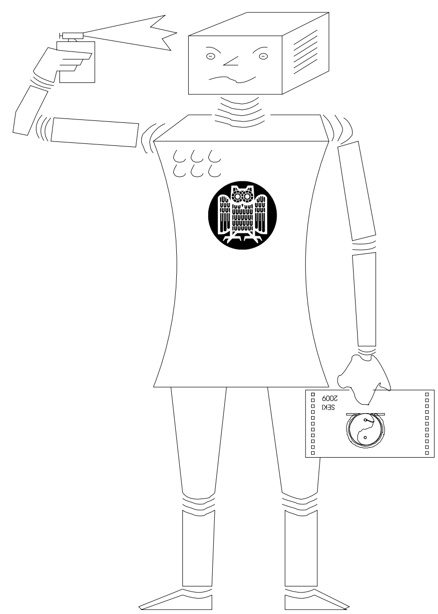
UNIVERSITÄT DES SAARLANDES
FACHRICHTUNG INFORMATIK
D–66123 SAARBRÜCKEN
GERMANY
WWW: http://www.ags.uni-sb.de/

SEKI-Working-Paper ISSN 1860-5931



Progress in Computer-Assisted Inductive Theorem Proving by Human-Orientedness and *Descente Infinie*?

CLAUS-PETER WIRTH
Dept. of Computer Sci., Saarland Univ.,
D–66123 Saarbrücken, Germany
wirth@logic.at

SEKI-Working-Paper SWP-2006-01

This SEKI-Working-Paper was internally reviewed by:

DIETER HUTTER
DFKI GmbH, Stuhlsatzenhausweg 3, D–66123 Saarbrücken, Germany
E-mail: hutter@dfki.de
WWW: http://www.dfki.de/~hutter

PETER PADAWITZ
Informatik, Universität Dortmund, D–44221 Dortmund, Germany
E-mail: peter.padawitz@udo.edu
WWW: http://fldit-www.cs.uni-dortmund.de/~peter

TOBIAS SCHMIDT-SAMOA
Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, D–67653 Kaiserslautern, Germany
E-mail: t.samoa@netcologne.de
WWW: http://agent.informatik.uni-kl.de/mitarbeiter/schmidt-samoa

Editor of SEKI series:

Claus-Peter Wirth
Brandenburger Str. 42, D–65582 Diez, Germany
E-mail: wirth@logic.at
WWW: http://www.ags.uni-sb.de/~cp

Progress in Computer-Assisted Inductive Theorem Proving by Human-Orientedness and *Descente Infinie*?

CLAUS-PETER WIRTH
Dept. of Computer Sci., Saarland Univ., D–66123 Saarbrücken, Germany
wirth@logic.at

Searchable Online Edition
Submitted March 21, 2006
First Print Edition August 25, 2006
Very Minor Improvements July 16, 2008
Very Minor Improvements May 11, 2010
Minor Improvements Aug. 31, 2010

Abstract

In this position paper we briefly review the development history of *automated inductive theorem proving* and *computer-assisted mathematical induction*. We think that the current low expectations on progress in this field result from a faulty narrow-scope historical projection. Our main motivation is to explain — on an abstract but hopefully sufficiently descriptive level — why we believe that future progress in the field is to result from human-orientedness and *descente infinie*.

Contents

1 Introduction

1.1 Subject Area

In this paper we are concerned with

- *automated inductive theorem proving* and
- *computer-assisted mathematical induction*.

Both terms refer to the task of doing mathematical induction with the computer. The former term puts emphasis on the importance of strong automation support, as found in the classical systems NQTHM [15, 16], INKA [7], and ACL2 [56] based on explicit induction. The latter and more general term, however, is to denote more human-oriented approaches in addition, as found in QUODLIBET [10] and other future systems based on *descente infinie*. Note that we do not believe in the usefulness of the extreme representatives of any of the two terms: Neither mere black-box automation nor mere proof-checkers can be too useful in mathematical induction. Above that, we think that a successful system has to put strong emphasis on both aspects and find a way to be both human- *and* machine-oriented.

1.2 Expectations and Importance of Future Progress

A majority of researchers in the area of computer-assisted mathematical induction seem to believe that no further progress can be expected in this area within the nearer future. Moreover, recently, between two talks at a conference, one of the leading German senior researchers in the field told me that he thinks that currently it is hardly possible to get any funding for research on computer-assisted mathematical induction.

Thus, we should ask for possible scientific reasons for the current funding situation. We ought to check the justification of the belief that progress in computer-assisted mathematical induction is unlikely to occur in the nearer future.

It is, however, obviously not the case that progress in computer-assisted mathematical induction is considered to be unimportant. Indeed, progress in computer-assisted mathematical induction is in high demand for mathematics assistance systems, for verification of software and hardware, and for synthesis of recursive programs. Due to a slow-down in progress of automated mathematical induction in the last decade, however, currently there does not seem to be much hope among scientists for further progress in the nearer future.

1.3 A Possible Way to Future Progress — Overall Thesis

To show a possible way to future progress is the aim of this position paper. Namely, to explain why we are confident that *descente infinie* can initiate a further breakthrough in computer-assisted mathematical induction.

Together with *descente infinie* we present our ideas on the importance of *human-oriented theorem proving*, a point of view we have been holding and furthering for more than a dozen years [110, 121].

“Human-oriented theorem proving” basically means that — to overcome the current stagnation — we have to develop *paradigms and systems* for the synergetic combination and cooperation of the human mathematician with its semantical strength and the machine with its computational strength.

Our thesis is that *descente infinie* is such a paradigm.

1.4 Organization of this Paper

The paper organizes as follows. In § 2 we describe the general context where and why mathematics and mathematicians should win from computers. As the reason for the little hope in progress in mathematical induction seems to be a wrong projection from the past into the future, we cannot reasonably state what we may hope to achieve by *human-orientedness* and *descente infinie* (§ 4) and why the two belong together (§ 5) before we have had a short look at the history of computer-assisted theorem proving in § 3. Without diving too deep into technical details, after presenting *descente infinie* (§ 6) and *explicit induction* (§§ 7 and 8), we then support our overall thesis of § 1.3 in §§ 9 and 10, and discuss the standard objections in § 11. Finally, we conclude in § 12.

2 Requirements Specification

From the ancient Greeks until today, mathematical theories, notions, and proofs are not developed the way they are documented. This difference is not only due to the iterative deepening of the development and the omission of easily reconstructible parts. Also the global order of presentation in publication more often than not differs from the order of development. This results in the famous *eureka* steps, which puzzle the freshmen in mathematics. The difference does not only occur in scientific publications where the *succinct presentation of results* may justify this difference, but also for the vast majority of textbooks and lectures where the objective should be

- to *teach how to find* proofs, notions, and theorems.

The conventional natural-language representation of mathematical proofs in advanced theoretical journals with its intentional vagueness [115, § 6.2] and hidden sophistication can only inform highly educated human beings about already found proofs. This conventional representation, however — as fascinating as it is as a summit of the ability of the human race to communicate deep structural knowledge effectively — does not tell much about the

- originally applied plans and methods of *proof construction*

and does not admit computers

- to *check for soundness* and
- to *take over the tedious, error-prone, computational, and boring parts* of proofs.

Obviously, a computer representation that admits the flexibility for and the support of the issues of all above items in parallel plus the computation of

- different conventional *natural language presentations* tailored to various purposes

is in great demand and could increase the efficiency of working mathematicians tremendously.

3 Short History of Computer-Assisted Theorem Proving

3.1 Formula Language and Calculi

Starting with the Cossists and VIÈTE in the 15th and 16th centuries, the formula language of mathematics and its semantics were adequately and rigorously modeled by the end of the 19th century in PEANO's ideography [78] and FREGE's Begriffsschrift [37].

An adequate rigorous representation that supports a working mathematician's *theorem proving*, however, has not been found until today. But already now the formula language of mathematics and its semantics can provide a powerful interface between human and machine.[1]

The numerous logic calculi developed during the 20th century were mostly designed to satisfy merely theoretical criteria, but not to follow the theorem-proving procedures of working mathematicians.

An important step toward human-oriented calculi was done by GERHARD GENTZEN (1909–1945) when he used his structural insights to refine his Natural Deduction calculi (which were close to natural-language mathematics) into sequent calculi [40]. These calculi meant a huge progress toward an adequate human-oriented representation of a working mathematician's deductive proof search. Sequent and tableau calculi capture the reductive (analytic, top-down, backward) reasoning from goals to subgoals directly in the essential calculus rules and the generative (synthetic, bottom-up, forward) reasoning from axioms to lemmas can be adequately realized with lemmatizing versions of the Cut rule [9, 10, 110, 113] (cf. Note 10). Based on GENTZEN's sequent calculus there has been further progress into this direction: Free-variable calculi [35, 71, 113] admit to defer commitments until the state of the proof attempt provides sufficient information for a successful choice. Thereby they help the mathematician to follow his proof plans more closely by overcoming premature witness decisions forced by GENTZEN's original calculus. Indexed formula trees [6] admit the mathematician to focus immediately on the crucial proofs steps and defer the problems of β-sequencing and γ-multiplicity [115].

3.2 Automation

Starting in the 1950s, there was great hope to automate theorem proving with the help of computers and machine-oriented logic calculi. State-of-the-art fully-automated theorem provers of today (*ATP*s, such as VAMPIRE [84] and WALDMEISTER [17, 67]) represent a summit in the history of creative engineering. That ATP systems will never develop into systems that can assist a mathematician in his daily work, however, is a general consensus among their developers for more than a dozen years now. The reason for this is the following:

> *The automatic theorem provers' search spaces are too huge for complete automation and completely different from the search spaces of the working mathematicians, who therefore can neither interact with these systems, nor transfer their human skills to them.*

Note that this does not mean that ATP systems are useless. They already now provide a powerful basis for the automation in mathematics assistance systems such as ΩMEGA [93].

3.3 Proof Planning

At the end of the 1980s, the ideas to overcome the approaching dead end in ATP were summarized under the keyword *proof planning*. Beside its human-science aspects [19], the idea of proof planning [18, 28] is to add smaller and more human-oriented *higher-level search spaces* to the theorem-proving systems on top of the *low level search spaces* of the logic calculi. In the 1990s, the major proof-planning systems OYSTER-CLAM [18, 22], ΩMEGA [93], and λCLAM [20] seem to have been led astray by the hopes that with these additional levels

1. the underlying logic calculus could be neglected,[2] and,

2. instead of the working mathematician himself, it would be sufficient to get his proof plans to the machine.

3.4 Alternative Point of View in Proof Planning

To the contrary of these hopes, we believe that progress in proof planning and computer-assisted mathematical theorem proving requires the further development of human-oriented state-of-the-art logic calculi, which free the higher levels from unnecessary low-level commitments and admit the mathematician to interact directly with the machine, even when the automation of proofs fails on the lowest logic level.

