DAG PRAWITZ

A NOTE ON KANGER'S WORK ON EFFICIENT PROOF PROCEDURES

Three of Stig Kanger's works belong to proof theory taken in a wide sense: the monograph *Provability in Logic* of 1957, the paper "A simplified proof method for elementary logic" of 1963, and, in between these two, the mimeographed *Handbok i logik* written in 1959. I concur in Göran Sundholm's remark in his paper of the present volume that Kanger's main interest in this connection was not proofs themselves but provability and derivability and in particular the relation of these notions to semantical ones. A case in point is Kanger's variant of Gentzen's calculus of sequents for classical logic, LK, which Kanger develops in *Provability in Logic*. The purpose is there to give a new demonstration of Gödel's completeness result that every valid formula is provable, i.e. has *some* proof, no matter which.

This picture of Kanger's proof theoretical interests is in need of some supplementations and qualifications, however. Kanger also gave a proof of Gentzen's Hauptsatz, a corner-stone in proof theory, which takes up most of Gentzen's classical paper. The theorem is now obtained as an easy corollary of Kanger's completeness result for a cut-free version of LK: If the sequents $\Gamma \Rightarrow \Delta, A$ and $A, \Gamma \Rightarrow \Delta$ are both provable, then in view of the soundness of the calculus they are valid and so is the sequent $\Gamma \Rightarrow \Delta$ (by the semantical validity of the cut rule), and hence by the completeness theorem, the sequent $\Gamma \Rightarrow \Delta$ is provable without use of the cut rule.

Kanger was very fond of his semantical proof of the Hauptsatz and in particular of the ease with which he obtained it. He devotes a section of eight lines to it in his otherwise very condensed monograph. Modestly he remarks that Gentzen's proof is superior to his own since it is finitary. But Kanger did not really care about a result being established in a finitary way. Therefore, his real attitude, as I remember it, was that the Hauptsatz, having been obtained for free, could hardly be a deep theorem.

This indifference to the significance of the Hauptsatz seems to confirm the initial impression that proofs themselves on the object level and their properties were not the kind of things that interested Kanger. But this is not the whole truth. Kanger had an interest in the art of engineering, in how things are made, and he saw that his way of establishing the completeness result for a

G. Holmström-Hintikka, S. Lindström and R. Slivinski (eds.),
Collected Papers of Stig Kanger with Essays on his Life and Work, Vol. II, 43–52.

cut-free formalism gave rise to a specific proof procedure. Although proof theory did not please his philosophical interest, model theory being his favourite, more practical questions concerning how to find proofs in an efficient way appealed to him.

Kanger's monograph *Provability in Logic* was his doctoral dissertation, presented in theoretical philosophy in Stockholm in the academic year 1956–57, at which time I was a beginner in philosophy. When my teacher Anders Wedberg, who had also been the teacher of Kanger, was to describe the content of Kanger's thesis to us beginners, he described it as essentially amounting to a new proof procedure for predicate logic, which in principle could be implemented on a computer.[1]

This stimulated me to try to automatize Kanger's procedure, and this in turn led Stig and me to some new insights in that field. I shall here give some glimpses of this early phase of automatic deduction.

1. KANGER'S FIRST PROOF PROCEDURE

The proof procedure that Kanger developed in *Provability in Logic*, also described in Sundholm's paper in this volume, consists essentially in applying the rules of a cut-free version of Gentzen's calculus of sequents backwards until one reaches an axiom at the top of each branch of the resulting tree. What did such a proof procedure amount to?

