STUDIES IN LOGIC, GRAMMAR AND RHETORIC 9 (22) 2006

Roman Murawski
Adam Mickiewicz University

THE PRESENT STATE OF MECHANIZED DEDUCTION, AND THE PRESENT KNOWLEDGE OF ITS LIMITATIONS*

1. Introduction

In 1936 Alan Turing and Alonzo Church proved two theorems which seemed to have destroyed all hopes of establishing a method of mechanizing reasonings. Turing in (1936–37) reduced the decidability problem for theories to the halting problem for abstract machines modelling the computability processes (and named after him) and proved that the latter is undecidable. Church (1936) solving Hilbert's original problem proved the undecidability of the full predicate logic and of various subclasses of it.

On the other hand results of Skolem and Herbrand (cf. Chapter 6 of Marciszewski and Murawski, 1995) showed that if a theorem is true then this fact can be proved in a finite number of steps – but this is not the case if the theorem is not true (in this situation either one can prove in some cases the falsity of the given statement or the verification procedure does not halt). This semidecidability of the predicate logic was the source of hope and the basis of further searches for the mechanized deduction systems. Those studies were heavily stimulated by the appearence of computers in early fifties. There appeared the idea of applying them to the automatization of logic by using the mechanization procedures developed earlier. The appearance of computers stimulated also the search for new, more effecitive procedures.

In the sequel we shall describe the history of those researches. In Section 2 the early attempts of applying computers to prove theorems will be

* The financial support of the Committee for Scientific Research (grant no 1 H01A 041 27) is acknowledge.

ISBN 978–83–7431–108–3 ISSN 0860-150X

presented, in particular we shall tell about results of Davis, Newell-Shaw-Simon, Gilmore, Gelernter *et al.*, Hao Wang and Davis-Putnam. Section 3 will be devoted to resolution and unification algorithms of Prawitz and Robinson and to their modifications. They turned out to be crucial for the further development of the researches towards mechanization and automatization of reasonings. We will sketch them in Section 4.

All those studies led to the idea of automated theorem proving by which one means the use of a computer to prove non-numerical results, i.e., to determine their truth (validity). One can demand here either a simple statement "proved" (what is the case in decision procedures) or human readable proofs. We can distinguish also two modes of operation: (1) fully automated proof searches and (2) man-machine interaction proof searches.

Note that the studies of mechanization of reasonings and automated theorem proving were motivated by two different philosophies. The first one – which we shall call the logic approach – can be characterized "by the presence of a dominant logical system that is carefully delineated and essentially static over the development stage of the theorem proving system. Also, search control is clearly separable from the logic and can be regarded as sitting 'over' the logic. Control 'Heuristics' exist but are syntax-directed primarily" (cf. Loveland 1984, p. 3). The second philosophical viewpoint is called the human simulation approach. It is generally the antithesis of the first one. It can be characterized shortly by saying that "the thrust is obviously simulation of human problem solving techniques" (cf. Loveland 1984, p. 3). Of course the logic and human simulation approaches are not always clearly delineated. Nevertheless this distinction will help us to order the results and to consider the influence of each of the approaches on various particular systems.

In the last Section 5 we indicate some limitations of mechanized deduction and automated theorem proving. In particular we distinguish effective and feasible computability/decidability/deducibility and show on some examples the complexity of decision procedures for certain theories. Connections with the famous problem $P =?NP$ are also indicated (and some information on the origins of this problem is presented). On the other hand we discuss the speed-up phenomenon that appears when moving from the logic of the level n to the logic of the level $n + 1$ and – following Boolos – give a simple example of a formula which can be proved (in the usual sense) in the second-order logic but which has no proof that could be written down in the first-order logic.

2. First mechanized deduction systems

The different philosophical backgrounds mentioned above can be spotted already in the first studies towards mechanization of reasonings and automated theorem proving. In 1954 Martin Davis wrote a program to prove theorems in additive arithmetic (this program was never published in a paper). It was developed on the computer "Johniac" in the Institute for Advanced Study and was a straight implementation of the classical Presburger's decision procedure for additive number theory (i.e. for the theory of non-negative integers in the language with zero, successor and addition only) (cf. Presburger 1929). The complexity of this decision procedure is very high and therefore the program proved only very simple facts (e.g. that the sum of two even numbers is also an even number – this was the first mathematical theorem in the history proved by a computer!).

The Presburger prover of M. Davis was an example of the logic approach. The second achievement in the field of automated theorem proving called the logic theorist should be included among examples of the human simulation approach. We mean here the program of A. Newell, J. S. Shaw and H. A. Simon presented in 1956 at the Dartmouth conference (cf. Newell *et al.* 1956). This program could prove some theorems in the propositional calculus of *Principia Mathematica* of A. N. Whitehead and B. Russell. Its goal was to mechanically simulate the deduction processes of humans as they prove theorems of the sentential calculus. Two methods were used: (1) substitution into established formulas to directly obtain a desired result and, failing that, (2) to find a subproblem whose proof represents progress towards proving the goal problem. This program was able to prove 38 theorems of *Principia*.

The Geometry Theorem-proving Machine of Gelernter and others from 1959 (cf. Gelernter *et al.* 1959, 1960) is also an example of the second approach. It applied the idea of M. Minsky that the diagram that traditionally accompanies plane geometry problems is a simple model for the theorem that could greatly prune the proof search. The program worked backwards, i.e., from the conclusion (goal) towards the premises creating new subgoals. The geometry model was used just to say which subgoals were true and enabled to drop the false ones. It should be noticed that this program was able to prove most high school exam problems within its domain and the running time was often comparable to high school student time.

Simultaneously with the Geometry Theorem-proving Machine two new efforts in the logic framework occurred. We mean here works of Gilmore and Hao Wang. They used methods derived from classical logic proof procedu-

res and in this way rejected opinions that logical methods cannot provide a useful basis for automated theorem proving. Such opinions were rather popular at that time. They were founded on the fact that logic oriented methods were inefficient and on the fact that the methods of Newell, Shaw, Simon and Gelernter proved to be successful. The method of P. C. Gilmore was based on Beth's semantic tableau technique. It was probably the first working mechanized proof procedure for the predicate logic – it proved some theorems of modest difficulty (cf. Gilmore 1959, 1960).

In the summer 1958 Hao Wang developed the first logic oriented program of automated theorem proving of IBM and continued this work at Bell Labs in 1959–63 (cf. Wang 1960, 1960a, 1961). Three programs were developed: (1) for propositional calculus, (2) for a decidable part of the predicate calculus and (3) for all of predicate calculus. Those programs were based on Gentzen-Herbrand methods, the last one proved about 350 theorems of *Principia Mathematica* (they were rather simple theorems of pure predicate calculus with identity).

During the Summer Institute for Symbolic Logic held at Cornell University, USA, in 1954 Abraham Robinson put forward, in his short lecture, the idea of considering the additional points, lines and circles – which must be used in a search for a solution of a geometrical problem – simply as elements of the so called Herbrand universe. This should enable us to abandon the geometrical constructs and to use directly Herbrand's methods.

This idea turned out to be very influential and significant. One of the first programs that realized it was implemented in 1960–62 by M. Davis and H. Putnam (cf. Davis–Putnam 1960). By Herbrand's theorem, the question of validity of a predicate calculus formula Z can be reduced to a series of validity questions about ever-expanding propositional formulas. More exactly one should consider so called Herbrand disjunctions $A_1 \vee \cdots \vee A_n$ (which can be effectively obtained from Z). It holds that Z is valid if and only if there exists an n such that the disjunction $A_1 \vee \cdots \vee A_n$ is valid. The formulas A_i are essentially substitition instances over an expanded term alphabet of Z with quantifiers removed. So one can test now $A_1 \vee \cdots \vee A_n$ for ever-increasing n for example by truth table and conclude that Z is valid if among formulas $A_1 \vee \cdots \vee A_n$ a tautology was found. But truth tables happen to be quite inefficient. The procedure of Davis and Putnam tried to overcome this difficulty. In fact they were considering unsatisfiability (instead of validity) of formulas and worked with conjunctive normal forms. Such a form is a conjunction of clauses, each clause being a disjunction of literals, i.e. atomic formulas (atoms) or their negations. The Davis–Putnam procedure can be described now as follows: "[it] made optimal use of simplification by can-

cellation due to one-literal clauses or because some literals might not have their complement (the same atomic formula but only one with a negation sign) in the formula. A simplified formula was split into two formulas so further simplification could recur anew" (cf. Loveland 1984).

3. Unification and resolution

The procedure of Davis and Putnam described in the last section had some defects. The main one was the enumeration substitutions – prior to this point substitutions were determined by some enumeration scheme that covered every possibility. Prawitz (1960) realized that the only important substitutions create complementary literals. He found substitutions by deriving a set of identity conditions (equations) that will lead to contradictory propositional formula if the conditions are met. In this way one got a procedure of substituting Herbrand terms. It is called today unification.