> *We need both high-level top-down interactive proof development and bottom-up support from a state-of-the-art flexible human-oriented calculus with strong automation.*

The neglect of the logic calculus and human–machine interaction is to be overcome in the system ISAPLANNER [28, 29, 30] and in the new ΩMEGA system currently under construction [9] by using the standard calculus of ISABELLE/HOL [70, 71] and the new human-oriented calculus of CORE [6], respectively.

3.5 Conclusion: Human-Oriented Automated Theorem Proving

The completely automatic generation of a non-trivial proof for a given input conjecture is typically not possible today and — contrary to the complete automation of chess playing — will probably never be.

Thus, beside some rare exceptions — as the automation of proof search will always fail on the lowest logic level from time to time — the only chance for automatic theorem proving to become useful for mathematicians is *a synergetic interplay between the mathematician and the machine*.

For this interplay, it does not suffice to compute human-oriented representations of machine-oriented proof attempts for interaction with a user interface during the proof search. Indeed, experience shows that the syntactical problems have to be presented accurately and in their exact form. Thus — to give the human user a chance to interact — the calculus *itself* must be *human-oriented*.

4 What Can we Hope to Achieve? And How?

After all that history of great original expectations and down-slowing progress, what can we reasonably hope for the nearer future?

As described in § 3.1 and Note 1, the formula language of mathematics and its semantics already now provides a powerful interface between human and machine. But we still have to find a representation of mathematical proofs supporting the issues mentioned in § 2, namely: machine assistance in and teaching of proof search, proof planning, and theory development; automation of tedious, error-prone, computational, and boring parts of proofs and checking for soundness; and the computation of various natural language presentations.

As full automation cannot succeed within the current paradigm, we have to follow the human mathematicians, although we do not know much about their procedures and they hardly know how to explain them.[3]

The first steps on this way are to give the mathematician the freedom to go his way and let the system assist him. Not the other way round as usual! We are convinced of a potential success of the following development cycle:

- In a first step, informal and formal logical calculi and the user interfaces have to provide the freedom to use all the required means in a human-oriented design, and then,
- in a second step, we have to learn the heuristics that admit a feasible proof search from the mathematicians; by human learning in the beginning, hopefully by artificial-intelligence machine-learning later.

And the starting point ought to be a human-oriented, machine-oriented, flexible state-of-the-art calculus [6, 113] and an administration of proof tasks in a proof data structure [9].

5 Why Mathematical Induction?

In this § 5, we briefly explain why we see an affinity between human-orientedness and mathematical induction and why this position paper is about both *descente infinie* and human-orientedness in parallel.

Beside some proof-theoretical peculiarities of mathematical induction that do not really have a practical effect,[4] mathematical induction is the area of mathematical theorem proving where our heuristic knowledge is best. This is the case both for human (*descente infinie*, cf. § 6) and for machine-oriented heuristics (*explicit induction*, cf. § 7). As these two heuristics are completely different in their surface structure and the progress in practical usefulness was quite moderate in the last decade, mathematical induction is a good area to look for evidence for our thesis on *human-orientedness*:

> *Human-oriented procedures can overcome the current slowdown of progress in computer-assisted theorem proving. Their — even compared to machine-oriented procedures — huge search spaces can be controlled by heuristics learned from human mathematicians working with advanced systems.*

6 *Descente Infinie*

In everyday mathematical practice of an advanced theoretical journal the frequent inductive arguments are hardly ever carried out explicitly. Instead, the proof just reads something like "by structural induction on n, q.e.d." or "by induction on (x, y) over $<$, q.e.d.", expecting that the mathematically educated reader could easily expand the proof if in doubt. In contrast, very difficult inductive arguments, sometimes covering several pages, such as the proofs of HILBERT's *first ε-theorem* [48, Vol. II] or GENTZEN's *Hauptsatz* [40], or confluence theorems such as the ones in [47, 109, 119] still require considerable ingenuity and *will* be carried out! The experienced mathematician engineers his proof roughly according to the following pattern:

> He starts with the conjecture and simplifies it by case analysis. When he realizes that the current goal becomes similar to an instance of the conjecture, he applies the instantiated conjecture just like a lemma, but keeps in mind that he has actually applied an induction hypothesis. Finally, he searches for some well-founded ordering in which all the instances of the conjecture he has applied as induction hypotheses are smaller than the original conjecture.

The hard tasks of proof by mathematical induction are

(Hypotheses Task)
to find the numerous induction hypotheses (as, e.g., in the proof of GENTZEN's Hauptsatz on Cut-elimination) and

(Induction-Ordering Task)
to construct an *induction ordering* for the proof, i.e. a well-founded ordering that satisfies the ordering constraints of all these induction hypotheses in parallel. (For instance, this was the hard part in the elimination of the ε-formulas in the proof of the 1st ε-theorem in [48, Vol. II], and in the proof of the consistency of arithmetic by the ε-substitution method in [2]).

The soundness of the above method for engineering hard induction proofs is easily seen when the argument is structured as a proof by contradiction, assuming a counterexample. For PIERRE FERMAT's (1607?–1665) historic reinvention of the method, it is thus just natural that he developed the method itself in terms of assumed counterexamples [12, 26, 33, 68, 116]. He called it "*descente infinie ou indéfinie*". Here it is in modern language, very roughly speaking: A proposition Γ can be proved by *descente infinie* as follows:

> *Show that for each assumed counterexample of Γ there is a smaller counterexample of Γ w.r.t. a well-founded ordering $<$, which does not depend on the counterexamples.*

There is historic evidence on *descente infinie* being the standard induction method in mathematics: The first known occurrence of *descente infinie* in history seems to be the proof of the irrationality of the golden number $\frac{1}{2}(1+\sqrt{5})$ by the Pythagorean mathematician HIPPASUS OF METAPONTUM (Italy) in the middle of the 5th century B.C. [38]. Moreover, we find many occurrences of *descente infinie* in the famous collection “Elements” of EUCLID OF ALEXANDRIA [32]. The following eighteen centuries showed a comparatively low level of creativity in mathematical theorem proving, but after FERMAT's reinvention of the Method of *Descente Infinie* in the middle of the 17th century, it remained the standard induction method of working mathematicians until today.

At FERMAT's time, natural language was still the predominant tool for expressing terms and equations in mathematical writing, and it was too early for a formal axiomatization. Moreover, note that an axiomatization captures only validity, but in general does neither induce a method of proof search nor provide the data structures required to admit both a formal treatment and a human-oriented proof search. The formalizable logic part, however, of *descente infinie* can be expressed in what is called the (second-order) *Theorem of* NOETHER*ian Induction* (N), after EMMY NOETHER (1882–1935). This is not to be confused with the *Axiom of Structural Induction*, which is generically given for any inductively defined data structure, such as the *Axiom* (S) *of Structural Induction for the natural numbers inductively defined by the constructors zero* $\mathsf{0}$ *and successor* s. Moreover, we need the definition $(\mathsf{Wellf}(<))$ of well-foundedness of a relation $<$.

$$(\mathsf{Wellf}(<)) \qquad \forall Q.\ \Big(\ \exists x.\ Q(x) \quad\Rightarrow\quad \exists m.\ \big(\ Q(m) \wedge \neg\exists w{<}m.\ Q(w)\ \big)\ \Big)$$

$$(\mathsf{N}) \qquad \forall P.\ \left(\ \forall x.\ P(x) \quad\Leftarrow\quad \exists{<}.\ \left(\begin{array}{ll} & \forall v.\ \big(\ P(v) \ \Leftarrow\ \forall u{<}v.\ P(u)\ \big) \\ \wedge & \mathsf{Wellf}(<) \end{array}\right)\right)$$

$$(\mathsf{S}) \qquad \forall P.\ \Big(\ \forall x.\ P(x) \quad\Leftarrow\quad P(\mathsf{0})\ \wedge\ \forall y.\ \big(\ P(\mathsf{s}(y)) \Leftarrow P(y)\ \big)\ \Big)$$

$$(\mathsf{nat1}) \qquad \forall x.\ \big(\ x = \mathsf{0}\ \vee\ \exists y.\ x = \mathsf{s}(y)\ \big)$$

$$(\mathsf{nat2}) \qquad \forall x.\ \ \mathsf{s}(x) \neq \mathsf{0}$$

$$(\mathsf{nat3}) \qquad \forall x, y.\ \big(\ \mathsf{s}(x) = \mathsf{s}(y)\ \Rightarrow\ x = y\ \big)$$

Let $\mathsf{Wellf}(\mathsf{s})$ denote $\mathsf{Wellf}(\lambda x, y.\,(\mathsf{s}(x) = y))$, which implies the well-foundedness of the ordering of the natural numbers. The natural numbers can be specified up to isomorphism either by (S), $(\mathsf{nat2})$, and $(\mathsf{nat3})$, or else by $\mathsf{Wellf}(\mathsf{s})$ and $(\mathsf{nat1})$. The first alternative follows RICHARD DEDEKIND (1831–1916) and is named after GUISEPPE PEANO (1858–1932). The second follows MARIO PIERI (1860–1913). As the instances for P and $<$ in (N) are often still easy to find when the instances for P in (S) are not, the second alternative together with (N) is to be preferred in theorem proving for its usefulness and elegance. Cf. [113] for more on this.

For a more detailed discussion of *descente infinie* from the historical and linguistic points of view see [116, § 2].

7 Explicit Induction

In the 1970s, the *School of Explicit Induction* was formed by computer scientists working on the automation of inductive theorem proving. Inspired by J. ALAN ROBINSON's resolution method [85], they tried to solve problems of logical inference via reduction to machine-oriented inference systems. Instead of implementing more advanced mathematical induction techniques, they decided to restrict the second-order Theorem of NOETHERian Induction (N) (cf. § 6) and the inductive Method of *Descente Infinie* to first-order *induction axioms* and deductive first-order reasoning [113, § 1.1.3].

Note that in these induction axioms, the subformula

$$\forall u{<}v.\ P(u)$$

of (N) is replaced with a conjunction of instances of $P(u)$ with predecessors of v like in (S). The induction axioms of explicit induction must not contain the induction ordering $<$.

Furthermore, note that although an induction axiom may take the form of a first-order instance of the second-order Axiom of Structural Induction (S) (cf. § 6), conceptually it is an instance of (N) and the whole concept of *explicit induction* is a child of the computer, whereas (S) was already applied by the ancient Greeks [1].

