been a part of FOL for a long time and has in fact turned out to be one of its most frequently used capabilities. Second, the capability to talk about formal functional application and to create lambda-abstractions of terms is needed. I have implemented this by introducing a 2-ary function APPLY(fx); obviously, in a system of the future one should be able to use the more mathematical $f(x)$ instead. Finally, the capability to formulate axioms so as to facilitate efficient symbol manipulation is needed. Here, again, the simple imitation of actual mathematical practice proves to be a wise choice. For example, to define a function as a set of ordered pairs is certainly correct but atrocious from the point of view of efficiency and clarity in notation. Similarly, one could define the pairing axiom as

$$\forall x\, y.\ \exists z.\ \forall u.(u \in z \equiv (u = x) \vee (u = y)),$$

which, though logically correct, clearly loses to

$$\forall x\, y\, u.(u \in \text{UNORDPAIR}(xy) \equiv (u = x) \vee (u = y))$$

in expressive power. This again emphasizes the highly abbreviated nature of mathematical reasoning and its efficient use of functional application.

The simpleminded suggestions given above were sufficient for me to achieve a fourfold reduction in the size of the proof of Ramsey's theorem from the previous known proof in FOL (Bulnes, 1979). However, we are still far from a mathematical proof-checker that can produce clear and easily comprehensible proofs.

2. AXIOMATIZATION OF SET THEORY

My axiomatization of set theory is fairly close to the standard Zermelo-Fraenkel theory with the addition of the binary operation APPLY (fx) representing functional application. Most of the axioms are phrased so as to be usable as rewriting rules. They are expressed in FOL-like notation. Formally, before we can state our axioms in FOL, we would have to declare the

variables, predicate, and function symbols to be used. For example, we declare a unary predicate symbol SET to represent the sort of sets and declare variables s, t, and u to be of this sort. The letters x, y, and z will then represent class variables. For more discussion of FOL, see Weyrauch (1977).

AXIOM NULL: $\forall x.(x \in 0 \equiv \text{FALSE})$;;
AXIOM EXTENSIONALITY: $\forall x\, y.((x = y) \equiv (\forall z.(z \in x \equiv z \in y)))$;;
AXIOM PAIRS: $\forall x\, y\, z.(z \in \text{UNORDPR}\ (x\, y) \equiv ((z = x) \vee (z = y)))$;;
AXIOM SINGLETONS: $\forall x\, y.(y \in \text{SINGLETON}(x) \equiv (y = x))$;;
AXIOM UNION: $\forall x\, y.(x \in \text{UNION}(y) \equiv \exists z.((x \in z) \wedge (z \in y)))$;;
AXIOM POWERSET: $\forall x\, y.(x \in \text{POWERSET}(y) \equiv (x \subset y))$;;
AXIOM INCLUSION: $\forall x\, y.(x \subset y \equiv \forall z.(z \in x \supset z \in y))$;;
AXIOM PAIRUNION: $\forall x\, y\, z.(z \in (x \cup y) \equiv ((z \in x) \vee (z \in y)))$;;
AXIOM INTERSECTION: $\forall x\, y\, z.(z \in (x \cap y) \equiv ((z \in x) \wedge (z \in y)))$;;
AXIOM DIFFERENCE: $\forall x\, y\, z.(z \in (x - y) \equiv ((z \in x) \wedge \neg(z \in y)))$;;
AXIOM ORDPAIRS: $\forall x\, y\, z\, u.((\text{PAIR}\ (x\, y) = \text{PAIR}\ (z,u)) \equiv ((x = z(\wedge\ (y = u)))$;;
AXIOM PROJECTION: $\forall x\, y.\ (\text{PROJ1}\ (\text{PAIR}\ (x\, y)) = x)$, $\forall x\, y.(\text{PROJ2}\ (\text{PAIR}\ (x\, y)) = y)$;;
AXIOM COMPREHENSION: $\forall s.(x \in \{y \mid P(y)\} \equiv P(s))$;;

The axioms stated above make up the most heavily used part of our theory. We will have a special simplification set, SETS, consisting of the corresponding rewrite rules.

AXIOM CHOICE: $\forall x.((\forall z.(z \in x \supset (\exists u.u \in z))) \supset \exists f.\ \forall z.(z \in x \supset (\text{APPLY}(f\, z) \in z)))$;;
AXIOM FOUNDATION: $\forall x.(x = 0 \vee \exists y.(\forall z.(z \in x \supset \neg(z \in y))))$;;

Finally, we have axioms for set formation:

AXIOM FORM: $\forall s\ t.\text{SET}\ (s \cup t)$, $\forall s\ t.\text{SET}\ (s \cap t)$
AXIOM FORM: $\forall s\ t.\text{SET}\ (s \cup t)$, $\forall s\ t.\text{SET}\ (s \cap t)$, $\forall s\ t.\text{SET}\ (s - t)$, $\forall s\ t.\text{SET}\ (\text{UNORDPR}\ (s\, t))$, $\forall s\ t.\text{SET}\ (\text{PAIR}\ (s\, t))$, $\forall s.\text{SET}\ (\text{UNION}\ (s))$, $\forall s.\text{SET}\ (\text{POWERSET}\ (s))$, $\forall s.\text{SET}\ (\text{PROJ1}\ (s))$, $\forall s.\text{SET}\ (\text{PROJ2}\ (s))$, $\forall s.\text{SET}\ (\{y \mid P(y) \wedge y \in s\})$,

$$\forall s\, t.\text{SET}\ (\{y \mid \exists z.\ (z \in s \wedge \text{APPLY}\ (t\ z) = y)\});;$$

AXIOM SETS: $\forall s\, x.(x \in s \supset \text{SET}(x));;$

The theory of natural numbers is included by considering the sort NATNUM as less general than the sort SET, and by stating the usual Peano axioms for them with the addition of suitable axioms for the order relation.