M. Davis developed right away the idea and combining it with the procedure of Davis–Putnam implemented it in a computer program based on a so called Linked-Conjunct method (cf. Davis 1963). This program used conjunctive normal forms of formulas and the unification algorithm developed by D. McIlroy in November 1962 at the Bell-Telephone-Laboratories. It was the first program which overcame the weaknesses of Herbrand procedure and improved it just by using conjunctive normal form and the unification algorithm (cf. Davis 1963 and Chinlund *et al.* 1964).

Simultaneously at Argonne National Lab near Chicago a group of scientists (G. Robinson, D. Carson, J. A. Robinson, L. Wos) was working on computer programs proving theorems. They used methods based on Herbrand theorem recognizing their inefficiency. Studying papers of Davis–Putnam and Prawitz they came to the idea of trying to find a general machine-oriented logical principle which would unify their ideas in a single rule of inference. Such a rule was found in 1963–64 by John Alan Robinson and published in (1965). It is called the resolution principle and is today one of the most fundamental ideas in the field of automated theorem proving. Therefore we shall describe it now more exactly.

The resolution principle is applied to formulas in a special form called conjunctive normal form. Given a formula A of the language of predicate calculus we first transform it into prenex normal form, i.e., to the form $(Q_1x_1)(Q_2x_2)\dots(Q_nx_n)B$ where Q_i $(i = 1, \dots, n)$ are universal or existential quantifiers and B is quantifier-free (B is called matrix). One can show that the prenex normal form of a formula A is logically equivalent to the

formula A. As an example consider the following formula (stating that the function f is continuous):

$$\forall\varepsilon\{\varepsilon > 0 \longrightarrow \exists\delta[\delta > 0 \wedge \forall x\forall y(|x - y| < \delta \longrightarrow |f(x) - f(y)| < \varepsilon)]\}.$$

Its prenex normal form is:

$$\forall\varepsilon\exists\delta\forall x\forall y[\varepsilon > 0 \longrightarrow \delta > 0 \wedge (|x - y| < \delta \longrightarrow |f(x) - f(y)| < \varepsilon)].$$

Observe that prenex normal form of a formula is not uniquely determined but on the other hand all prenex normal forms of a given formula are logically equivalent.

The next step of the transformation of formulas we need is skolemization (cf. Section 6.5 of Marciszewski and Murawski, 1995). Recall here only that it consists of dropping of all the quantifiers and of replacing every occurrence of each variable bound by an existential quantifier by a functional term containing the variables that in this formula are bound by universal quantifiers preceding the considered existential quantifier. A formula obtained in this way is said to be in a skolemized prenex normal form or in its functional form for satisfiability. The skolemized prenex normal form of the formula from our above example is the following:

$$\varepsilon > 0 \longrightarrow g(\varepsilon) > 0 \wedge (|x - y| < g(\varepsilon) \longrightarrow |f(x) - f(y)| < \varepsilon).$$

Note that the skolemized normal form of a formula is not equivalent to the given formula but it is satisfiable (inconsistent) if and only if the given formula is satisfiable (inconsistent).

The next step of getting the conjunctive normal form consists of the elimination of connectives $\longleftrightarrow$ and $\longrightarrow$ and moving all negation signs $\neg$ to the atoms. We proceed according to the following rules:

$$\begin{aligned}
A \longleftrightarrow B &\quad \text{is logically equivalent to} \quad (A \longrightarrow B) \wedge (B \longrightarrow A)\\
A \longrightarrow B &\quad \text{is logically equivalent to} \quad (\neg A \vee B)\\
\neg(A \wedge B) &\quad \text{is logically equivalent to} \quad (\neg A \vee \neg B)\\
\neg(A \vee B) &\quad \text{is logically equivalent to} \quad (\neg A \wedge \neg B)\\
\neg\neg A &\quad \text{is logically equivalent to} \quad A.
\end{aligned}$$

A formula obtained in such a way is said to be in the negation normal form. For example the negation normal form of the formula considered above is:

$$\neg\varepsilon > 0 \vee [g(\varepsilon) > 0 \wedge (\neg|x - y| < g(\varepsilon) \vee |f(x) - f(y)| < \varepsilon)].$$

Note that atoms and negated atoms are usually called literals.

The last step of the considered transformation consists of the application of the following distributivity laws:

$$A \vee (B \wedge C) \longleftrightarrow (A \vee B) \wedge (A \vee C),$$
$$(A \wedge B) \vee C \longleftrightarrow (A \vee C) \wedge (B \vee C).$$

In this way the obtained formula is of the form of conjunctions of disjunctions of literals. Such a form is called the conjunctive normal form and the disjunctions of literals are called clauses. Clauses consisting of a single literal are called unit clauses. Clauses with only one positive literal are called Horn clauses. They correspond to formulas of the form $A_1 \wedge \ldots \wedge A_n \longrightarrow B$. To illustrate the last step note that the conjunctive normal form of our formula stating that a function f is continuous is the following:

$$(\neg\varepsilon > 0) \vee [g(\varepsilon) > 0 \wedge \neg|x - y| < g(\varepsilon)] \vee [g(\varepsilon) > 0 \wedge |f(x) - f(y)| < \varepsilon].$$

Note that if $(A_1^1 \vee \cdots \vee A_{n_1}^1) \wedge \ldots \wedge (A_1^l \vee \cdots \vee A_{n_l}^l)$ is a conjunctive normal form of a formula A then we write it sometimes also as: $\{A_1^1 \vee \cdots \vee \vee A_{n_1}^1, \ldots, A_1^l \vee \cdots \vee A_{n_l}^l\}$ or as $\{\{A_1^1, \cdots, A_{n_1}^1\}, \ldots, \{A_1^l, \cdots, A_{n_l}^l\}\}$. This form is called the clause form. In this way we have shown that for any formula A of the language of predicate calculus there exists a formula A' in conjunctive normal form such that the formula A is satisfiable (inconsistent) if and only if the formula A' is satisfiable (inconsistent).

Having described the needed form of formulas we can introduce now the resolution principle. First observe that a formula B is a logical consequence of formulas $A_1, \ldots, A_n$ if and only if the formula $A_1 \wedge \ldots \wedge A_n \wedge \neg B$ is inconsistent, i.e., unsatisfiable. Let $\square$ denote an empty clause (i.e., a contradiction) and let formulas $A_1, \ldots, A_n, \neg B$ be in conjunctive normal form. Hence to show that B follows logically from $A_1, \ldots, A_n$ it suffices to prove that $\square$ is contained in the set $\mathcal{S}$ of all clauses constituting $A_1, \ldots, A_n, \neg B$ or that $\square$ can be deduced from this set (the sense of the word 'deduce' will be explained below). This is the main idea of the method of resolution. Hence we can say that this method is a negative test calculus.

The resolution calculus introduced by J. A. Robinson (1965) is a logical calculus in which one works only with formulas in clause form. It has no logical axioms and only one inference rule (the resolution rule). The simplest version of this rule has the following form: if C_1 and C_2 are two clauses such that C_1 contains a literal L_1 and C_2 contains a literal L_2 which is inconsistent with L_1 (i.e., L_1 and L_2 are complementary literals) then one obtains a new clause C consisting of all literals of C_1 except L_1 and all literals of C_2 except L_2. Symbolically it can be written as:

$$\begin{array}{ll} \text{clause } C_1: & K_1 \vee \cdots \vee K_n \vee L \\ \text{clause } C_2: & M_1 \vee \cdots \vee M_m \vee \neg L \\ & \hline \\ & K_1 \vee \cdots \vee K_n \vee M_1 \vee \cdots \vee M_m \end{array}$$

The clause $K_1 \vee \cdots \vee K_n \vee M_1 \vee \cdots \vee M_m$ is called the resolvent of C_1 and C_2 and clauses C_1 and C_2 are called parent clauses. We say that L_1 and L_2 are the literals resolved upon when the resolvent exists. Observe that the resolvent is a logical consequence of the parent clauses.

Let $\mathcal{S}$ be a set of clauses. A refutation (or a proof of unsatisfiability) of $\mathcal{S}$ is a finite sequence $C_1, \ldots, C_n$ of clauses such that (1) any C_i $(i = 1, \ldots, n)$ either belongs to $\mathcal{S}$ or there exist C_j and C_k, $j, k < i$ such that C_i is a resolvent of C_j and C_k and (2) the last clause C_n is $\Box$. This explains what we meant above by saying that $\Box$ can be "deduced" from the set $\mathcal{S}$ of clauses.

We give now two examples.

A. We show that U is a logical consequence of formulas $P \longrightarrow S$, $S \longrightarrow U$ and P. Indeed it suffices to show that the formula $(P \longrightarrow S) \wedge$ $\wedge (S \longrightarrow U) \wedge P \wedge \neg U$ is unsatisfiable (inconsistent). Writing it in conjunctive normal form we get

$$(\neg P \vee S) \wedge (\neg S \vee U) \wedge P \wedge \neg U.$$

Its clause form is $\{\neg P \vee S, \neg S \vee U, P, \neg U\}$. Denote this set of clauses by $\mathcal{S}$. The following sequence of formulas is a proof of unsatisfiability of $\mathcal{S}$:

(1) $\neg P \vee S$
(2) $\neg S \vee U$
(3) P
(4) $\neg U$
(5) $\neg P \vee U$ resolvent of (1) and (2)
(6) U resolvent of (3) and (5)
(7) $\Box$ resolvent of (4) and (6).