The so-called "waterfall"-method of the pioneers of this approach [15] refines this process into a fascinating heuristic, and the powerful inductive theorem proving system NQTHM [15, 16] has shown the success of this reduction approach already in the 1970s. For comprehensive surveys on explicit induction cf. [105] and [20]. Cf. [114] for a survey on the alternative approaches of implicit and inductionless induction.[5]

BOYER & MOORE's NQTHM [15, 16] and BUNDY & HUTTER's rippling[6] [13, 23, 24, 49, 50, 52, 95] are prime examples of practically useful automation-supported theorem proving and proof planning, respectively. Mainly associated with the development of explicit induction systems such as OYSTER-CLAM [18, 22], λCLAM [20], and INKA [7], there was still evidence for considerable improvements over the years until the end of the 20th century [53]. Since then, explicit induction has become a standard in education in the ✓ERIFUN project [106]. Today, the application-oriented explicit induction system ACL2 [56] is still undergoing some minor improvements. ACL2 easily outperforms even a good mathematician on the typical inductive proof tasks that arise in his daily work or as subtasks in software verification. These methods and systems, however, do not seem to scale up to hard mathematical problems and *program synthesis* (where the computer-assisted inductive proof of a property of an underspecified program actually is to synthesize the recursive definitions of the program). We believe that there are *principled reasons* for this shortcoming.

8 Why Sticking to Explicit Induction Blocks Progress

8.1 Flow of Information

Apart from sociological reasons,[7] explicit induction blocks progress because it does not admit a *natural flow of information* in the sense that a decision can be delayed or a commitment deferred until the state of the proof attempt provides sufficient information for a successful choice. Indeed, explicit induction unfortunately must solve the two hard tasks mentioned in § 6 (namely the Hypotheses Task and the Induction-Ordering Task) already *before* the proof has actually started. A proper induction axiom must be generated without any information on the structural difficulties that may arise in the proof later on. For this reason, it is hard for an explicit-induction procedure to guess the right induction axioms for difficult proofs in advance.

8.2 Recursion Analysis and Induction Variables

One of the most developed and fascinating applications of heuristic knowledge found in artificial intelligence, informatics, and computer science is *recursion analysis* [15]. This is a technique for guessing a proper induction axiom by static analysis of the syntax of the conjecture and the recursive definitions. In this paper, we subsume under the notion of "classical recursion analysis" also its minor improvements [98, 99, 102, 103, 104]. Under the notion of "recursion analysis" we also subsume *ripple analysis*, an important extension of classical recursion analysis.

Ripple analysis is sketched already in [21, § 7] and nicely described in [20, § 7.10]. On the one hand, by rejecting recursive definitions whose unfolding would block the application of the induction hypothesis, ripple analysis excludes some unpromising induction axioms of classical recursion analysis. On the other hand, by considering lemmas of a reductive character in addition to the actual recursive definitions, ripple analysis can find more useful induction axioms than classical recursion analysis.

A requirement, however, which we put on the notion of "recursion analysis" is that it does not perform dynamical proof search but has a limited lookahead into the proof, typically one rewrite step for each term in a set of subterms that covers all occurrences of *induction variables*. Note that although "induction variable" is a technical term in recursion analysis, roughly speaking, this notion is also common among working mathematicians when they say that something is shown "by induction on y", for a variable y, for instance.

8.3 The Hypotheses Problem

However fascinating and highly developed recursion analysis may get, even the disciples of the School of Explicit Induction admit the inherent limitations of explicit induction: In [81, p. 43], we find not only small verification examples already showing these limits, but also the conclusion:

Problem 8.1 ([81, p. 43]) "We claim that computing the hypotheses *before* the proof is not a solution to the problem and so the central idea for the lazy method is to postpone the generation of hypotheses until it is evident which hypotheses are required for the proof."

This "lazy method" removes only some limitations of explicit induction as compared to *descente infinie*. It focuses more on efficiency than on a clear separation of concepts, and there is no implementation of it available anymore. The labels "lazy induction" and "lazy hypotheses generation" that were coined in this context are nothing but a reinvention of parts of FERMAT's *descente infinie* by the explicit-induction community.

8.4 The Induction-Ordering Problems

Computer scientists from the School of Explicit Induction used to consider the tasks of

- induction (i.e. the choice of an induction axiom; e.g. by recursion analysis) and
- deduction (i.e. the rest of the proof; e.g. by standard first-order reasoning techniques or by rippling [13, 23, 24, 49, 50, 52, 95])

to be orthogonal. Working mathematicians know that this is wrong. Especially the choice of a proper induction ordering interacts with the several cases of a proof in such a way that a new proof idea tends to be in conflict with the induction ordering of the previous cases.

- On the one hand, it is standard in explicit induction to fix induction orderings eagerly, at the very beginning of a proof.
- On the other hand, fixing an induction ordering earlier than in the last steps of an induction proof has hardly any benefit ever:
  - For *difficult* proofs, this is obvious to any working mathematician.
  - For *simple* proofs, the simple fact that any equation has a left- and a right-hand side provides us with sufficient pragmatics for searching in that area of the search space where the smaller induction hypotheses use to be applicable; provided that the specifier has written his specifications in the standard style and the user has activated his lemmas for rewriting with a suitable orientation.

Problem 8.2 Explicit induction has to commit to a fixed and unchangeable induction ordering eagerly, at the very beginning of an induction proof. Such a commitment comes far too early and is a typical cause of failure. Moreover, it is superfluous because there is hardly any heuristic benefit of committing to an induction ordering earlier than in the last steps of an induction proof.

Beside the restriction of explicit induction to enforce an *eager* computation of induction axioms (i.e. induction hypotheses and the related induction orderings), explicit induction by recursion analysis has also another limitation:

Problem 8.3 Computing induction axioms by recursion analysis can only result in such induction orderings that are recombinations of orderings resulting from the recursive definitions (and from the currently known lemmas of a reductive character) of the related specifications.

As a matter of fact, most of the non-trivial induction proofs do not work out with such induction orderings. Moreover, for the case of program synthesis, we do not want to be restricted to such induction orderings. For an instance of this see [25], where the quick-sort algorithm is to be synthesized from the requirements specification of the sorting function.

9 Why *Descente Infinie* is Promising Now

The theoretical research paper [113] provides us with the *integration* of *descente infinie* into deductive calculi. It is — to the best of our knowledge — the first such combination in the history of logic, which does *not encode* induction, but actually *models* the mathematical process of proof search by *descente infinie itself* and *directly* supports it with the data structures required for a formal treatment.[8]

This integration (presented for state-of-the-art free-variable sequent and tableau calculi) is well-suited for an efficient interplay of human interaction and automation and combines raising [69], explicit representation of dependence between free γ- and δ-variables (according to SMULLYAN's classification [96]), the liberalized δ-rule, preservation of solutions, and unrestricted applicability of lemmas and induction hypotheses. Moreover, the integration is natural in the sense that it goes together well with context-improved reasoning as in [6], with modern proof data structures as in [9], with program synthesis as in [25], and with logical binders such as λ and ε [42, 48, 65, 112, 118]. The semantical requirements for the integration are satisfied for practically all[9] two-valued logics, such as clausal logic, classical first-order logic, and higher-order modal logic [113, Note 8].

When computer-assisted inductive theorem proving started in the early 1970s, the induction axioms of explicit induction were the only known feasible formal means to integrate induction into deductive calculi. Today, however, we are in a better situation because the results of [113] provide us with a simple, elegant, and both machine- and human-oriented integration of *descente infinie itself*.

The only overhead this integration requires is to add a weight term to each sequent or proof *goal*. These weight terms stay inactive until a goal is applied as an induction hypothesis. Compared to the application of a goal as lemma, such an induction-hypothesis application produces an additional ordering subgoal, which asks us to show that the induction hypothesis is smaller than the goal to which it is applied in some well-founded ordering.

On a more technical level — to integrate *descente infinie* into a given logic calculus — we need

1. to augment the goals (sequents) of the calculus with weight terms,
2. to add a lemma application[10] to the calculus if not already present,
3. to patch the lemma application of the calculus to admit induction-hypothesis application, which generates an additional ordering subgoal for soundness based on the weights of induction hypothesis (lemma) and goal, and
4. to solve the ordering constraints of the induction-hypothesis applications.

Typically, these requirements are easily satisfied, although there may be problems with calculi based on fixed logical frameworks.[11]

10 The Fundamental Practical Advantage of *Descente Infinie*

For human mathematicians, non-trivial mathematical proofs appear to have a *semantical* nature. Therefore, mathematics assistance systems should comply with natural human proof techniques and should be able to follow the exact order in which the human user organizes his semantical problem solving.

Automated theorem proving, however, works on *syntactical* domains. These syntactical domains are different from the semantical ones. Typically, they admit neither a global view on the proof task nor the realization of a false commitment. For instance, if recursion analysis results in a useless induction axiom, the proof attempt fails completely.[12] Of course, this does not mean that automatic search in a calculus is useless. To the contrary, an anytime and sparse automated syntactic search through the semantically highly redundant search spaces of a logic calculus is most helpful in parallel to the interaction of the human user.

In such a parallel approach, the "hot" constraints should always be solved first. With the term "*hot* constraints", we mean constraints with solutions that are strongly indicated by the current state of the proof attempt in the sense that there is a committing step toward their solution that makes a success of the proof attempt more likely or without that the proof can hardly succeed. Although those constraints that are hot for a mechanic procedure and those that are hot for a human mathematician in the construction of the proof idea will be different more often than not, man and machine can cooperate very well, provided that the constraints can be solved in any intended order and the effects can be communicated on the basis of a common view. Note that a step from either side will typically change the set of hot constraints of the other.

We are very well aware of the fundamental difficulties and open questions that have to be solved for such a cooperation of man and machine. It actually cannot be denied that there seem to be several divergences between man and machine, especially:

- Automation prefers fully expanded definitions while the human user prefers a concise representation with composite notions.
- The higher the automatization the more difficult the analysis of a failed proof attempt for the human user.

Nevertheless, we are convinced that a cooperation of man and machine on the basis of a common view is a realistic goal.

Now we finally just have to mention the fundamental practical advantage of *descente infinie* as compared to encodings of some form of induction. (Such an encoding can be found, e.g., in the induction rule of [41], or by application of the second-order Theorem of NOETHERian Induction (N) (cf. § 6), or the second-order Axiom of Structural Induction (S) (cf. § 6), or by generation of first-order induction axioms.):

> *The fundamental practical advantage of our integration of* descente infinie *is that the constraints of the inductive proof search can now be solved together with all other constraints of the whole deduction in any suitable order.*

Thus, if recursion analysis shows us the proper way, we can solve the constraints in the order according to the heuristics of explicit induction. But any other order is also possible. And we may delay solving the harder constraints until the state of the proof attempt provides us with information sufficient for a successful choice.

11 Discussion

11.1 Paradigm Shift without Sacrifice — Really?

In blank opposition to our evaluation of *descente infinie* in § 9 as promising, in the 1990s and still in the beginning of the 21st century, some leading scientists from the explicit-induction community used to claim

(1) that *descente infinie* would be too complicated to be useful in practice, and

(2) that the proper induction axioms could be computed before the actual proof search by a partial inspection of the proof in a specialized presentation different from the actual proof search with some advanced artificial-intelligence techniques [51].