We may separate the sentential and the quantification part of the procedure. As for the first part, it is instructive to make a comparison. Every beginning student of logic learns after a while that to verify that a formula in sentential logic is a tautology one does not need to go through all the possible truth value assignments to the atomic formulas. Given for instance the formula

$$[A \to (B \to C)] \to [(A \,\&\, B) \to C]$$

it is sufficient to reason as follows: A falsifying assignment must make the antecedent true and the succedent $(A \,\&\, B) \to C$ false; to achieve the latter A and B must be assigned truth and C falsity; but to achieve the first, i.e. to make the antecedent $A \to (B \to C)$ true, either A must be assigned falsity, which possibility has already been excluded, or $B \to C$ must get the value truth, requiring in turn that either B is assigned falsity or that C is assigned truth, which two possibilities have also been excluded. Hence there is no falsifying assignment.

This way of reasoning has exactly the same structure as applying the sequent rules as formulated by Kanger backwards: to make a formula true corresponds to putting it in the antecedent of the sequent, i.e. before the arrow,

and to make it false to putting it in the succedent of the sequent, i.e. after the arrow; that a formula cannot be both true and false corresponds to the fact that a sequent in which a formula occurs both before and after the arrow is an axiom. To use sequent rules in this way thus seemed to be a very appropriate way of finding proofs in sentential logic corresponding to known short cuts in the handling of truth tables.

As for the quantificational part, backward applications of the quantification rules amounted to the generation of instances of quantified formulas. In the case of an existential formula in the antecedent or a universal formula in the succedent a new constant was introduced to replace the quantified variable (which corresponds semantically to introducing in the counter model sought for the name of an individual that satisfies an instance of the existential formula or falsifies an instance of the universal formula, respectively). In the case of a universal formula in the antecedent or an existential formula in the succedent, instances were instead formed by systematically substituting for the quantified variable all constants that had been introduced in operations of the first kind.

The method was quite straightforward and not very sophisticated. But it was a complete proof search that took advantage of the sub-formula property of cut-free proofs, and the achievement was to formulate it precisely. After Kanger's public defence of his dissertation in the spring of 1957, I sat down in the summer to automatize the procedure. I was then also inspired by Beth's semantical tableaux used in his proof of the completeness result, which parallels Kanger's method in many respects.

At that time there was a computer in Stockholm known as BESK. It was a huge machine occupying several rooms of what had been the premises of the Royal Technical University. Just then it had the record as the fastest computer of the world. But this was a time when there were no programming languages. One had to use the machine code, which I did not know and did not want to learn. Instead I invented a programming language of my own suitable for the particular task in question, and it was translated to the machine code by my father, who had sometimes been using BESK for certain mathematical calculations. The program was run on the machine by Neri Voghera in 1958, and the whole project was presented at the First International Conference on Information Processing, which was arranged in Paris in 1959 by UNESCO. It was published in 1960[2] and was one of the very first automated proof procedures. Some roughly equivalent procedures were implemented on other computers and presented in journals at more or less the same time.

Our program proved simple theorems of logical textbooks, e.g. the one saying that a transitive and irreflexive relation is asymmetric was proved in 12

seconds. But as soon as it came to more advanced theorems, it was hopelessly inadequate, in spite of the record speed of the computer.

2. THE DUMMY METHOD

The reason for the inefficiency of the method was very obvious. After having introduced a few constants because of existential formulas occurring in the antecedent or universal formulas occurring in the succedent, the number of possible substitutions becomes very large when one is to apply the other two quantification rules backwards to universal formulas in the antecedent or existential formulas in the succedent .

As an illustration suppose that we have a sequent of the form

$$\forall x_1 \forall x_3 \forall x_2 \forall x_4 A(x_1,x_2,x_3,x_4) \Rightarrow \forall x_1 \forall x_2 \forall x_3 \forall x_4 A(x_1,x_2,x_3,x_4)$$

where the antecedent and succedent are the same except for a permutation of the quantifiers. A human recognizes the validity of the sequent as soon as she sees that the only difference between the two formulas is the permutation of $\forall x_2 \forall x_3$. However, to prove the sequent by our procedure, one has to try out different ways of breaking down the antecedent formula by substituting the four constants c_i generated by the succedent formula , i.e., after having first obtained the sequent