Hence U is a logical consequence of $P \longrightarrow S$, $S \longrightarrow U$ and P.

B. We show that the formula $\neg Q$ is a logical consequence of $P \longrightarrow$ $\longrightarrow (\neg Q \vee (R \wedge S)), P, \neg S$. Hence consider the formula

$$(P \longrightarrow (\neg Q \vee (R \wedge S))) \wedge P \wedge \neg S \wedge Q.$$

Its clause form is

$$\mathcal{S} = \{\neg P \vee \neg Q \vee R, \neg P \vee \neg Q \vee S, P, \neg S, Q\}.$$

We have the following proof of unsatisfiability of $\mathcal{S}$:

(1) $\neg P \vee \neg Q \vee R$
(2) $\neg P \vee \neg Q \vee S$
(3) P
(4) $\neg S$
(5) Q
(6) $\neg P \vee \neg Q$ resolvent of (2) and (4)
(7) $\neg P$ resolvent of (5) and (6)
(8) $\square$ resolvent of (3) and (7).

So far we considered the simplest form of the resolution principle and its application in the propositional calculus. In the case of formulas containing variables the whole situation is more complicated.

First we describe a substitution device called unification (we shall do it following Chang-Lee 1973). By a substitution we mean a finite set of the form $\{t_1/v_1, \ldots, t_n/v_n\}$ where v_i are variables and t_i are terms different from v_i. An empty substitution will be denoted by ε. If $\theta = \{t_1/v_1, \ldots, t_n/v_n\}$ is a substitution and E is a formula then by $E\theta$ we denote the formula $E(v_1/t_1, \ldots, v_n/t_n)$. Observe that substitutions can be composed, i.e., if $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ and $\lambda = \{u_1/y_1, \ldots, u_m/y_m\}$ are two substitutions then by $\theta \circ \lambda$ we shall denote a composition of θ and λ and define it as a substitution obtained from the set $\{t_1\lambda/x_1, \ldots, t_n\lambda/x_n, u_1/y_1, \ldots, u_m/y_m\}$ by removing from it all the elements $t_j\lambda/x_j$ such that $t_j\lambda = x_j$ and all the elements u_i/y_i such that $y_i \in \{x_1, \ldots, x_n\}$. Note that the composition $\circ$ is associative and that ε is the left and right unit, i.e., $\varepsilon \circ \theta = \theta \circ \varepsilon = \varepsilon$.

A substitution θ is said to be a unifier of a set of formulas $\{E_1, \ldots, E_k\}$ if and only if $E_1\theta = E_2\theta = \cdots = E_k\theta$. If there exists a unifier of a set $\{E_1, \ldots, E_k\}$ then this set is said to be unifiable. A unifier σ of the set $\{E_1, \ldots, E_k\}$ is called a most general unifier if and only if for any unifier θ of this set there exists a substitution λ such that $\theta = \sigma \circ \lambda$. J. A. Robinson showed that for any set $\mathcal{S}$ of formulas there exists at most one most general unifier.

We shall describe now an algorithm of finding a most general unifier. It is called the unification algorithm.

Let $\mathcal{S}$ be a nonempty set of formulas. A disagreement set of $\mathcal{S}$ is defined as follows: one indicates the first (from the left) position such that there are two formulas from $\mathcal{S}$ which differ on this position. Then for

every element E of $\mathcal{S}$ we write its subformula beginning with the symbol being written on this position. The set of these subformulas is just the disagreement set of $\mathcal{S}$. For example the disagreement set of the set $\{P(x, f(y, z)), P(x, a), P(x, g(h(k(x))))\}$ is $\{f(y, z), a, g(h(k(x)))\}$.

The unification algorithm can now be given by the following rules. Let a set $\mathcal{S}$ of formulas be given.

Step 1: $\mathcal{S}_0 = \mathcal{S}$, $\sigma_0 = \varepsilon$.

Step 2: If $\mathcal{S}_k$ is a unit clause then STOP and σ_k is the most general unifier of $\mathcal{S}$. Otherwise let D_k be the disagreement set of $\mathcal{S}_k$.

Step 3: If there are v_k and t_k in D_k such that v_k is a variable not occuring in t_k then move to Step 4. Otherwise STOP: $\mathcal{S}$ is not unifiable.

Step 4: Let $\sigma_{k+1} = \sigma_k \circ \{t_k/v_k\}$ and $\mathcal{S}_{k+1} = \mathcal{S}_k\{t_k/v_k\}$. (Observe that $\mathcal{S}_{k+1} = \mathcal{S}\sigma_{k+1}$.)

Note that the unification algorithm always halts when applied to a finite nonempty set of formulas. J. A. Robinson proved that if $\mathcal{S}$ is a finite nonempty unifiable set of formulas then the unification algorithm halts on Step 2 and the last σ_k is the most general unifier of $\mathcal{S}$.

We give some examples. First find a most general unifier of the set $\mathcal{S} = \{Q(f(a), g(x)), Q(y, y)\}$. We proceed as follows:

1. $\sigma_0 = \varepsilon$, $\mathcal{S}_0 = \mathcal{S}$.
2. Since $\mathcal{S}_0$ is not a unit clause we find a disagreement set D_0 of $\mathcal{S}_0$. We have $D_0 = \{f(a), y\}$. Hence $v_0 = y, t_0 = f(a)$.
3. $\sigma_1 = \sigma_0 \circ \{t_0/v_0\} = \sigma_0 \circ \{f(a)/y\} = \{f(a)/y\}$,
   $\mathcal{S}_1 = \mathcal{S}_0\{t_0/v_0\} = \{Q(f(a), g(x)), Q(f(a), f(a))\}$.
4. $\mathcal{S}_1$ is not a unit clause and we have $D_1 = \{g(x), f(a)\}$. By Step 3 we conclude that the set $\mathcal{S}$ is not unifiable.

We give one more example.

Let $\mathcal{S} = \{P(a, x, f(g(x))), P(z, f(z), f(u))\}$. Find a most general unifier of $\mathcal{S}$. We proceed as follows:

1. $\sigma_0 = \varepsilon$, $\mathcal{S}_0 = \mathcal{S}$.
2. $D_0 = \{a, z\}$ and we have $v_0 = z$, $t_0 = a$.
3. $\sigma_1 = \sigma_0 \circ \{t_0/v_0\} = \{a/z\}$,
   $\mathcal{S}_1 = \mathcal{S}_0\{t_0/v_0\} = \{P(a, x, f(g(y))), P(a, f(a), f(u))\}$.
4. $D_1 = \{x, f(a)\}$ and we put $v_1 = x$, $t_1 = f(a)$.
5. $\sigma_2 = \sigma_1 \circ \{t_1/v_1\} = \{a/z, f(a)/x\}$,
   $\mathcal{S}_2 = \mathcal{S}_1 \circ \{t_1/v_1\} = \{P(a, f(a), f(g(y))), P(a, f(a), f(u))\}$.
6. $D_2 = \{g(y), u\}$ and we put $v_2 = u$, $t_2 = g(y)$.

7. $\sigma_3 = \sigma_2 \circ \{t_2/v_2\} = \{a/z, f(a)/x, g(y)/u\}$,
   $\mathcal{S}_3 = \mathcal{S}_2\{t_2/v_2\} = \{P(a, f(a), f(g(y)))\}$.

By Step 2 the set $\mathcal{S}$ is unifiable and σ_3 is the most general unifier of it.

We have to define two more notions to formulate at last the general form of the resolution rule. If two or more literals (with the same sign) of a clause C have a most general unifier σ then the clause $C\sigma$ is called a factor of C. Let C_1 and C_2 be two clauses which have no common variable and let L_1 and L_2 be two literals of C_1 and C_2, resp. If L_1 and $\neg L_2$ have a most general unifier σ then the clause $(C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma)$ is called a binary resolvent of C_1 and C_2. The clauses C_1 and C_2 are called parent clauses and we say that L_1 and L_2 are literals resolved upon. A resolvent of two parent clauses C_1 and C_2 is now defined as one of the following resolvents: (1) binary resolvent of C_1 and C_2, (2) binary resolvent of C_1 and a factor of C_2, (3) binary resolvent of C_2 and a factor of C_1, (4) binary resolvent of a factor C_1 and a factor of C_2.