Claim (1) has already been falsified by the successful treatment of *descente infinie* in the theorem prover QUODLIBET [8, 10, 61, 62, 66, 88, 89, 90, 91, 109, 110, 113, 119]. Although QUODLIBET does not use any induction axioms, it is competitive with the leading inductive theorem prover ACL2 [56], with the practically important exception that ACL2 is so efficiently implemented that it can be used for both verification *and testing* of software.

We believe that also Claim (2) is wrong and that we need the freedom to solve the two hard tasks mentioned in § 6 (namely the Hypotheses Task and the Induction-Ordering Task) in small portions spread over the whole search of the actual proof. This belief was also confessed to by others in [58, § 4.5] and in [45, § 13.4], and there is further recent evidence for this in [91, § 8]. Even if Claim (2) were right and the proposed procedure feasible, it would still be an uneconomic procedure because there is no need to plan the induction axiom with specialized tools based on a special additional representation before searching for the actual proof.

The deeper reasons behind the Claims (1) and (2) seem to be conservatism and the fear that the heritage of the great heuristic contributions to inductive theorem proving developed within the paradigm of explicit induction could be lost. Although such losses are typical for paradigm shifts [63, 117], the fear seems to be completely unjustified in our case:

- Theoretically, *descente infinie* includes explicit induction.
- Practically, QUODLIBET has shown that in our framework of *descente infinie*, the heuristic knowledge of *recursion analysis* in the field of explicit induction is still applicable, indispensable, and at least as useful as before. We will explain this in § 11.3. That also rippling probably stays as important as before is sketched in § 11.4.

Eng ist die Welt, und das Gehirn ist weit.
Leicht beieinander wohnen die Gedanken,
Doch hart im Raume stoßen sich die Sachen.

— FRIEDRICH VON SCHILLER; WALLENSTEINS TOD,
2. AUFZUG, 2. AUFTRITT; WALLENSTEIN

11.2 Schism in Minds vs. Schism in Systems

Actually, the schism between explicit induction on the one side and *descente infinie* on the other, never really existed in the minds of most of the leading scientists of the field, especially not since the year 1996 [114, § 4.2]. This expertise, however, has neither been published nor communicated to the outside of the inner circle. Moreover — contrary to the flexibility of the minds — in the powerful inductive theorem prover ACL2 [56] and most other such systems this schism is still manifest:

Problem 11.1 (No Natural Flow of Information in ACL2)
The only way to get ACL2 to use an induction ordering which is not of the kind of Problem 8.3 is to add a recursive function f terminating over this ordering and to hint the prover to use the ordering of its termination proof for the eager generation of a *eureka* induction axiom. Note that the function f is typically nonsense and will be used nowhere and especially not in the theorem, so that the hint to use it is really necessary.

11.3 The Rôle of Recursion Analysis in *Descente Infinie*

The cases where eager induction-hypotheses generation is needed to guide the proof into the right direction (cf. e.g. [113, § 3.3]) are so rare in practice that the current standard induction heuristic of the *descente infinie* system QUODLIBET [10, 88] generates induction hypotheses only lazily, whereas the case splits for the induction variables are done eagerly right at the beginning (after simplification). The possibility to be lazy even simplifies recursion analysis when different induction schemes are in conflict because we do not have to merge them completely: Compare [61, § 8.3] with the complicated problems of [103, 104]!

Nevertheless, recursion analysis plays an important rôle also in QUODLIBET and in *descente infinie* in general. Even without generating induction hypotheses and the induction ordering eagerly, the case analysis suggested by recursion analysis is of great heuristic value. Indeed, nothing is more helpful than to know how to start the proof of a conjecture (after simplification).

The recursion analysis in *descente infinie* is most useful for solving the following task of case analysis:

(Task of Case Analysis on Induction Variables)

Which outermost universal variables of the (simplified) conjecture to are to be used as induction variables, and which lemmas are to be used for the case analysis on the structure of the induction variables?

For instance, which lemmas of the following form are to be chosen for our induction variables $m : \mathsf{nat}$ and $l : \mathsf{list}(\mathsf{nat})$ for a natural number and a list of natural numbers, respectively?

$$m = \mathsf{0} \;\;\vee\;\; \exists n : \mathsf{nat}.\ (m = \mathsf{s}(n))$$

$$m = \mathsf{0} \;\;\vee\;\; \exists p : \mathsf{list}(\mathsf{nat}).\ \left(\begin{array}{rl} & m = \prod p \\ \wedge & \mathsf{Every}(\mathsf{Prime}, p) \end{array}\right)$$

$$l = \mathsf{nil} \;\;\vee\;\; \exists n : \mathsf{nat}.\ \exists k : \mathsf{list}(\mathsf{nat}).\ (l = \mathsf{cons}(n, k))$$

$$l = \mathsf{nil} \;\;\vee\;\; \exists n : \mathsf{nat}.\ \exists k : \mathsf{list}(\mathsf{nat}).\ (l = \mathsf{append}(k, \mathsf{cons}(n, \mathsf{nil})))$$

Note that this task is most critical for explicit induction, because the eager induction-hypotheses generation fixes the result of this case analysis and makes a later adjustment impossible. In *descente infinie*, however, this task is non-critical because it serves only as a heuristic hint on how to start proof search. This is shown in the following example.

Example 11.2

Consider the toy example of the even-predicate on natural numbers in the clause

$$\mathsf{Even}(x+z),\ \neg\mathsf{Even}(x+y),\ \neg\mathsf{Even}(y+z).$$

When recursion analysis based on $\mathsf{s}(v)+w = \mathsf{s}(v+w)$ suggests a base case of $x{=}\mathsf{0}$ and a step case of $x{=}\mathsf{s}(x')$, then a proof attempt by explicit induction fails.

A proof attempt by *descente infinie*, however, can go on with a second case distinction on $x'{=}\mathsf{0}$ and $x'{=}\mathsf{s}(x'')$ and actually proceed by the two base cases of $x{=}\mathsf{0}$ and $x{=}\mathsf{s}(\mathsf{0})$ and a step case of $x{=}\mathsf{s}(\mathsf{s}(x''))$. This, however, is not possible for explicit induction based on any form of recursion analysis. Note that the three cases of $y{=}\mathsf{0}$, $z{=}\mathsf{0}$, and $y{=}\mathsf{s}(y') \wedge z{=}\mathsf{s}(z')$ provide yet another way for *descente infinie* to extend the proof attempt into a successful proof.

11.4 Rippling and *Descente Infinie*

Although QUODLIBET does not implement *rippling* [13, 23, 24, 49, 50, 52, 95] yet (but applies a less syntactically restricted search by a refined contextual rewriting with markings [90, 91]), we expect that the restrictions of the search space introduced by rippling can be more useful in the less restrictive framework of *descente infinie* than in the more restrictive framework of explicit induction.

When induction hypotheses are not generated eagerly, "creational rippling" [23] or "blowing up of terms" [51, 52] are not required. Instead, the induction variables occur as additional sinks in the induction conclusion.

- On the one hand, this makes rippling technically and intuitively simpler (esp. for destructor style recursion) and better suited for human–computer interaction.
- On the other hand, however, the induction variables in the conclusion must be somehow limited in their character of being a sink: Unless we limit these sinks to swallow wave fronts consisting of destructors, we will have difficulties in finding a well-founded induction ordering justifying the induction-hypothesis applications.

11.5 Further Historical Limitations in Explicit Induction

Beside overcoming the must of generating induction axioms, it should be noted that QUODLIBET has some additional advantages over classical explicit-induction systems:

- The strong *admissibility restrictions* of explicit induction systems (i.e. specification only by functional programs, requiring their *completeness and termination proofs in advance*) have shown to superfluous [10, 62, 119, 120]. For the successful automation of inductive theorem proving we do not have to enforce a complete specification of the the recursive functions that participate in the recursion analysis. Indeed, QUODLIBET requires *neither termination nor completeness* for those recursive function definitions. Nevertheless, in QUODLIBET, these recursive function definitions come with a guarantee on consistency and are used for recursion analysis and other special heuristics. Thus, overspecification can be avoided and stepwise refinement of specifications becomes possible, with a guarantee on the monotonicity of validity [120].

  This is of practical relevance in applications. For instance, in [66], BERND LÖCHNER (who is not a developer but a user of QUODLIBET) writes:

  > "The translation into the input language of the inductive theorem prover QUODLIBET was straightforward. We later realized that this is difficult or impossible with several other inductive provers as these have problems with mutual recursive functions and partiality" . . .

- Another advantage compared to ACL2 with its poor user interface and its restriction to a complete reset after failure is the following: When automation fails, QUODLIBET typically stops early and presents the state of the proof attempt in a human-oriented form, whereas everything is lost (and only some of the developers may know what to do) when explicit induction generates a useless induction axiom (cf. Problem 11.1 in § 11.2).

11.6 Conclusion

Those researchers of the explicit induction community who realized what a strong restriction it is to fix the induction axiom before the actual induction proofs starts — the most important being [81, 82], [45], and [25] — always suffered from the wish to synthesize induction axioms. The same holds for the synthesis of simple recursive programs from their inductive soundness proofs [51, 58] and the more general task of instantiating meta-variables of the input theorem, where they also make sense as placeholders for concrete bounds and side conditions of the theorem that only a proof can tell. Indeed, the force to commit to a fixed induction axiom eagerly is only acceptable for simple proofs or simple theorems without meta-variables.

All in all, we have listed powerful arguments in §§ 9 and 10 and rebutted perceivable counterarguments in this § 11.

12 Conclusion

12.1 Human-Orientedness

As explained in § 3.2, completely automated black-box theorem proving is approaching its conceptual limits. Significant future progress will require a paradigm different from the artificial-intelligence exploration of the huge search spaces of machine-oriented misanthropic calculi. Human-oriented theorem proving and human-oriented calculi provide the only known alternative and have been gaining more and more acceptance within the last dozen years. The major tasks in the intended advanced form of human–computer interaction are

- the further development of interface notions following both hidden human cognitive concepts and the needs for powerful automation support, and
- the further improvement of the exploitation of the semantical information for the syntactical search processes.

The basic paradigm of interaction should be an anytime search process that knows about the humans' semantical strength and asks the human users for advice in their area of competence before getting lost in complexity. With a human-oriented main-stream integration following this paradigm, we can make man and machine a winning team.

12.2 *Descente Infinie*

Induction axioms were never necessary for the working mathematicians and are not anymore necessary in formalized mathematics or automated theorem proving due to [113]. It now suffices to solve the two hard tasks mentioned in § 6 (namely the Hypotheses Task and the Induction-Ordering Task) in mathematics as well as in automated theorem proving.