$$\forall x_1 \forall x_3 \forall x_2 \forall x_4 A(x_1,x_2,x_3,x_4) \Rightarrow A(c_1,c_2,c_3,c_4),$$

one has to test at random substitutions of c_1, c_2, c_3, and c_4 for the four universally quantified variables in the formula in the antecedent. There are thus $4^4 = 256$ possible formulas of the form $A(t_1,t_2,t_3,t_4)$ to generate, and we may assume that only one of them, $A(c_1,c_2,c_3,c_4)$, yields a proof of the original sequent. There was no method for seeking out the relevant substitution instance, and hence one could expect that, on an average, one had to generate a sequent containing 128 substitution instances in the antecedent — quite a long sequent in view of the simplicity of the original sequent. Imagine a similar example but with 10 quantifiers instead of 4 — then there are 10^{10} possible formulas of the form $A(t_1,t_2,...,t_{10})$ to generate, and the right one will be found after 5 000 000 000 steps on an average! Clearly the speed of the computer did not matter much.

At a seminar where my automatization of Kanger's and Beth's proof procedures was presented, I draw attention to this inefficiency of the method and suggested that one should try to find the right substitutions by some kind of calculation. I compared our method to solving equations, say $8x + 37 = 445$, by systematically trying different values for x until the right one was found. Just

as we find the roots of equations by performing certain calculations and not by running through possible values 1, 2, 3, ..., we should find the right substitution instance directly by some kind of calculation. At this seminar Kanger suggested as a solution simply to replace the variable by a dummy, which could also be called a meta-variable, and to let it stand while continuing as before until one sees what substitutions are appropriate to make for the dummy. He provokingly pronounced that this dummy-method solved the problem.

I remember that I was slightly irritated by this proposal. Just to replace the variables by something called dummies did not solve the problem; in the essay discussed at the seminar I had also suggested that one should drop quantifiers, replace the variables by meta-variables ranging over the constants to be substituted later, and go on with applying the sentential operations to the formulas — the problem was to "see", as Kanger expressed it, what values should be assigned to the dummies, or how to "put together different proof branches to a proof tree" as I had expressed it. Nevertheless, it did turn out that to replace the variables by dummies or to treat them as meta-variables was a very good strategy, or even *the* right strategy. The crucial idea was to postpone substitutions for the variables in question until there was a better opportunity for making a suitable choice. When attention was focused on this idea, it was not difficult to point out situations in which it was fairly easy to see what the appropriate substitutions should be. Somewhat later it turned out that one could make different selections of such situations.

Kanger described his idea in his *Handbok i logik* and again in the paper "A simplified proof method for elementary logic". This paper is note-worthy also for other features to be mentioned below. The dummy method is described as follows. New constants are introduced by backward applications of quantification rules as before. At applications of the other quantification rules (that do not generate new constants), we substitute not a constant but instead a new dummy d and list at the same time what values the dummies can assume, i.e. we make a note $d/t_1, t_2, .., t_n$, called a substitution list, containing all constants and dummies that occur in the sequent to which the rule is applied. At some stages we stop and "check whether we can choose values for the dummies from the substitution lists in such a way that all top sequents will be directly demonstrable when we replace the dummies by their values". "Directly demonstrable" means here to be either an axiom or derivable from an axiom by inference rules for identity.

Applied to the sequent occurring in the example above, the method works as follows: The formula in the succedent is replaced by $A(c_1,c_2,c_3,c_4)$, where c_1, c_2, c_3, and c_4 are four new constants, while the variables of the formula in the

http://www.springer.com/978-1-4020-0111-6

Collected Papers of Stig Kanger with Essays on his Life and Work Volume II

Holmström-Hintikka, G.; Lindström, S.; Sliwinski, R. (Eds.)

2001, XII, 281 p., Hardcover
ISBN: 978-1-4020-0111-6