Now we can define a resolution deduction. Let a set $\mathcal{S}$ of clauses and a clause A be given. A resolution deduction of A from $\mathcal{S}$ is a finite sequence $C_1, \ldots, C_n$ of clauses such that: (1) C_n is A, (2) for any $i, 1 \leq i \leq n$, C_i is either a member of $\mathcal{S}$ or there exist $j, k < i$ such that C_i is a resolvent of C_j and C_k (i.e., C_i is obtained from C_j and C_k by the resolution rule). A resolution deduction of the empty clause $\square$ from $\mathcal{S}$ is called a refutation (or a proof of unsatisfiability) of $\mathcal{S}$.

We shall give some examples (cf. Chang-Lee 1973).

1. Show that the formula $B =: \exists x(S(x) \wedge R(x))$ is a logical consequence of formulas $A_1 =: \forall x[P(x) \longrightarrow (Q(x) \wedge R(x))]$ and $A_2 =: \exists x(P(x) \wedge S(x))$. It suffices to show that the formula $A_1 \wedge A_2 \wedge \neg B$ is unsatisfiable. Transform the given formulas $A_1, A_2, \neg B$ into conjunctive normal form. We get the following formulas, resp.:

$(\neg P(x) \vee Q(x)) \wedge (\neg P(x) \vee R(x))$,
$P(a) \wedge S(a)$,
$\neg S(x) \vee \neg R(x)$.

We can construct now the following resolution deduction:

(1)	$\neg P(x) \vee Q(x)$	from A_1
(2)	$\neg P(x) \vee R(x)$	from A_1
(3)	$P(a)$	from A_2
(4)	$S(a)$	from A_2
(5)	$\neg S(x) \vee \neg R(x)$	from $\neg B$

(6) $R(a)$ resolvent of (2) and (3)
(7) $\neg R(a)$ resolvent of (4) and (5)
(8) $\square$ resolvent of (6) and (7).

Hence we have shown that B is a logical consequence of A_1 and A_2.

2. We show that the formula $B =: \exists x(P(x) \wedge R(x))$ is a logical consequence of the following formulas:

$A_1 =: \forall x[Q(x) \wedge \neg T(x) \longrightarrow \exists y(S(x,y) \wedge R(y))]$,
$A_2 =: \exists x[P(x) \wedge Q(x) \wedge \forall y(S(x,y) \longrightarrow P(y))]$,
$A_3 =: \forall x(P(x) \longrightarrow \neg T(x))$.

Transforming the formulas A_1, A_2, A_3 and $\neg B$ into conjunctive normal form we get the following clauses:

(1) $\neg Q(x) \vee T(x) \vee S(x, f(x))$ } from A_1
(2) $\neg Q(x) \vee T(x) \vee R(f(x))$ } from A_1
(3) $P(a)$ } from A_2
(4) $Q(a)$ } from A_2
(5) $\neg S(a,y) \vee P(y)$ } from A_2
(6) $\neg P(x) \vee \neg T(x)$ from A_3
(7) $\neg P(x) \vee \neg R(x)$ from $\neg B$

The needed deduction of $\square$ can be the following:

(8) $\neg T(a)$ resolvent of (3) and (6)
(9) $T(a) \vee R(f(a))$ resolvent of (2) and (4)
(10) $R(f(a))$ resolvent of (8) and (9)
(11) $T(a) \vee S(a, f(a))$ resolvent of (1) and (4)
(12) $S(a, f(a))$ resolvent of (8) and (11)
(13) $P(f(a))$ resolvent of (5) and (12)
(14) $\neg R(f(a))$ resolvent of (7) and (13)
(15) $\square$ resolvent of (10) and (14).

The method of resolution has an important property called the refutation completeness. One can prove that a set $\mathcal{S}$ of clauses is unsatisfiable if and only if there exists a resolution deduction of the empty clause $\square$ from $\mathcal{S}$. The proof of this fact uses semantical trees and Herbrand's theorem. Hence the method gives a semidecidability of the predicate logic: if a given formula is a logical consequence of a given set $\mathcal{S}$ of formulas then by a systematic application of the resolution rule we get in a finite number of steps the empty

clause $\square$, but if it is not a logical consequence then sometimes one can decide it after a finite number of steps but in general the procedure does not halt. On the other hand the considered method does not give a procedure of finding a formal proof of a formula B on the basis of formulas $A_1, \ldots, A_n$ in the case when B is a logical consequence of $A_1, \ldots, A_n$.

Recall that we used in the resolution most general unifiers, i.e., most general substitutions that allow the equality of literals. This guarantees the elimination of branching of search due to different possible substitutions that equate those atoms but lead to different clauses. Therefore the method of resolution is simple, elegant and powerful. But the world is not so perfectly beautiful – this method has also some defects. If one generates from a given set of clauses new clauses by the method of resolution then they accumulate at a rapid rate. Indeed given a set $\mathcal{S}$ of clauses one can obtain new clauses systematically using the level-saturation method which is described by the following equations: $\mathcal{S}^0 = \mathcal{S}$, $\mathcal{S}^{n+1} = \{\text{resolvents of } C_1 \text{ and } C_2 : C_1 \in \mathcal{S}^0 \cup \cup \ldots \cup \mathcal{S}^n, C_2 \in \mathcal{S}^n\}$, $n = 1, 2, \ldots$ In this way we get all the resolvents of all pairs of elements of $\mathcal{S}$, add them to $\mathcal{S}$, further we calculate all the resolvents of elements of this new set, etc., till we come to the empty clause $\square$. Among clauses generated in this procedure there are many irrelevant ones and the total number of clauses grows rapidly. Hence the idea of improving the method of resolution by finding restrictions and strategies to control the growth of the number of clauses. We shall tell here only about some of the proposed improvements (Loveland 1978 summarizes in Appendix twenty five such improvements but more exist). By a restriction of resolution we mean a variant for which some clauses generated by the basic resolution procedure are not generated. A strategy of resolution only rearranges the order of generation to get likely useful clauses earlier.

One of the earliest refinements of resolution is unit-preference introduced in Wos-Carson-Robinson (1964). This strategy guarantees that the resolvent is shorter than the longer parent clauses. In the same paper L. Wos, D. F. Robinson and G. A. Carson introduced the set-of-support restriction. It can be described as follows: one chooses a subset $\mathcal{T}$ (called a support) of a given set $\mathcal{S}$ of clauses and then two clauses from $\mathcal{S} - \mathcal{T}$ are never resolved together. This means that every resolvent has in its deduction history some clause of $\mathcal{T}$. In practice $\mathcal{T}$ is usually chosen to be a (sub)set of the clauses special to the considered problem.

J. A. Robinson in (1965a) introduced a restriction called hyperresolution. It restricts resolutions to where one parent clause contains only positive literals and any resolvent containing a negative literal is immediately used in all permitted resolutions and then discarded.

D. W. Loveland (1970) and D. Luckham (1970) proposed another restriction called linear resolution (Luckham used the name "ancestry-filter form"). It constrains the deduction so that a new clause is always derived from the preceding clause of the deduction by resolving against an earlier clause of the deduction. In this way one is always seeking to transform the last clause obtained into a clause closer to the goal clause. This method was further developed by R. Anderson and W. W. Bledsoe (1970). R. Yates, B. Raphael and T. Hart (1970), R. Reiter (1971), D. W. Loveland (1972) and R. Kowalski and D. Kuehner (1971).

D. W. Loveland (1968) and (1969) introduced a procedure which is not a resolution procedure but is close to a very restricted form of linear resolution – it is called model elimination. R. Kowalski and D. Kuehner (1971) provided the translation of it into a very restricted linear resolution format called SL-resolution. It was used by A. Colmerauer and P. Roussel to an early version of a programming language Prolog.

There appeared a number of resolution refinements which reduce multiple derivations of the same clause by ordering literals in clauses. An example of this type of procedures is the method called locking or lock-resolution introduced by R. S. Boyer (1971). Its main idea is to use indices to order literals in clauses from a given set of clauses. The occurrences of literals are indexed by integers. Then resolution need be done using only the lowest indexed integer of each clause.

We should mention here also the semantic resolution of J. R. Slagle (1967) which generalizes hyperresolution of J. A. Robinson (1965a), resolution with renaming of B. Meltzer (1966) and the set-of-support restriction of L. Wos, G. A. Robinson and D. F. Carson (1965).

All types of refinements of resolution given above are refutation complete. There exist also two forms of resolution restrictions which are incomplete. We mean here unit clause resolution and input clause resolution. The former was introduced by L. Wos, D. F. Carson and G. A. Robinson (1964). It permits resolution only when one parent is a unit clause, i.e., it consists of one literal. The input clause resolution was introduced by C. L. Chang (1970). It is a restricted form of linear resolution where one parent is always an input (given) clause. Chang proved that the unit clause and input clause resolutions are of equal power – they are complete over the class of Horn formulas.

Most mathematical theories contain among its symbols the equality symbol and among its axioms – the equality axioms. The immediate application of the usual resolution procedure generates in this case a lot of undesired clauses. Hence L. Wos and G. A. Robinson (1970) introduced

a procedure called paramodulation. This is the equality replacement rule with unification. It replaces all equality axioms except certain reflexivity axioms for functions. When paramodulation is restricted to replacement of the (usually) shorter term by the longer term with no instantiation allowed in the formula incorporating the replacement then one uses the term demodulation (cf. Wos-Robinson-Carson-Shalla 1967). It should be mentioned that there are also two other systems treating equality: system introduced by E. E. Sibert (1969) and E-resolution introduced by J. B. Morris (1969).