There is no need to make the generation of induction axioms more flexible, because we are in the lucky situation that we can have the cake *and* eat it: Indeed, we can remove the restrictions induction axioms put on us and improve the usefulness of the heuristic knowledge developed within the paradigm of explicit induction at the same time.

When recursion analysis or eager induction-hypotheses generation show us the right way, we can take it. When they do not, we do not have to care for them. We do not have to find a way to walk out of the maze of explicit induction. We can fly over it.

After a proof has been completed, we can read out of it what the induction axioms would have been.

As we do not need any induction axioms, however, we do not have to care at all whether our induction axioms should be *destructor style* or *constructor style* or whatever mixed styles one could imagine.

Moreover, note that — as discussed in Example 11.2 of § 11.3 — the case analysis suggested by recursion analysis is critical for the failure of explicit induction, but it serves only as a heuristic hint on how to start proof search in *descente infinie*.

Beside the recursion analysis telling us how to start off and beside the termination check of the induction ordering typically at the end, we do not need any special procedures for induction. An induction-hypothesis application is just a lemma application generating an additional ordering subgoal.

Descente infinie and explicit induction do not differ in the task (establishing inductive validity [120]) but in the way the proof search is organized. For simple proofs there is always a straightforward translation between the two. The difference becomes obvious only for proofs of difficult theorems.

The results of [113] on how to combine state-of-the-art deduction with *descente infinie* globally without induction axioms were not available when explicit induction started in the early 1970s. But now that we know how to do it, sticking to explicit induction as a must is scientifically backward. *Descente infinie* anyway admits a simulation of explicit induction that can profit from all the heuristics gathered in this field with the additional advantages

- that — contrary to explicit induction [15, 103, 104] — conflicting induction axioms do not have to be combined completely (because the major heuristic achievement of recursion analysis is to tell which variables to start induction with, cf. Example 11.2 of § 11.3), and
- that the induction ordering may stay open until the very end when all cases of the proof are known (because an earlier fixing of the induction ordering is hardly of any heuristic benefit ever).

Both items are of great practical effect [88, 91].

12.3 Summary

> *While the heuristics developed within the paradigm of explicit induction remain the method of choice for routine tasks, explicit induction is an obstacle to progress in program synthesis and in the automation of difficult proofs, where the proper induction axioms cannot be completely guessed in advance. Shifting to the paradigm of descente infinie overcomes this obstacle without sacrificing previous achievements.*

Acknowledgments

I would like to thank Dieter Hutter and Peter Padawitz for some very helpful advice, and Tobias Schmidt-Samoa for many fruitful discussions and a long list of most profound critical suggestions for this paper. Moreover, it is a great pleasure to thank Tobias for the wonderful years we shared working on QuodLibet and *descente infinie*.

Notes

Note 1 For instance, the basic paradigm of the human-oriented automated inductive theorem prover QUODLIBET [10] is the following: The working mathematician can feed the machine with his semantical knowledge of the domain by stating lemmas, and the machine can use these lemmas for sparse but deep proof search [88, 89, 90, 91]. When this search fails, the graphical user interface presents a not too deep state of the proof where progress stopped to the mathematician in a carefully designed human-oriented calculus [10, 61, 110] who may provide help with additional lemmas and other hints. It should be remarked, however, that the practical implementation of this paradigm is still more a task than an achievement. Cf. § 10 for more on this.

Note 2

- The OYSTER-CLAM *system* [18, 22] has to solve the very hard task of constructing proofs in the intuitionistic MARTIN-LÖF type theory of OYSTER, whereas the vast majority of mathematicians and ATP engineers would use transformations such as the one to the modal logic S4 [44, 34, 101] to prove intuitionistic theorems.

- Proof planning in the *old* ΩMEGA *system* [93] severely suffers from its commonplace natural deduction calculus, because it exports low-level tasks to higher levels of abstraction; these low-level tasks have turned out to be most problematic in practice because they can neither be ignored nor properly treated on the higher levels.

- The λCLAM *system* [20] does not have any fixed logic level at all.

Note 3 (Teaching Proof Search Procedures in Mathematics Lectures)
In the best lecture course I ever attended, every lecture an emeritus professor came into the lecture hall and asked what he is expected to teach here. “Analysis II!” “Do you know the theorem of so-and-so?” “What is that?” “. . . ” “No, we do not know that!” Then the emeritus gave a precise (but often incomplete) statement of the theorem, discussed it, and (after the students had a clear idea on the meaning of the theorem!) started proof *search*. In the lecture I learned most, he presented a proof that failed three times and was finally finished successfully overtime, not before patching the theorem. But this seems to be the best universities can give to their mathematics students today. (The missing systematics they had better learn from textbooks.) An apprentice is explained the easy procedures and shown the hard ones. Then, as we do not explain proof search to our students, it is probably one of the hard ones. Nevertheless, I do hope we will be able to do this some time.

Note 4 (Proof-Theoretical Peculiarities of Mathematical Induction)

The following often mentioned (cf. e.g. [20, § 5]) proof-theoretical peculiarities of mathematical induction do not really have a special practical effect on inductive theorem proving, simply because efficiency problems cause the same effects already for the case of deductive theorem proving:

- As the theory of arithmetic is not enumerable ([43, 44]), completeness of a calculus w.r.t. the standard notion of validity cannot be achieved.

  In practice, however, it does not matter whether our proof attempt fails because our theorem will not be enumerated ever or will not be enumerated before doomsday.

- By GENTZEN's Hauptsatz on Cut elimination [40] there is no need to invent new formulas in a proof of a deductive theorem. Indeed, such a proof can be restricted to "sub"-formulas of the theorem under consideration. In contrast to lemma application (i.e. Cut) in a deductive proof tree, the application of induction hypotheses and lemmas inside an inductive reasoning cycle cannot generally be eliminated in the sense that the sub-formula property could be obtained, cf. [59]. Thus, for inductive theorem proving, "creativity" cannot be restricted to finding just the proper instances, but may require the invention of new lemmas and notions.

  Again, in practice, however, it does not matter whether we have to extend our proof search to additional lemmas and notions for principled reasons or for tractability [11].

Note 5 (Implicit and Inductionless Induction)

Alternative approaches to automation of mathematical induction evolved from the *Knuth–Bendix Completion Procedure* and were summarized in the *School of Implicit Induction*, which comprises Proof by Consistency (Inductionless Induction), *descente infinie* and implicit induction orderings (term orderings). Furthermore, there is pioneering work on the combination of induction and co-induction; cf. e.g. [73]. While Proof by Consistency and implicit induction orderings seem to be of merely theoretical interest today [114], we should carefully distinguish *descente infinie* from the mainstream work on explicit induction.

Note 6 (The Idea of Rippling)

Roughly speaking, the success in proving *simple* theorems by induction automatically, can be explained as follows: If we look upon the task of proving a simple theorem as reducing it to a tautology, then we have more heuristic guidance when we know that we probably have to do it by mathematical induction: Tautologies are to be found everywhere, but the induction hypothesis we are going to apply can restrict the search space tremendously.

In a famous cartoon of ALAN BUNDY's, the original theorem is symbolized as a zigzagged mountainscape and the reduced theorem after the unfolding of recursive operators as a lake with ripples. Instead of searching for an arbitrary tautology, we know that we have to *get rid of the ripples* to be able to apply an instance of the theorem as induction hypothesis, as mirrored by the calm surface of the lake.

Note 7 (The Sociological Aspect of Explicit Induction as Normal Science)

Another way in that explicit induction blocks scientific progress is a sociological one. The heuristics to generate induction axioms in explicit induction have hardly changed since the end of the 1970s. Some minor conceptual improvements (such as [103, 104], e.g.) have turned out to be contra-productive in the practical context of a highly optimized "waterfall", because later phases were already optimized to patch the weaknesses of the previous ones. With all the men-power that went into explicit induction systems such as INKA [7] or ACL2 [56], these systems have become so well-tuned to all simple standard problems that it is hardly possible to demonstrate their shortcomings to referees within the time they are willing to spend on the subject.

Beside that, to become competitive with ACL2 requires a common effort and years of work with little chance for economic support or academic funding, approval, or rewards. In spite of this, mainly due to the idealism of ULRICH KÜHLER and TOBIAS SCHMIDT-SAMOA and a bunch of their students, *descente infinie* in QUODLIBET [10] — as explained in §§ 9 and 11 — has already by now been able to outperform the formerly well-funded *normal-science* [63, 117] School of Explicit Induction.

Note 8 (On the Likeliness of Alternative Integrations of *Descente Infinie* into State-of-the-Art Deductive Calculi)

I consider the integration of *descente infinie* into state-of-the-art free-variable sequent and tableau calculi to be the my most important scientific contribution. Since I actually have searched the whole conceivable space of possible combinations far beyond what is documented in [113], I am pretty sure that my paper [113] presents not only an elegant and both human- and machine-oriented combination of *descente infinie* and state-of-the-art deduction (including liberalized versions (δ^+) of the δ-rules), but also the only possible one (up to isomorphism and beside some possible variations (cf. [111]) and simplifications in formalizing variable-conditions) that actually *models* the mathematical process of proof search by *descente infinie itself* and *directly* supports it with the data structures required for a formal treatment and does not *encode* some form of induction. (Such an encoding can be found, e.g., in the induction rule of [41], or by application of the second-order Theorem of NOETHERian Induction (N) (cf. § 6), or the second-order Axiom of Structural Induction (S) (cf. § 6), or by generation of first-order induction axioms.)

Note 9 (Semantical Requirements of [113])

As described in [113, § 2.1.4] all we need for the soundness of our integration of *descente infinie* into two-valued logics are the validity of

- the well-known *Substitution* [*Value*] *Lemma* (as, e.g., shown for different logics in [3, Lemma 3], [4, Lemma 5401(a)], [31, p. 127], [35, p. 120], and [36, Proposition 2.31]) and
- the trivial *Explicitness Lemma* (i.e. the values of variables not explicitly freely occurring in a term or formula have no effect on the value of the term or formula, resp.) (as, e.g., shown for different logics in [3, Lemma 2], [4, Proposition 5400], and [36, Proposition 2.30]).