2.1 *Implementation of FOL to Lambda-abstractions*

The most interesting part of the FOL program consists of the use of the metatheoretic capabilities of FOL to formulate the principles of lambda-abstraction and inductive definition. This part of FOL is documented in Weyrauch (1978).

To implement the general schema of lambda-abstraction, one proceeds in FOL as follows:

DECLARE OPCONST mklambda(TERM, INDVAR) = WFF;
DECLARE INDVAR term initerm ∈ TERM;
DECLARE INDVAR indvar ∈ INDVAR;
AXIOM LAMBDA: ∀term indvar. THEOREM(mklambda(term indvar));;
ATTACH mklambda to MKLAMBDA;

MKLAMBDA is a suitable function in the LISP environment of FOL. The command REFLECT LAMBDA ⟨term⟩, ⟨variable to be bound⟩; will then yield a new line of the form

$$\exists f.\ \forall\langle\text{variable}\rangle.(\text{APPLY}\ (f\ \langle\text{variable}\rangle) = \langle\text{term}\rangle)$$

without any dependencies where f is a new variable not appearing in term provided that ⟨variable⟩ is of a sort contained in SET.

It should be mentioned that the notion of domain is not of particular interest here. While one could talk explicitly about domains of functions, it is far more efficient and elegant to let the matter arise implicitly—functions are simply left undefined outside the current domain of interest.

The principle of inductive definition is implemented in a similar fashion.

2.2 *The Proof of Ramsey's Theorem*

This brief sketch of the proof of Ramsey's theorem is phrased in terms of sets unbounded in the set of all natural numbers. The letters *i, j, k, l, n,* and *m* have been declared to be of the sort NATNUM.

DECLARE PREDCONST UNB 1;
AXIOM UNB: $\forall x.(\text{UNB}(x) \equiv (\forall i.\ \exists j.(i < j \vee j \in x)));;$

The following elementary facts about unbounded sets are needed for the proof; their proofs are short and straightforward.

AXIOM FACT: UNB(N)

$$\forall x\, y\, z.(\text{UNB}(x) \wedge (x \subset y \cup z) \supset (\text{UNB}(y) \vee \text{UNB}(z))),$$
$$\forall x\, y.(\text{UNB}(x) \wedge (x \subset y) \supset \text{UNB}(y)),$$
$$\forall x\, f.(\text{UNB}(x) \wedge \forall i(\text{APPLY}(f\, i) = 0 \vee \text{APPLY}(f\, i) = 1) \supset (\text{UNB}(\{i \mid i \in x \wedge \text{APPLY}\ (f\, i) = 0\}) \vee \text{UNB}(\{i \mid i \in x \wedge \text{APPLY}\ (f\, i) = 1\})));;$$

The following lemmas are also needed; RLEM is crucial to the proof.

AXIOM TRANS: $\forall k.Q\,(k\, k+1) \wedge \forall k\, 1\, n.\ (Q\,(k\, 1) \wedge Q\,(1\, n) \supset Q(k\, n)) \supset \forall k\, 1.(k < 1 \supset Q\,(k\, 1));;$

AXIOM RLEM: $\forall f\, g.(\forall n.(\text{UNB}(\text{APPLY}(f\, n)) \wedge \text{APPLY}\ (f\, n+1) \subset \text{APPLY}\ (f\, n)) \wedge \forall i.(\text{APPLY}(g\, i) = 1 \vee \text{APPLY}(g\, i) = 0) \supset \exists k.((k = 0 \vee k = 1) \wedge \forall n.\text{UNB}(\{i \mid i \in \text{APPLY}(f\, n) \wedge \text{APPLY}(g\, i) = k\})));;$

Ramsey's theorem is proved in the following form:

$$\forall f.(\forall i\, j.\text{APPLY}(f,\text{UNORDPR}(i,j)) \in \text{UNORDPR}(x,1) \supset \exists l\, b.\ \forall m\, n.(n < m \supset (\text{APPLY}(f,\text{UNORDPR}(\text{APPLY}(b,n),\text{APPLY}(b,m))) = 1 \wedge \text{APPLY}(b,n) < \text{APPLY}(b,m)))).$$

REFERENCES

Bulnes, J. *GOAL: A goal-oriented command language for interactive proof construction* (Memo AIM–328). Stanford, Calif.: Stanford University, Stanford Artificial Intelligence Laboratory, Stanford University, 1979.

Ketonen, J. On a decidable fragment of predicate calculus. In P. Suppes (Ed.), *University-level computer-assisted instruction at Stanford: 1968–1980*. Stanford, Calif.: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1981.

McCarthy, J. Personal communication, 1980.

Weyrauch, R. *A user's manual of FOL* (Memo AIM–235). Stanford, Calif.: Stanford University, Stanford Artificial Intelligence Laboratory, 1977.

Weyrauch, R. *Prolegomena to a theory of formal reasoning* (Memo AIM–315). Stanford, Calif.: Stanford University, Stanford Artificial Intelligence Laboratory, 1978.