To finish this section we should add that almost simultaneosly with Robinson's invention of resolution, J. Ju. Maslov in the USSR introduced a proof procedure very close to resolution in spirit. His method is called the inverse method and it is a test for validity rather than unsatisfiability (cf. Maslov 1964, 1971 and Kuehner 1971).

4. Development of mechanized deduction after 1965

The last section was devoted to the method of resolution and to its refinements. This approach to the mechanization and automatization of reasonings was dominating in the sixties. Nevertheless there were developed also other methods. In this section we shall discuss them briefly and sketch further development of the researches in the field after 1965.

We should begin with the Semi Automated Mathematics (SAM) project that spanned 1963 to 1967. It belongs to the human simulation approach (cf. Section 1). In the framework of this project a succession of systems designed to interact with a mathematician was developed. They used many sorted ω-order logic with equality and λ-notation. The system SAM I was a proof checker but the theorem proving power continued to increase through SAM V which had substantial automatic capability. Only a part of the work in this project got recorded in the literature – cf. J. R. Guard-F. C. Oglesby-J. H. Bennett-L. G. Settle (1969).

Another project belonging to the human simulation approach developed in the mid-sixties was ADEPT, the Ph. D. thesis of L. M. Norton (cf. Norton 1966). It was a heuristic prover for group theory.

The dominant position of the resolution methods brought sharp criticism from some researchers. Their main argument was that there cannot be a unique procedure which would suffice to realize (to simulate) the real intelligence. They stressed the necessity of using many components. One of those critics was M. Minsky from MIT. In 1970 C. Hewitt, a Ph. D.

student at MIT wrote a dissertation on a new programmic language called PLANNER (cf. Hewitt 1971). Its goal was to structure a theorem prover system in such a way that locally distributed knowledge could be represented at various positions of the proving program. In fact it was not a theorem prover *per se*, but a language in which a "user" was to write his own theorem prover, specifically tailored to the problem domain at hand. This language was never fully implemented, only a subset of it was realized (microPLANNER).

About the same time another effort in human similation approach was undertaken by A. Nevins (cf. Nevins 1974, 1975, 1975a). He built in fact at least two provers that were able to prove theorems which most resolution provers could not touch. For example Nevins could prove fully automatically that:

$$x^3 = e \longrightarrow f(f(a,b),b) = e$$

where $f(x,y) = xyx^{-1}y^{-1}$. This result is much harder than the implication: "$x^2 = e \longrightarrow$ the group is abelian" which constituted the limit of capability of ADEPT.

We should mention also works of the group of scientists gathered around W. W. Bledsoe at the University of Texas which proved to be very important. They were working not for single uniform rule of inference for the whole mathematics but were seeking specific methods for particular domains of mathematics such as analysis, set theory or nonstandard analysis (cf. Bledsoe 1983, 1984).

So far we spoke about studies in seventies of the automated theorem proving which could be classified as human simulation approach. It does not mean that this was the only direction. There were also some new ideas within the logic approach. We should say here first of D. E. Knuth and P. B. Bendix (1970). They used the idea of rewrite rules – a device familiar to logicians. It is a replacement rule and allow to replace the left hand side by the right hand side at any occurrence of the left hand side. In equational theories one converts equations just to rewrite rules. Those rules enable us to reduce terms and to equate them. Knuth and Bendix proposed an algorithm which for a class of equational theories gave a complete set of rewrite rules – i.e., a set of rewrite rules sufficient to check the truth of every equation of the theory by demanding that equal terms reduce to the same normal form. An example of a theory to which the algorithm applies is the theory of groups (while the theory of abelian groups is not). There is an open problem connected with this algorithm: what equational theories have complete sets of rewrite rules?

Recall that the unification algorithm (described in the previous section) permits computation of a most general substitution for variables to make atomic formulas identical. Working with special theories, usually equational theories, one can simplify the procedure given by unification and resolution by introducing special unification. Its main idea is that the usual unification is augmented by equations or rewrite rules obtained from axioms of the considered theory. This idea was first formalized by G. D. Plotkin (1972) and further developed by M. E. Stickel (1981), (1985) and M. Livesey and J. Siekmann (1976). The idea of theory resolution of Stickel can be summerized as follows: since the resolution rule enlarges the whole number of steps, it is desirable to find macrorules in which certain sequences of steps could be performed in one step.

Another example of results which should be classified as logic oriented is the system of R. Overbeek developed later by S. Winker, E. Lusk, B. Smith and L. Wos and named AURA (Automated Reasoning Assistant). It was based on the old (i.e., coming from the sixties) ideas of unit preference, set-of-support for resolution, paramodulation and demodulation to which hyperresolution as well as more flexibility in demodulation and preprocessors for preparation of input from a variety of formats were added.

Around 1973 there appeared a very interesting effort different from the resolution approach and the strongly human oriented prover of Bledsoe. We mean here the Computational Logic Theorem Prover of R. S. Boyer and J. S. Moore (1975), (1979), (1981). This system uses the language of quantifier-free first order logic with equality and includes a general induction principle among the inference rules. It can be used to work within traditional mathematics (e.g., number theory) as well as to prove properties of programs and algorithms (so called proofs of correctness).

We should tell also about graph representation and about prover for systems of higher order. The former is based on the idea of enriching the structure of basic data with additional information, e.g. by representing the potential resolution steps in the graph structure. Literals or clauses and possible complementary literals form vertices of graphs which are connected by edges. This approach was introduced by R. Kowalski (1975) (cf. also S. Sickel 1976, R. E. Shostak 1976 and P. Andrews 1976).

The first proving system for higher order logics was developed by a group of scientists working under the direction of J. R. Guard in the early sixties. The studies were continued by W. E. Gould (1966), G. P. Huet (1975) and D. C. Jensen and T. Pietrzykowski (1976). The most important and influential group of people working in this direction is today at the Carnegie-Melon University (its chief is P. B. Andrews). They developed

in the late seventies a theorem prover for type theory (TPS) (cf. Andrews 1981 and Miller, Cohen, Andrews 1982). It can prove for example Cantor's theorem as well as numerous first order theorems.

We shall finish this survey of activities in the field of mechanization and automatization of reasonings in the sixties and seventies by mentioning an ambitous theorem prover now being developed at the University of Karlsruhe and named Markgraf Karl Refutation Procedure (cf. K. Bläsius, N. Eininger, J. Siekmann, G. Smolka, A. Herold, C. Walther 1981).

5. Some limitations

Having described so far the positive achievements in the field of mechanized deduction and automated theorem proving let us turn now to the discussion of some (essential) limitations in this process.

It is a trivial observation that an automatic theorem prover would have wide application if it operated effectively enough. Hence it is useful to distinguish effective and feasible computability (decidability). Both are intuitive (nonformal) concepts. Recall that a problem is said to be effectively decidable (a function is effectively computable) if and only if there exists a definite mechanical procedure which can solve in a finite number of prescribed steps every instance of the problem (calculate the value of the function for any given arguments). It is believed that the concept of recursive computability (or equivalently of Turing machine computability) is an adequate mathematical formalization (counterpart) of this concept – this is stated by Church's Thesis. On the other hand by "feasible" one means "computable in practice" or "computable in the real world". As a mathematical model of this intuitive notion one can consider the concept of polynomial time computability, i.e., computability by a deterministic Turing machine in the time bounded by a polynomial of the size of the input, hence by a deterministic Turing machine that needs a number of steps bounded by a polynomial of the size of the input. Denote by P the class of predicates (problems) recognizable (solvable) in a polynomial time. A closely related class is the class NP of problems recognizable in nondeterministic polynomial time. There is an open problem whether problems in NP are feasibly decidable or are polynomial time decidable – it is shortly denoted as $P =?NP$ and is a central problem in the contemporary computer science.