Note 10 (Lemma Application)

Lemma application works as follows. Suppose that our proof goals consist of *sequents* which are just disjunctive lists of formulas. (This is the simplest form of a sequent that will do for two-valued logics.) When a lemma $A_1, \ldots, A_m$ is a subsequent of a sequent Γ to be proved (i.e. if, for all $i \in \{1, \ldots, m\}$, the formula A_i is listed in Γ), its application closes the branch of this sequent (*subsumption*). Otherwise, the conjugates of the missing formulas C_i are added to the child sequents (premises), one child per missing formula. This can be seen as Cuts on C_i plus subsumption. More precisely — modulo associativity, commutativity, and idempotency — a sequent $A_1, \ldots, A_m, B_1, \ldots, B_n$ can be reduced by application of the lemma $A_1, \ldots, A_m, C_1, \ldots, C_p$ to the sequents

$$\overline{C_1}, A_1, \ldots, A_m, B_1, \ldots, B_n \qquad \cdots \qquad \overline{C_p}, A_1, \ldots, A_m, B_1, \ldots, B_n.$$

In addition, roughly speaking, any time we apply a lemma, we can instantiate its free variables locally and arbitrarily. Cf. [113, 115] for more on this.

Note 11 (Integration of *Descente Infinie* into Logical Frameworks)

Item 4 of the enumeration in § 9 is typically no problem because we can get along with semantical orderings [110, Definition 13.7]. Indeed, we do not need term orderings [97] anymore, contrary to what was the case with QUODLIBET's predecessor UNICOM [46].

Items 1, 2, and 3, however, do not seem to be easily achievable with ISABELLE/HOL [70, 71], for instance. A logical framework (such as ISABELLE [74, 75, 76]) can hardly mirror general mathematical activity, but only the logic calculi known at the time of its development. This makes progress toward human-oriented automatable calculi very difficult. As a convenient realization of *descente infinie* does not seem to be so easily possible in ISABELLE-based systems, a lot of additional lemmas (or else ingenious recursive specification) may be necessary as described in § 1 (or else the solution) of [100]. Moreover, for the idea to support program synthesis via *descente infinie* on the lower level of inductive theorem proving for software verification (cf. our § 8 and [25]), the recursion facilities of ISABELLE/HOL seem to be insufficient: KONRAD SLIND's recursion theorems [94] require termination proofs at a too early stage of development [119].

Note 12 (Productive Use of Failure and Patching Faulty Conjectures)
Although, a failure of a proof is a complete one in case of a wrong induction axiom in explicit induction, from such a failure, we might gain some insight on the proof [55] or on the conjecture [82, 83]. And then we may start a more promising proof attempt with different settings.

References

[1] FABIO ACERBI (2000). PLATO*: Parmenides 149a7–c3. A Proof by Complete Induction?*. Archive for History of Exact Sciences **55**, pp. 57–76, Springer.

[2] WILHELM ACKERMANN (1940). *Zur Widerspruchsfreiheit der Zahlentheorie.* Mathematische Annalen **117**, pp. 163–194. Received Aug. 15, 1939. `http://dz-srv1.sub.uni-goettingen.de/sub/digbib/loader?did=D37625` (July 23, 2007).

[3] PETER B. ANDREWS (1972). *General Models, Descriptions, and Choice in Type Theory.* J. Symbolic Logic **37**, pp. 385–394.

[4] PETER B. ANDREWS (2002). *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* 2nd ed. (1st ed. 1986), Academic Press (Elsevier).

[5] GÜNTER ASSER (ed.) (1990). GUISEPPE PEANO *— Arbeiten zur Analysis und zur mathematischen Logik.* Teubner-Archiv zur Mathematik, Vol. 13, B. G. Teubner Verlagsgesellschaft, Leipzig.

[6] SERGE AUTEXIER (2003). *Hierarchical Contextual Reasoning.* PhD thesis. Saarland Univ..

[7] SERGE AUTEXIER, DIETER HUTTER, HEIKO MANTEL, AXEL SCHAIRER (1999). INKA *5.0 — A Logical Voyager.* 16th CADE 1999, LNAI 1632, pp. 207–211, Springer.

[8] SERGE AUTEXIER, CHRISTOPH BENZMÜLLER, CHAD E. BROWN, ARMIN FIEDLER, DIETER HUTTER, ANDREAS MEIER, MARTIN POLLET, TOBIAS SCHMIDT-SAMOA, JÖRG SIEKMANN, GEORG ROCK, WERNER STEPHAN, MARC WAGNER, CLAUS-PETER WIRTH (2004). *Mathematics Assistance Systems.* Lecture course at Saarland Univ., WS 2004/5. `http://www.ags.uni-sb.de/~omega/teach/MAS0405/` (April 15, 2005).

[9] SERGE AUTEXIER, CHRISTOPH BENZMÜLLER, DOMINIK DIETRICH, ANDREAS MEIER, CLAUS-PETER WIRTH (2006). *A Generic Modular Data Structure for Proof Attempts Alternating on Ideas and Granularity.* 4th MKM 2005, LNAI 3863, pp. 126–142, Springer. `http://www.ags.uni-sb.de/~cp/p/pds` (July 22, 2005).

[10] JÜRGEN AVENHAUS, ULRICH KÜHLER, TOBIAS SCHMIDT-SAMOA, CLAUS-PETER WIRTH (2003). *How to Prove Inductive Theorems?* QUODLIBET*!*. 19th CADE 2003, LNAI 2741, pp. 328–333, Springer. `http://www.ags.uni-sb.de/~cp/p/quodlibet` (July 23, 2003).

[11] MATTHIAS BAAZ, ALEXANDER LEITSCH (1995). *Methods of Functional Extension.* Collegium Logicum, Annals of the KURT GÖDEL Society, Vol. 1, pp. 87–122, Springer.

[12] KLAUS BARNER (2001). *Das Leben* FERMAT*s.* DMV-Mitteilungen 3/2001, pp. 12–26.

[13] DAVID BASIN, TOBY WALSH (1996). *A Calculus for and Termination of Rippling.* J. Automated Reasoning **16**, pp. 147–180, Kluwer (Springer).

[14] SUSANNE BIUNDO, BIRGIT HUMMEL, DIETER HUTTER, CHRISTOPH WALTHER (1986). *The Karlsruhe Induction Theorem Proving System.* 8th CADE 1986, LNCS 230, pp. 672–674, Springer.

[15] ROBERT S. BOYER, J S. MOORE (1979). *A Computational Logic.* Academic Press (Elsevier).

[16] ROBERT S. BOYER, J S. MOORE (1988). *A Computational Logic Handbook.* Academic Press.

[17] ARNIM BUCH, THOMAS HILLENBRAND (1996). WALDMEISTER*: Development of a High Performance Completion-Based Theorem Prover*. SEKI-Report SR–96–01, FB Informatik, Univ. Kaiserslautern.

[18] ALAN BUNDY (1988). *The use of Explicit Plans to Guide Inductive Proofs.* DAI Research Paper No. 349, Dept. Artificial Intelligence, Univ. Edinburgh. Short version in: 9th CADE 1988, LNAI 310, pp. 111–120, Springer.

[19] ALAN BUNDY (1989). *A Science of Reasoning.* DAI Research Paper No. 445, Dept. Artificial Intelligence, Univ. Edinburgh. Also in: [64], pp. 178–198.

[20] ALAN BUNDY (1999). *The Automation of Proof by Mathematical Induction.* Informatics Research Report No. 2, Division of Informatics, Univ. Edinburgh. Also in: [86], Vol. 1, pp. 845–911.

[21] ALAN BUNDY, FRANK VAN HARMELEN, JANE HESKETH, ALAN SMAILL, ANDREW STEVENS (1989). *A Rational Reconstruction and Extension of Recursion Analysis.* In: N. S. Sridharan (ed.). *Proc. 11th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 359–365, Morgan Kaufman (Elsevier).

[22] ALAN BUNDY, FRANK VAN HARMELEN, C. HORN, ALAN SMAILL (1990). *The Oyster-Clam System.* 10th CADE 1990, LNAI 449, pp. 647–648, Springer.

[23] ALAN BUNDY, ANDREW STEVENS, FRANK VAN HARMELEN, ANDREW IRELAND, ALAN SMAILL (1991). *Rippling: A Heuristic for Guiding Inductive Proofs.* DAI Research Paper No. 567, Dept. Artificial Intelligence, Univ. Edinburgh. Also in: Artificial Intelligence (1993) **62(2)**, pp. 185–253.

[24] ALAN BUNDY, DIETER HUTTER, DAVID BASIN, ANDREW IRELAND (2005). *Rippling: Meta-Level Guidance for Mathematical Reasoning.* Cambridge Univ. Press.

[25] ALAN BUNDY, LUCAS DIXON, JEREMY GOW, JACQUES FLEURIOT (2006). *Constructing Induction Rules for Deductive Synthesis Proofs.* Electronic Notes in Theoretical Computer Sci. **153**, pp. 3–21, Elsevier.

[26] PAOLO BUSSOTTI (2006). *From* FERMAT *to* GAUSS*: indefinite descent and methods of reduction in number theory.* Algorismus **55**, Dr. Erwin Rauner Verlag, Augsburg.

[27] RICARDO CAFERRA, GERNOT SALZER (eds.) (2000). *Automated Deduction in Classical and Non-Classical Logics.* LNAI 1761, Springer.

[28] LOUISE A. DENNIS, MATEJA JAMNIK, MARTIN POLLET (2005). *On the Comparison of Proof Planning Systems* λCLAM, ΩMEGA *and* ISAPLANNER. Electronic Notes in Theoretical Computer Sci. **151**, pp. 93–110, Elsevier.

[29] LUCAS DIXON (2005). *Interactive and Hierarchical Tracing of Techniques in* ISAPLANNER. Accepted for presentation at: Workshop on User Interfaces for Theorem Provers (UITP 2005). `http://homepages.inf.ed.ac.uk/ldixon/papers/uitp-05-traces/isaptracing.ps.gz (Aug. 11, 2005).`

[30] LUCAS DIXON, JACQUES FLEURIOT (2003). ISAPLANNER*: A Prototype Proof Planner in* ISABELLE. 19th CADE 2003, LNAI 2741, pp. 279–283, Springer.

[31] HERBERT B. ENDERTON (1973). *A Mathematical Introduction to Logic.* 2nd printing, Academic Press (Elsevier).

[32] EUCLID OF ALEXANDRIA (ca. 300 B.C.). *Elements.* English translation: Thomas L. Heath (ed.). *The Thirteen Books of* EUCLID*'s Elements*. Cambridge Univ. Press, 1908. Web version: `http://www.perseus.tufts.edu/cgi-bin/ptext?doc=Perseus:text:1999.01.0086 (Aug. 15, 2006).`. Alternative English web version: D. E. Joyce (ed.). EUCLID*'s Elements*. Dept. Math. & Comp. Sci., Clark Univ., Worcester, MA. `http://aleph0.clarku.edu/~djoyce/java/elements/Euclid.html (March 24, 2003).`

[33] PIERRE FERMAT (1891ff.). *Œuvres de* FERMAT. Paul Tannery, Charles Henry (eds.), Gauthier-Villars, Paris. `http://fr.wikisource.org/wiki/%C5%92uvres_de_Fermat (Aug. 15, 2006).`

[34] MELVIN FITTING (1983). *Proof Methods for Modal and Intuitionistic Logics.* D. Reidel, Dordrecht.

[35] MELVIN FITTING (1996). *First-Order Logic and Automated Theorem Proving.* 2nd extd. ed. (1st ed. 1990), Springer.

[36] MELVIN FITTING (2002). *Types, Tableaus, and* GÖDEL*'s God.* Kluwer (Springer).

[37] GOTTLOB FREGE (1879). *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens.* Verlag von L. Nebert, Halle an der Saale.

[38] KURT VON FRITZ (1945). *The Discovery of Incommensurability by* HIPPASUS OF METAPONTUM. Annals of Mathematics **46**, pp. 242–264. German translation *Die Entdeckung der Inkommensurabilität durch* HIPPASOS VON METAPONT in: Oscar Becker (ed.). *Zur Geschichte der griechischen Mathematik*, pp. 271–308, Wissenschaftliche Buchgesellschaft, Darmstadt, 1965.

[39] DOV GABBAY, CHRISTOPHER JOHN HOGGER, J. ALAN ROBINSON (eds.) (1993ff.). *Handbook of Logic in Artificial Intelligence and Logic Programming.* Clarendon Press.

[40] GERHARD GENTZEN (1934f.). *Untersuchungen über das logische Schließen.* Mathematische Zeitschrift **39**, pp. 176-210, 405–431.

[41] GERHARD GENTZEN (1938). *Die gegenwärtige Lage in der mathematischen Grundlagenforschung – Neue Fassung des Widerspruchsfreiheitsbeweises für die reine Zahlentheorie.* Forschungen zur Logik und zur Grundlegung der exakten Wissenschaften, Folge 4, Leipzig.

[42] MARTIN GIESE, WOLFGANG AHRENDT (1999). HILBERT*'s ε-Terms in Automated Theorem Proving.* 8th TABLEAUX 1999, LNAI 1617, pp. 171–185, Springer.

[43] KURT GÖDEL (1931). *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.* Monatshefte für Mathematik und Physik **38**, pp. 173–198.

[44] KURT GÖDEL (1986ff.). *Collected Works.* Solomon Feferman (ed.), Oxford Univ. Press.

[45] JEREMY GOW (2004). *The Dynamic Creation of Induction Rules Using Proof Planning.* PhD thesis, School of Informatics, Univ. Edinburgh.

[46] BERNHARD GRAMLICH, WOLFGANG LINDNER (1991). *A Guide to* UNICOM, *an Inductive Theorem Prover Based on Rewriting and Completion Techniques.* SEKI-Report SR–91–17 (SFB), FB Informatik, Univ. Kaiserslautern. `http://agent.informatik.uni-kl.de/seki/1991/Lindner.SR-91-17.ps.gz (May 09, 2000).`

[47] BERNHARD GRAMLICH, CLAUS-PETER WIRTH (1996). *Confluence of Terminating Conditional Term Rewriting Systems Revisited.* 7th RTA 1996, LNCS 1103, pp. 245–259, Springer. `http://www.ags.uni-sb.de/~cp/p/rta96 (Aug. 05, 2001).`

[48] DAVID HILBERT, PAUL BERNAYS (1968/70). *Grundlagen der Mathematik.* 2nd rev. ed. (1st ed. 1934/39), Springer.

[49] DIETER HUTTER (1990). *Guiding Inductive Proofs.* 10th CADE 1990, LNAI 449, pp. 147–161, Springer.

[50] DIETER HUTTER (1991). *Mustergesteuerte Strategien für das Beweisen von Gleichungen.* PhD thesis, Univ. Karlsruhe.

[51] DIETER HUTTER (1994). *Synthesis of Induction Orderings for Existence Proofs.* 12th CADE 1994, LNAI 814, pp. 29–41, Springer.

[52] DIETER HUTTER (1997). *Colouring Terms to Control Equational Reasoning.* J. Automated Reasoning **18**, pp. 399–442, Kluwer (Springer).

[53] DIETER HUTTER, ALAN BUNDY (1999). *The Design of the CADE-16 Inductive Theorem Prover Contest.* 16th CADE 1999, LNAI 1632, pp. 374–377, Springer.

[54] DIETER HUTTER, WERNER STEPHAN (eds.) (2005). *Mechanizing Mathematical Reasoning: Essays in Honor of* JÖRG SIEKMANN *on the Occasion of His 60th Birthday.* LNAI 2605, Springer.

[55] ANDREW IRELAND, ALAN BUNDY (1994). *Productive Use of Failure in Inductive Proof.* DAI Research Paper No. 716, Dept. Artificial Intelligence, Univ. Edinburgh. Also in: J. Automated Reasoning (1996) **16(1-2)**, pp. 79–111, Kluwer (Springer).

[56] MATT KAUFMANN, PANAGIOTIS “PETE” MANOLIOS, J S. MOORE (2000). *Computer-Aided Reasoning: An Approach.* Kluwer (Springer).

[57] HUBERT C. KENNEDY (1973). *Selected works of* GUISEPPE PEANO. George Allen & Unwin, London.

[58] INA KRAAN, DAVID BASIN, ALAN BUNDY (1995). *Middle-Out Reasoning for Synthesis and Induction.* DAI Research Paper No. 729, Dept. Artificial Intelligence, Univ. Edinburgh. Also in: J. Automated Reasoning (1996) **16(1–2)**, pp. 113–145, Kluwer (Springer).

[59] GEORG KREISEL (1965). *Mathematical Logic.* In: [87], Vol. III, pp. 95–195.

[60] HANS-JÖRG KREOWSKI, UGO MONTANARI, FERNANDO OREJAS, GRZEGORZ ROZENBERG, GABRIELE TAENTZER (eds.) (2005). *Formal Methods in Software and Systems Modeling, Essays Dedicated to* HARTMUT EHRIG, *on the Occasion of His 60th Birthday.* LNCS 3393, Springer.

[61] ULRICH KÜHLER (2000). *A Tactic-Based Inductive Theorem Prover for Data Types with Partial Operations*. PhD thesis, Infix, Akademische Verlagsgesellschaft Aka GmbH, Sankt Augustin, Berlin. `http://www.ags.uni-sb.de/~cp/p/kuehlerdiss` (July 23, 2005).

[62] ULRICH KÜHLER, CLAUS-PETER WIRTH (1996). *Conditional Equational Specifications of Data Types with Partial Operations for Inductive Theorem Proving*. SEKI-Report SR–96–11, FB Informatik, Univ. Kaiserslautern. Short version in: 8th RTA 1997, LNCS 1232, pp. 38–52, Springer. `http://www.ags.uni-sb.de/~cp/p/rta97` (Aug. 05, 2001).

[63] THOMAS S. KUHN (1962). *The Structure of Scientific Revolutions*. Univ. Chicago Press.

[64] JEAN-LOUIS LASSEZ, GORDON D. PLOTKIN (eds.) (1991). *Computational Logic — Essays in Honor of* J. ALAN ROBINSON. MIT Press.

[65] ALBERT C. LEISENRING (1969). *Mathematical Logic and* HILBERT*'s ε-Symbol*. Gordon and Breach, New York.

[66] BERND LÖCHNER (2006). *Things to know when implementing LPO*. Int. J. Artificial Intelligence Tools (2006) **15(1)**, pp. 53–79. Short version in: Geoff Sutcliffe, Stephan Schulz, T. Tammet (eds.). Proc. 1st Workshop on Empirically Successful First Order Reasoning (ESFOR'04), 2004.

[67] BERND LÖCHNER (2006). *Advances in Equational Theorem Proving — Architecture, Algorithms, and Redundancy Avoidance*. PhD thesis, Univ. Kaiserslautern. `http://kluedo.ub.uni-kl.de/volltexte/2006/1969/` (Aug. 25, 2006).

[68] MICHAEL SEAN MAHONEY (1994). *The Mathematical Career of* PIERRE *de* FERMAT *1601–1665*. 2nd rev. ed. (1st ed. 1973), Princeton Univ. Press.

[69] DALE A. MILLER (1992). *Unification under a Mixed Prefix*. J. Symbolic Computation **14**, pp. 321–358, Academic Press (Elsevier).

[70] TOBIAS NIPKOW, LAWRENCE C. PAULSON, MARKUS WENZEL (2000). ISABELLE*'s Logics: HOL*. Web only. `http://isabelle.in.tum.de/PSV2000/doc/logics-HOL.pdf` (Sept. 11, 2005).

[71] TOBIAS NIPKOW, LAWRENCE C. PAULSON, MARKUS WENZEL (2002). ISABELLE/HOL *— A Proof Assistant for Higher-Order Logic*. LNCS 2283, Springer.

[72] P. ODIFREDDI (ed.) (1990). *Logic and Computer Science*. Academic Press (Elsevier).

[73] PETER PADAWITZ (2005). *Expander2*. In: [60], pp. 236–258.

[74] LAWRENCE C. PAULSON (1989). *The Foundation of a Generic Theorem Prover*. J. Automated Reasoning **5**, pp. 363–397, Kluwer (Springer).

[75] LAWRENCE C. PAULSON (1990). ISABELLE*: The Next 700 Theorem Provers*. In: [72], pp. 361–386.

[76] LAWRENCE C. PAULSON (2004). *The* ISABELLE *Reference Manual*. With Contributions by TOBIAS NIPKOW and Markus Wenzel, April 20, 2004. Web only. `http://www.cl.cam.ac.uk/Research/HVG/Isabelle/dist/packages/Isabelle/doc/ref.pdf` (Sept. 13, 2005).

[77] GUISEPPE PEANO (ed.) (1884). ANGELO GENOCCHI — *Calcolo differenziale e principii di calcolo integrale.* Fratelli Bocca, Torino. German translation: [80].

[78] GUISEPPE PEANO (1896f.). *Studii di Logica Matematica.* Atti della Reale Accademia delle Scienze di Torino — Classe di Scienze Morali, Storiche e Filologiche e Classe di Scienze Fisiche, Matematiche e Naturali **32**, pp. 565–583. Also in: Atti della Reale Accademia delle Scienze di Torino — Classe di Scienze Fisiche, Matematiche e Naturali **32**, pp. 361–397. English translation *Studies in Mathematical Logic* in: [57], pp. 190–205. German translation: [79].

[79] GUISEPPE PEANO (1899). *Über mathematische Logik.* German translation of [78]. In: [80], Appendix 1. Facsimile also in: [5], pp. 10–26.

[80] GUISEPPE PEANO (ed.) (1899). ANGELO GENOCCHI — *Differentialrechnung und Grundzüge der Integralrechnung.* German translation of [77], B. G. Teubner Verlagsgesellschaft, Leipzig.

[81] MARTIN PROTZEN (1994). *Lazy Generation of Induction Hypotheses.* 12th CADE 1994, LNAI 814, pp. 42–56, Springer.