It seems that this problem was stated (in a certain sense) for the first time in a letter of Kurt Gödel to John von Neumann from 20th March 1956 (cf. Gödel 2003, p. 373–375). Gödel was thinking about computational

complexity of Turing machine computations and asked von Neumann about the computational complexity of a problem (which is in fact an NP complete problem; note that we are using the modern terminology, in Gödel's letter the problem is not referred to as NP complete) about proof systems and wondered if it could be solved in linear or possibly quadratic time. He asked how hard is it (computationally) to decide if a statement has a proof of length n in a formal system (it is of course a question about the fundamental nature of mathematics). It is worth quoting here some fragments of Gödel's letter:

> Obviously, it is easy to construct a Turing machine that allows us to decide, for each formula F of the restricted functional calculus and every natural number n, whether F has a proof of length n [length = number of symbols]. Let $\psi(F, n)$ be the number of steps required for the machine to do that, and let $\varphi(n) = \max_F \psi(F, n)$. The question is, how rapidly does $\varphi(n)$ grow for an optimal machine? It is possible to show that $\varphi(n) \geq Kn$. If there really were a machine with $\varphi(n) \sim Kn$ (or even just $\sim Kn^2$) then that would have consequences of the greatest significance. Namely, this would clearly mean that the thinking of a mathematician in the case of yes-or-no questions could be completely[1] replaced by machine, in spite of the unsolvability of the Entscheidungsproblem. n would merely have to be chosen so large that, when the machine does not provide a result, it also does not make any sense to think about the problem. Now it seems to me to be quite within the realm of possibility that $\varphi(n)$ grows so slowly. For 1.) $\varphi(n) \geq Kn$ seems to be the only estimate obtainable by generalizing the proof of the unsolvability of the Entscheidungsproblem; 2.) $\varphi(n) \sim Kn$ (or $\sim Kn^2$) just means that the number of steps when compared to pure trial and error can be reduced from N to $log\, N$ (or $log\, N^2$). Such significant reductions are definitely involved in the case of other finitist problems, e.g., when computing the quadratic remainder symbol by repeated application of the law of reciprocity. It would be interesting to know what the case would be, e.g., in determining whether a number is prime, and how significantly *in general* for finitist combinatorial problems the number of steps can be reduced when compared to pure trial and error.

To formulate the problems raised by Gödel more clearly let us think of first-order predicate logic formalized in one of the usual Hilbert-style systems with a finite set of axiom schemata and modus ponens and generalization as the only rules of inference. Let the symbol $\vdash_n \varphi$ denote that φ has a first-order proof of $\leq n$ symbols. Gödel was asking about the difficulty of answering questions of the form "$\vdash_n \varphi$?". Let $A = \{\langle \varphi, 0^n \rangle : \vdash_n \varphi\}$.

[1] Except for the formulation of axioms.

Gödel's questions is now: is the set A recognizable on a (multitape) Turing machine in time $O(n)$ or in time $O(n^2)$. It can be shown that there is an effective algorithm for deciding membership in A and that the set A is in fact NP-complete (it is NP-complete even for propositional logic – cf. Buss 1995).

It is interesting that Gödel was thinking in his letter about problems related to $P =?NP$ well before these classes were widely stated (cf. Hartmanis 1989 for the discussion of Gödel's letter). Note also that Gödel treated linear or quadratic time computability as corresponding to feasible computation and did not realize the importance of polynomial time as a mathematical model of feasible computability.

The problem $P =?NP$ belongs to the most famous open problems in computer science. On the other hand it is known today that even in the case of simple mathematical theories the decision procedures are of high complexity. The following theorems can serve as examples indicating the measure of the complexity.

Let F be a function defined in the following way:

$$F(n,1) = 2^n, \qquad F(n,m+1) = 2^{F(n,m)}.$$

Denote by $l(\varphi)$ the length of a formula φ, i.e., the number of (logical and nonlogical) symbols occurring in φ.

It has been shown that:

- (Meyer 1975) Let $f(n)$ be the function

  $$F(n,[dn]).$$

  There exists a constant $d > 0$ such that for any Turing machine P deciding the weak second-order monadic theory of one successor WS1S there exist infinitely many sentences φ with the property that the machine P needs more than $f(l(\varphi))$ steps to decide whether $\varphi \in$ WS1S or not.

- (Meyer 1975) The complexity of the theory of linear order is at least $F(n,[dn])$ for a certain positive constant d, i.e., to decide a formula of the length n one needs at least $F(n,[dn])$ steps.

- (Fisher and Rabin 1974) A decision procedure for Presburger arithmetic is of the complexity at least $2^{2^{cn}}$ for a certain constant $c > 0$, i.e., to decide whether a formula φ of the length n is a theorem of Presburger arithmetic one needs at least $2^{2^{cn}}$ steps.

- (Fisher and Rabin 1974) The complexity of the theory $Th(\langle \mathbf{N}, \cdot \rangle)$ is at least $F(cn,3)$, i.e., $2^{2^{2^{cn}}}$ for a certain constant $c > 0$.

So one sees that even for such simple theories as $Th(\langle \mathbf{N}, + \rangle)$ or $Th(\langle \mathbf{N}, \cdot \rangle)$ decision procedures are of exponential complexity. Add also that the decision procedure for the theory of real numbers is doubly exponential in the number of quantifier blocks (cf. Heintz *et al.* 1989) and that even deciding first-order sentences for the ordered group $\langle \mathbb{R}, +, < \rangle$ is exponentially hard (cf. Fisher *et al.* 1974).

On the other hand if one moves from the level of first-order logic to higher systems (or generally from the level n to the level $n+1$) the complexity of proofs (their lengths) can be reduced. This observation was made by Gödel in (1936). Having already shown that a logic S_{n+1} of a higher order could prove formulas that a logic S_n of a lower order could not prove, in this abstract he considered the question of formulas that can be proved in both the weaker and the stronger logics. He stated that if the length of a proof is defined to be the number of lines in it, there are formulas that can be proved in both S_n and S_{n+1} but that have a proof in S_{n+1} much shorter than their shortest proof in S_n. This speed-up can be by arbitrary function computable in S_n, i.e., for any function F computable in S_n there exist infinitely many formulas φ such that if k is the length of a shortest proof of φ in S_n and l is the length of a shortest proof of φ in S_{n+1} then $k > F(l)$. Hence "passing to the logic of the next higher order has the effect, not only of making provable certain propositions that were not provable before, but also of making it possible to shorten, by an extraordinary amount, infinitely many of the proofs already available" (Gödel 1936).

Gödel did not give a proof of his result. An analogous result (taking the length of a proof to be its Gödel number rather than the number of lines in it) was given by Mostowski in (1952). Similar results were also proved by Ehrenfeucht and Mycielski (1971) and by Parikh (1971). Statman in (1978) has shown that there is no function F provably recursive in second-order arithmetic such that whenever a first-order formula φ is derivable in a certain standard system of second-order logic with length $\leq l$ then φ is derivable in a certain standard system of first-order logic with length $\leq F(l)$.

Note that the problem $P =? NP$ discussed above can be thought of as a speed-up question.

A nice illustration of the phenomenon indicated by Gödel was given by Boolos in (1987). He considered there the following set of axioms (in the language with function symbols F and S and a unary predicate D):

(1) $\forall n F(n, 1) = S(1)$,

(2) $\forall x F(1, S(x)) = SS(F(1, x))$,

(3) $\forall n \forall x F(S(n), S(x)) = F(n, F(S(n), x))$,

(4) $D(1)$,

(5) $\forall x[D(x) \longrightarrow D(S(x))]$.

In the intended interpretation of the above formulas the variables range over the natural numbers, 1 denotes the number one and S is the successor function. There is no particular interpretation intended for D. By this interpretation F denotes an Ackermann-style function $f : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ defined by the following equations: $f(1, x) = 2x$, $f(n, 1) = 2$ and $f(n + 1, x + 1) = f(n, f(n + 1, x))$. This is a rapidly growing function. In fact one can easily show that $f(1, x) = 2x$, $f(2, x) = 2^x$, $f(3, x) = 2^{2^{\cdot^{\cdot^{\cdot^2}}}}$ (the value of a stack of x 2's), $f(4, 1) = 2$, $f(4, 2) = 4$, $f(4, 3) = 2^{2^{\cdot^{\cdot^{\cdot^2}}}}$ (= 64 K), $f(4, 4) = 2^{2^{\cdot^{\cdot^{\cdot^2}}}}$ (64 K of 2's in all).

In second-order logic one can deduce from the given set of axioms that

$$D(F(SSSS(1), SSSS(1)).$$

The usage of the second-order logic is essential here (one uses, e.g., the comprehension principle). The proof can be found in (Boolos 1987, Appendix). This formula can be also proved in a first-order system but any derivation of it must contain at least $f(4, 4)$ symbols (!) – details of the proof of this (metatheoretical) statement can be found in Appendix to (Boolos 1987).[2] Hence it is impossible to write down such a proof, no actual or conceivable creature or device could do it. There are simply far too many symbols in any such derivation. It shows that first-order logic is in a certain sense practically incomplete. Boolos says even in (1987, p. 135) that "no standard first-order logical system can be taken to be a satisfactory idealization of the psychological mechanisms or processes, whatever they might be, whereby we recognize (first-order!) logical consequences. "Cognitive scientists" ought to be suspicious of the view that logic as it appears in logic texts adequately represents the whole of the science of valid inference." We can add that this thesis should be taken into account not only by cognitive scientists but also by specialists in mechanized deduction and automated theorem proving (as well as in the artificial intelligence).

[2] There is another example of a similar result: H. Friedman has shown (cf. Nerode and Harrington 1984) that a certain "finitization" of a combinatorial theorem due to J. Kruskal concerning embeddings of trees can be proved in Zermelo-Fraenkel set theory with the axiom of choice (ZFC) in a few pages but not in the system of second-order arithmetic called ATR (for Arithmetic Transfinite Recursion) in under $f(3, 1000)$ pages.

6. Final remarks

In the previous sections we have presented the recent period of the history of efforts to find an automated theorem prover. They were stimulated by the appearence of computers which made possible the practical realization of earlier ideas. The emphasis was put on the resolution procedure and the unification algorithm and their modifications because they proved to be the most influential ideas.

As was proved by Turing, Gödel and Church there exists no universal automated theorem prover for the whole mathematics – and even more, there are no such provers for most mathematical theories. Proving theorems in mathematics and logic is too complex a task for total automation because it requires insight, deep thought and much knowledge and experience. Nevertheless the semidecidability of mathematical theories was a sufficient motivation for looking for weaker theorem provers. We have described those efforts in the previous sections.

What does one expect from an automated theorem prover? First of all one obtains a certain unification of reasonings and their automatization. Having that one can shift the burden of proof finding from a mathematician and a logician to the computer. In this way we are also assured that faulty proofs would never occur. Are such automated theorem provers more clever than people? Of course they can proceed quicker than a human being. But can they discover new mathematical results? The answer is YES. Some open questions have been answered in this way within finitely axiomatizable theories. For example S. Winker, L. Wos and E. Lusk (1981) answered positively the following open question: does there exist a finite semigroup which simultaneously admits of a nontrivial antiautomorphism without admitting a nontrivial involution? The progress in more complex theories such as analysis or set theory is slower, but there are also provers being able to prove some nontrivial theorems such as for example Cantor's theorem stating that a set has more subsets than elements (cf. P. Andrews, D. A. Miller, E. L. Cohen, F. Pfenning 1984) and various theorems in introductory analysis. The latter includes limit theorems of calculus such as

- the sum, product and composition of two continuous functions is continuous,
- differentiable functions are continuous,
- a uniformly continuous function is continuous,

as well as theorems of intermediate analysis (on the real numbers) such as

- Bolzano-Weierstrass theorem,

- if the function f is continuous on the compact set S then f is uniformly continuous on S,
- if f is continuous on the compact set S then $f[S]$ is compact,
- intermediate value theorem

(cf. Bledsoe 1984).

All those achievements can be treated as partial realizations and fulfilments of Leibniz's dreams of *characteristica universalis* and *calculus ratiocinator*. They are still far from what Leibniz did expect but they prove that a certain progress in the mechanization and automatization of reasonings and generally human thought has been made. On the other hand one should be aware of some limitations indicated above.

References

Anderson, R. and W. W. Bledsoe, 1970, A Linear Format for Resolution with Merging and a New Technique for Establishing Completeness, *Journal of the Association for Computing Machines*, 17, 525–534.

Andrews, P., 1976, Refutations by Matings. *IEEE Trans. on Computers*, C-25, 801–807.

Andrews, P., 1981, Transforming Matings into Natural Deduction Proofs. In: *Proc. Fifth Conf. on Automated Deduction*, ed. by W. Bibel and R. Kowalski, *Lecture Notes in Computer Science 87*, Berlin – Heidelberg – New York: Springer-Verlag, pp. 281–292.

Andrews, P., D. A. Miller, E. L. Cohen and F. Pfenning, 1984, Automating Higher-Order Logic. In: *Automated Theorem Proving. After 25 Years*, ed. by W. W. Bledsoe and D. W. Loveland, *Contemporary Mathematics*, 29, AMS, Providence, Rhode Island, pp. 169–192.

Bläsius, K., N. Eininger, J. Siekmann, G. Smolka, A. Herold and C. Walther, 1981, The Markgraf Karl refutation procedure. In: *Proc. Seventh Intern. Joint Conf. on Artificial Intelligence*, pp. 511–518.

Bledsoe, W. W., 1983, Using Examples to Generate Instantiations for Set Variables. In: *Proc. Intern. Joint Conf. on Artificial Intelligence*.

Bledsoe, W. W., 1984, Some Automatic Proofs in Analysis. In: *Automated Theorem Proving. After 25 Years*, ed. by W. W. Bledsoe and D. W. Loveland, *Contemporary Mathematics*, 29, AMS, Providence, Rhode Island, pp. 89–118.

Boolos G., 1987, A curious inference. *Journal of Philosophical Logic* 16, 1–12.

Boyer, R. S., 1971, *A Restriction of Resolution*. Ph. D. Thesis, University of Texas at Austin, Texas.

Boyer, R. S. and J. S. Moore, 1975, Proving Theorems about LISP Functions. *Journal of the Association for Computing Machines*, 22, 129–144.

Boyer, R. S. and J. S. Moore, 1979, *A Computational Logic*. New York: Academic Press.

Boyer, R. S. and J. S. Moore, 1981, A Verification Condition Generator for FORTRAN. In: *The Correctness Problem in Computer Science*, ed. by R .S. Boyer and J. S. Moore, London.

Buss S.R., 1995, On Gödel's theorem on lengths of proofs II: Lower bounds for recognizeing k symbol provability. In: *Feasible Mathematics II*, eds. P. Clote and J. Remmel, Birkhäuser, 57–90.

Chang, C. L., 1970, The Unit Proof and the Input Proof in Theorem Proving. *Journal of the Association for Computing Machines*, 17, 698–707.

Chang, C. L. and R. Lee, 1973, *Symbolic Logic and Mechanical Theorem Proving*. New York – San Francisco – London: Academic Press.

Chinlund, T. J., M. Davis, G. Hineman and D. McIlroy, 1964, *Theorem Proving by Matching*. Bell Laboratories.

Church, A., 1936, A Note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1, 40–41, 101–102.

Davis, M., 1963, Eliminating the Irrelevant from Mechanical Proofs. *Proc. Symp. Applied Mathematics*, 25, 15–30.

Davis, M. and H. Putnam, 1960, A Computing Procedure for Quantification Theory. *Journal of the Association for Computing Machines*, 7, 201–215.

Ehrenfeucht A. and Mycielski J., 1971, Abbreviating proofs by adding new axioms, *Bulletin of the American Mathematical Society* 77, 366–367.

Fisher, M. J. and Rabin, M. O., 1974, Super Exponential Complexity of Presburger's Arithmetic. In: Karp, R. (Ed.) *Complexity of Computation, Proc. SIAM-AMS Sympos. Appl. Math.* **7**, pp. 27–41.

Gelernter, H., 1959, Realization of a Geometry Theorem-Proving Machine. In: *Proc. Intern. Conf. on Information Processing*, Paris: UNESCO House. pp. 273–282. Also in: *Computers and Thought*, ed. by Feigenbaum and Feldman. McGrow-Hill.

Gelernter, H., J. R. Hanson and D. W. Loveland, 1960, Empirical Explorations of the Geometry-Theorem Proving Machine. In: *Proc. Western Joint Computer Conf.*, pp. 143–147. Also in: *Computers and Thought*, ed. by Feigenbaum and Feldman. McGrow-Hill.

Gilmore, P. C., 1959, A Program for the Production of Proofs for Theorems Derivable within the First Order Predicate Calculus from Axioms. In: *Proc. Intern. Conf. on Information Processing*. Paris: UNESCO House.

Gilmore, P. C., 1960, A Proof Method for Quantification Theory: its Justification and Realization. *IBM Journal Research and Devel.* 28–35.

Gödel K., 1936, Über die Länge von Beweisen, *Ergibnisse eines mathematischen Kolloquiums* 7, 23–24. Reprinted with English translation: 'On the length of proofs' in: Gödel K., *Collected Works*, vol. I, eds. Feferman S. *et al.*, Oxford University Press, New York, and Clarendon Press, Oxford 1986, 396–399.

Gödel K., 2003, Collected Works, vol. V, eds. Feferman S. et al., Clarendon Press, Oxford.

Gould, W. E., 1966, A Matching Procedure for ω-Order Logic. *Air Force Cambridge Research Laboratories, Report 66–781–4.*

Guard, J. R., F. C. Oglesby, J. H. Bennett and L. G. Settle, 1969, Semiautomated mathematics. *Journal of the Association for Computing Machines*, 16, 49–62.

Hartmanis J., 1989, Gödel, von Neumann and the $P =?NP$ problem, *Bulletin of the European Association for Theoretical Computer Science* (EATCS) 38, 101–107.

Heintz, J., M. F. Roy and P. Solerno, 1989, Complexité du principe de Tarski-Seidenberg, *C. R. Acad. Sci. Paris*, Sér. Math. **309**, 825–830.

Hewitt, C., 1971, *Description and theoretical analysis* (*using schemata*) *of PLANNER: a language for proving theorems and manipulating models in a robot*, Ph.D. Thesis, MIT.

Huet, G. D., 1975, A Unification Algorithm for Typed λ-Calculus. *Theoretical Computer Science*. 27–57.

Jensen, D. C. and T. Pietrzykowski, 1976, Mechanizing ω-Order Type Theory through Unification. *Theoretical Computer Science*. 123–171.