[82] MARTIN PROTZEN (1995). *Lazy Generation of Induction Hypotheses and Patching Faulty Conjectures.* PhD thesis, Infix, Akademische Verlagsgesellschaft Aka GmbH, Sankt Augustin, Berlin.

[83] MARTIN PROTZEN (1996). *Patching Faulty Conjectures.* 13th CADE 1996, LNAI 1104, pp. 77–91, Springer.

[84] ALEXANDER RIAZANOV, ANDREI VORONKOV (2001). *Vampire 1.1 (System Description).* 1st IJCAR 2001, LNAI 2083, pp. 376–380, Springer.

[85] J. ALAN ROBINSON (1965). *A Machine-Oriented Logic based on the Resolution Principle.* In: [92], Vol. 1, pp. 397–415.

[86] J. ALAN ROBINSON, ANDREI VORONKOV (eds.) (2001). *Handbook of Automated Reasoning.* Elsevier.

[87] T. L. SAATY (ed.) (1965). *Lectures on Modern Mathematics.* John Wiley & Sons, New York.

[88] TOBIAS SCHMIDT-SAMOA (2004). *The New Standard Tactics of the Inductive Theorem Prover* QUODLIBET. SEKI-Report SR–2004–01, ISSN 1437-4447. `http://www.ags.uni-sb.de/~cp/p/sr200401` (April 15, 2005).

[89] TOBIAS SCHMIDT-SAMOA (2006). *An Even Closer Integration of Linear Arithmetic into Inductive Theorem Proving.* Electronic Notes in Theoretical Computer Sci. **151**, pp. 3–20, Elsevier. `http://www.elsevier.com/locate/entcs` (Aug. 20, 2006).

[90] TOBIAS SCHMIDT-SAMOA (2006). *Flexible Heuristics for Simplification with Conditional Lemmas by Marking Formulas as Forbidden, Mandatory, Obligatory, and Generous.* J. Applied Non-Classical Logics **16(1–2)**, pp. 209–239. `http://www.ags.uni-sb.de/~cp/p/jancl` (March 08, 2006).

[91] TOBIAS SCHMIDT-SAMOA (2006). *Flexible Heuristic Control for Combining Automation and User-Interaction in Inductive Theorem Proving.* PhD thesis, Univ. Kaiserslautern. `http://www.ags.uni-sb.de/~cp/p/samoadiss` (July 30, 2006).

[92] JÖRG SIEKMANN, GRAHAM WRIGHTSON (eds.) (1983). *Automation of Reasoning.* Springer.

[93] JÖRG SIEKMANN, CHRISTOPH BENZMÜLLER, VLADIMIR BREZHNEV, LASSAAD CHEIKHROUHOU, ARMIN FIEDLER, ANDREAS FRANKE, HELMUT HORACEK, MICHAËL KOHLHASE, ANDREAS MEIER, ERICA MELIS, MARKUS MOSCHNER, IMMANUËL NORMANN, MARTIN POLLET, VOLKER SORGE, CARSTEN ULLRICH, CLAUS-PETER WIRTH, JÜRGEN ZIMMER (2002). *Proof Development with* ΩMEGA. 18th CADE 2002, LNAI 2392, pp. 144–149, Springer. `http://www.ags.uni-sb.de/~cp/p/omega (July 23, 2003).`

[94] KONRAD SLIND (1996). *Function Definition in Higher-Order Logic.* 9th Int. Conf. on Theorem Proving in Higher-Order Logics (TPHOLs), Turku (Finland), 1996, LNCS 1125, pp. 381–397, Springer.

[95] ALAN SMAILL, IAN GREEN (1996). *Higher-Order Annotated Terms for Proof Search.* 9th Int. Conf. on Theorem Proving in Higher-Order Logics (TPHOLs), Turku (Finland), 1996, LNCS 1125, pp. 399–413, Springer.

[96] RAYMOND M. SMULLYAN (1968). *First-Order Logic.* Springer.

[97] JOACHIM STEINBACH (1995). *Simplification Orderings — History of Results.* Fundamenta Informaticae **24**, pp. 47–87.

[98] ANDREW STEVENS (1988). *A Rational Reconstruction of* BOYER *and* MOORE*'s Technique for Constructing Induction Formulas.* Y. Kodratoff (ed.). *8th European Conf. on Artificial Intelligence (ECAI 1988)*, pp. 565–570, Pitman Publ., London.

[99] ANDREW STEVENS (1990). *An Improved Method for the Mechanization of Inductive Proof.* PhD thesis, Dept. Artificial Intelligence, Univ. Edinburgh.

[100] CHRISTIAN URBAN, CHRISTINE TASSON (2005). *Nominal Techniques in* ISABELLE/HOL. 20th CADE 2005, LNAI 3632, pp. 38–53, Springer.

[101] LINCOLN A. WALLEN (1990). *Automated Proof Search in Non-Classical Logics.* MIT Press.

[102] CHRISTOPH WALTHER (1988). *Argument-Bounded Algorithms as a Basis for Automated Termination Proofs.* 9th CADE 1988, LNAI 310, pp. 601–622, Springer.

[103] CHRISTOPH WALTHER (1992). *Computing Induction Axioms.* 3rd LPAR 1992, LNAI 624, pp. 381–392, Springer.

[104] CHRISTOPH WALTHER (1993). *Combining Induction Axioms by Machine.* In: Ruzena Bajcsy (ed.). *Proc. 13th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 95–101, Morgan Kaufman (Elsevier).

[105] CHRISTOPH WALTHER (1994). *Mathematical Induction.* In: [39], Vol. 2, pp. 127–228.

[106] CHRISTOPH WALTHER, STEPHAN SCHWEIZER (2003). *About* ✓ERIFUN. 19th CADE 2003, LNAI 2741, pp. 322–327, Springer.

[107] CHRISTOPH WALTHER, STEPHAN SCHWEIZER (2005). *Automated Termination Analysis of Incompletely Defined Programs.* 11th LPAR 2004, LNAI 3452, pp. 332–346, Springer.

[108] CHRISTOPH WALTHER, STEPHAN SCHWEIZER (2005). *Reasoning about Incompletely Defined Programs*. 12th LPAR 2005, LNAI 3835, pp. 427–442, Springer.

[109] CLAUS-PETER WIRTH (1995). *Syntactic Confluence Criteria for Positive/Negative-Conditional Term Rewriting Systems*. SEKI-Report SR–95–09 (SFB), FB Informatik, Univ. Kaiserslautern. `http://www.ags.uni-sb.de/~cp/p/sr9509 (Aug. 05, 2001).`

[110] CLAUS-PETER WIRTH (1997). *Positive/Negative-Conditional Equations: A Constructor-Based Framework for Specification and Inductive Theorem Proving*. PhD thesis, Verlag Dr. Kovač, Hamburg. `http://www.ags.uni-sb.de/~cp/p/diss (Aug. 05, 2001).`

[111] CLAUS-PETER WIRTH (1998). *Full First-Order Sequent and Tableau Calculi With Preservation of Solutions and the Liberalized δ-Rule but Without Skolemization*. Report 698/1998, FB Informatik, Univ. Dortmund. Short version in: GERNOT SALZER, RICARDO CAFERRA (eds.). Proc. 2nd Int. Workshop on First-Order Theorem Proving (FTP'98), pp. 244–255, Tech. Univ. Vienna, 1998. Short version also in: [27], pp. 283–298. `http://www.ags.uni-sb.de/~cp/p/ftp98 (Aug. 05, 2001).`

[112] CLAUS-PETER WIRTH (2002). *A New Indefinite Semantics for* HILBERT*'s epsilon*. 11th TABLEAUX 2002, LNAI 2381, pp. 298–314, Springer. `http://www.ags.uni-sb.de/~cp/p/epsi (Feb. 04, 2002).`

[113] CLAUS-PETER WIRTH (2004). *Descente Infinie + Deduction*. Logic J. of the IGPL **12**, pp. 1–96, Oxford Univ. Press. `http://www.ags.uni-sb.de/~cp/p/d (Sept. 12, 2003).`

[114] CLAUS-PETER WIRTH (2005). *History and Future of Implicit and Inductionless Induction: Beware the old jade and the zombie!*. In: [54], pp. 192–203. `http://www.ags.uni-sb.de/~cp/p/zombie (Dec. 02, 2002).`

[115] CLAUS-PETER WIRTH (2006). $\lim +$, δ^+*, and Non-Permutability of* β*-Steps*. SEKI-Report SR–2005–01, ISSN 1437-4447, rev. ed. of July 30, 2006. `http://www.ags.uni-sb.de/~cp/p/nonpermut (July 30, 2006).`

[116] CLAUS-PETER WIRTH (2006). *A Self-Contained Discussion of* FERMAT*'s Only Explicitly Known Proof by Descente Infinie*. SEKI-Working-Paper SWP–2006–02, ISSN 1860-5931. `http://www.ags.uni-sb.de/~cp/p/fermatsproof (Aug. 25, 2006).`

[117] CLAUS-PETER WIRTH (2007). THOMAS S. KUHN*: The Structure of Scientific Revolutions — Zweisprachige Auszüge mit Deutschem Kommentar*. SEKI-Working-Paper SWP–2007–01, ISSN 1860-5931. `http://www.ags.uni-sb.de/~cp/p/kuhn (June 20, 2008).`

[118] CLAUS-PETER WIRTH (2008). HILBERT*'s epsilon as an Operator of Indefinite Committed Choice*. J. Applied Logic, Elsevier, `http://dx.doi.org/10.1016/j.jal.2007.07.009`. `http://www.ags.uni-sb.de/~cp/p/epsi (Oct. 15, 2007).`

[119] CLAUS-PETER WIRTH (2008). *Shallow Confluence of Conditional Term Rewriting Systems*. J. Symbolic Computation, Elsevier, `http://dx.doi.org/10.1016/`

j.jsc.2008.05.005. http://www.ags.uni-sb.de/~cp/p/shallow (May 31, 2008).

[120] CLAUS-PETER WIRTH, BERNHARD GRAMLICH (1994). *On Notions of Inductive Validity for First-Order Equational Clauses.* 12th CADE 1994, LNAI 814, pp. 162–176, Springer. http://www.ags.uni-sb.de/~cp/p/cade94 (Aug. 05, 2001).

[121] CLAUS-PETER WIRTH, CHRISTOPH BENZMÜLLER, ARMIN FIEDLER, ANDREAS MEIER, SERGE AUTEXIER, MARTIN POLLET, CARSTEN SCHÜRMANN (2003). *Human-Oriented Theorem Proving — Foundations and Applications.* Lecture course at Saarland Univ., WS 2003/4. http://www.ags.uni-sb.de/~cp/teaching/hotp (Sept. 12, 2003).