Knuth, D. E. and P. B. Bendix, 1970, Simple Word Problems in Universal Algebra. In: *Combinatorial Problems in Abstract Algebras*, ed. by Leech, Pergamon – New York, pp. 263–270.

Kowalski, R., 1975, A Proof Procedure Using Connection Graph. *Journal of the Association for Computing Machines*, 22.

Kowalski, R. and D. Kuehner, 1971, Linear Resolution with Selection Function. *Artificial Intelligence*, 2, 227–260.

Kuehner D. G., 1971, A Note on the Relation between Resolution and Maslov's Inverse Method. In: *Machine Intelligence*, 6, ed. by Meltzer and D. Michie. New York, pp. 73–76.

Livesey M. and J. Siekmann, 1976, Unification of A+C-terms (bags) and A+C+I-terms (Sets). *Universität Karlsruhe, Interner Bericht Nr. 5/76*, Karlsruhe.

Loveland, D. W., 1968, Mechanical Theorem Proving by Model Elimination. *Journal of the Association for Computing Machines*, 15, 236–251.

Loveland D. W., 1969, A Simplified Format for the Model Elimination Procedure. *Journal of the Association for Computing Machines*, 16, 349–363.

Loveland, D. W., 1970, A Linear Format for Resolution. In: *Proc. IRIA Symp. on Automatic Demonstration*, Versailles, France 1968, LNM 125, Berlin – New York: Springer – Verlag, pp. 147–162.

Loveland, D. W., 1972, A Unifying View of some Linear Herbrand Procedures. *Journal of the Association for Computing Machines*, 19, 366–384.

Loveland, D. W., 1978, *Automated Theorem Proving: A Logical Basis*. Amsterdam – New York – Oxford: North-Holland Publ. Comp.

Loveland, D. W., 1984, Automated Theorem-Proving: A Quarter-Century Review. In: *Automated Theorem Proving. After 25 Years*. Eds. W. W. Bledsoe and D. W. Loveland, *Contemporary Mathematics*, 29, AMS, Providence, Rhode Island, pp. 1–46.

Luckham, D., 1970, Refinements in Resolution Theory. In: *Proc. IRIA Symp. on Automatic Demonstration, Versailles*, France 1968, LNM 125, Berlin – New York: Springer-Verlag, pp. 163–190.

Marciszewski W. and R. Murawski, 1995, *Mechanization of Reasoning in a Historical Perspective*, Editions Rodopi, Amsterdam/Atlanta, GA.

Maslov, S. Ju., 1964, An Inverse Method of Establishing Deducibility in Classical Predicate Calculus. *Dokl. Akad. Nauk SSR*. 17–20.

Maslov, S. Ju., 1971, Proof-Search Strategies for Methods of the Resolution Type. *Machine Intelligence*, 6, ed. by B. Meltzer and D. Michie, New York, pp. 77–90.

Meltzer, B., 1966, Theorem-Proving for Computers: Some Results on Resolution and Renaming. *Computer Journal*, 8, 341–343.

Meyer, A. R., 1975, The Inherent Complexity of Theories of Ordered Sets. In: *Proceedings of the International Congress of Mathematicians, Vancouver 1974*, vol. 2, pp. 477–482.

Miller, D. A., E. L. Cohen and P. B. Andrews, 1982, A Look at TPS. In: D. W. Loveland (Ed.), *Proc. Sixth Conf. on Automated Deduction*. Lecture Notes in Computer Science 138. Berlin – Heidelberg – New York: Springer-Verlag, pp. 60–69.

Morris, J. B., 1969, E-resolution: Extension of Resolution to Include the Equality. In: *Proc. Intern. Joint Conf. Artificial Intelligence*. Washington D.C., pp. 287–294.

Mostowski A., 1952, *Sentences Undecidable in Formalized Arithmetic: An Exposition of the Theory of Kurt Gödel*. North-Holland Publishing Company, Amsterdam.

Nerode A. and L. A. Harrington, 1984, The work of Harvey Friedman, *Notices of the American Mathematical Society* 31, 563–566.

Nevins, A. J., 1974, A Human Oriented Logic for Automatic Theorem Proving. *Journal of the Association for Computing Machines*, 21, 606–621.

Nevins, A. J., 1975, Plane Geometry Theorem Proving Using Forward Chaining. *Artificial Intelligence*, 6, 1–23.

Nevins, A. J., 1975a, A Relaxation Approach to Splitting in an Automatic Theorem Prover. *Artificial Intelligence*, 6, 25–39.

Newell, A., J. C. Shaw and H. A. Simon, 1956, Empirical Explorations of the Logic Theory Machine: A Case Study in Heuristics. In: *Proc. Western Joint Computer Conf.*, 15, pp. 218–239. Also in: Feigenbaum and Feldman (Eds.), *Computers and Thought*. McGrow-Hill 1963.

Norton, L. M., 1966, *ADEPT – a Heuristic Program for Proving Theorems of Group Theory*, Ph. D. Thesis, MIT.

Parikh R., 1971, Existence and feasibility in arithmetic, *The Journal of Symbolic Logic* 36, 494–508.

Plotkin, G. D., 1972, Building-in Equational Theories. In: B. Meltzer and D. Michie (Eds.), *Machine Intelligence*, 7, New York, pp. 73–90.

Prawitz, D., 1960, An Improved Proof Procedure. *Theoria*, 26, 102–139.

Presburger, M., 1929, Über die Vollständigkeit eines gewisses Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: *Sprawozdanie z I Kongresu Matematyków Krajów Słowiańskich – Comptus Rendus, I Congres des Math. des Pays Slaves.* Warszawa, pp. 92–101, 395.

Reiter, R., 1971, Two Results on Ordering for Resolution with Merging and Linear Format. *Journal of the Association for Computing Machines*, 18, 630–646.

Robinson, J. A., 1965, A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the Association for Computing Machines*, 12, 23–41.

Robinson, J. A., 1965a, Automated Deduction with Hyperresolution. *International Journal Comput. Math.*, 1, 227–234.

Shostak, R. E., 1976, Refutation Graphs. *Artificial Intelligence*, 7.

Sibert, E. E., 1969, A Machine Oriented Logic Incorporating the Equality Relation. In: B. Meltzer and D. Michie (Eds.), *Machine Intelligence*, 4, New York, pp. 103–134.

Sickel, S., 1976, Interconnectivity Graphs. *IEEE Trans. on Computers, C-25.*

Slagle, J. R., 1967, Automated Theorem Proving with Renamable and Semantic Resolution. *Journal of the Association for Computing Machines*, 14, 687–697.

Statman R., 1978, Bounds for proof-search and speed up in the predicate calculus, *Annals of Mathematical Logic* 15, 225–287.

Stickel, M.E., 1981, A Complete Unification Algorithm for Associative-Commutative Functions. In: *Proc. Fourth Intern. Joint Conf. on Artificial Intelligence*, Tbilisi, USSR. Also in: *Journal of the Ass. for Computing Machines*, 28, 423–434.

Stickel, M. E., 1985, Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, 1, 333–356.

Turing, A. M., 1936–1937, On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. of the London Mathematical Society*, Series 2, vol. 42, pp. 230–265.

Wang, H., 1960, Toward Mechanical Mathematics. *IBM Journal Research and Devel.*, 2–22. Also in: *Logic, Computers and Sets*, Chelsea, New York, 1970.

Wang, H., 1960a, Proving Theorems by Pattern Recognition. Part I. *Commun. Assoc. Comput. Mach.*, 3, 220–234.

Wang, H., 1961, Proving Theorems by Pattern Recognition. Part II, *Bell System Technical Jour.*, 40, 1–41.

Winker, S., L. Wos and E. Lusk, 1981, Semigroups, Antiautomorphisms and Involutions: A Computer Solution to an Open Problem, I. *Math. of Computation*, 533–545.

Wos, L., G. A. Robinson, D. F. Carson and L. Shalla, 1967, The Concept of Demodulation in Theorem Proving. *Journal of the Association for Computing Machines*, 14, 698–709.

Wos, L., D. F. Carson and G. A. Robinson, 1964, The Unit Preference Strategy in Theorem Proving. *AFIPS Conf. Proc. 26*, Washington D.C., pp. 615–621.

Wos, L., G. A. Robinson and D. F. Carson, 1965, Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. *Journal of the Association for Computing Machines*, 12, 536–541.

Wos, L. and G. A. Robinson, 1970, Paramodulation and Set-of-Support. In: *Proc. IRIA Symp. on Automatic Demonstration,* Versailles, France 1968, LNM 125, Berlin – New York: Springer-Verlag, pp. 276–310.

Yates, R., B. Raphael and T. Hart, 1970, Resolution Graphs. *Artificial Intelligence*, 1, 257–289.

Roman Murawski
Adam Mickiewicz University
Faculty of Mathematics and Comp. Sci.
ul. Umultowska 87
61–614 Poznań, Poland
e-mail: rmur@amu.edu.pl