Mathematical Induction by

Descente Infinie

Claus-Peter Wirth

Application for a DFG research grant
(Antrag auf Gewährung einer Sachbeihilfe, Neuantrag)

`http://www.ags.uni-sb.de/~cp/p/d/welcome.html`

June 30, 2004

1 General Information

1.1 Applicant

- Dr. Claus-Peter WIRTH
- Research and Teaching Fellow (BAT IIa)
- * April 18, 1963, German
- Fachrichtung Informatik
  Universität des Saarlandes
  D–66123 Saarbrücken
  Tel.: 0 681 302 4608
  Fax: 0 681 302 5076
  E-mail: `wirth@logic.at`
- Private: Brandenburger Str. 42
  D–65582 Diez
  Tel.: 0 6432 5961
  Fax: 0 6432 9249115

1.2 Topic

Mathematical Induction by *Descente Infinie*

1.3 Code Word

Descente Infinie

1.4 Scientific Discipline and Field of Work

Artificial Intelligence, Formalized Mathematics, Mathematics Assistance Systems, Human-Oriented Interactive Theorem Proving, Automated Theorem Proving

1.5 Scheduled Total Duration

4 years

1.6 Application Period

2 years

1.7 Proposed Begin of Funding Period

October 1, 2004

1.8 Summary / Zusammenfassung

1.8.1 Summary

Unlike *explicit induction*, mathematical induction by *descente infinie* does not enforce an early commitment (i.e. an eager strategy) to a fixed induction ordering and to a fixed set of induction hypotheses. The work schedule of the project *Descente Infinie* is to integrate utilities for proof development by mathematical induction into the assistance system ΩMEGA,[1] to evaluate the feasibility of our approach to *descente infinie*,[2] and to find the extensions necessary in practice. Regarding the development and presentation of complex induction proofs, the ultimate goal of this integration is to demonstrate that computer assistance can not only satisfy the requirements of a working mathematician on usability, flexibility, and modularity, but even improve the current mathematical practice in certain aspects.

As complex induction proofs are only one of many challenging aspects of mathematical activity tackled by the ΩMEGA research group,[3] the project is to be scientifically and socially embedded into a wider context. The vision of a powerful mathematical assistance environment which provides computer-based support for most tasks of a mathematician has recently stimulated new projects and international research networks across the disciplinary and systems boundaries. Examples are the European CALCULEMUS[4] and MKM[5] initiatives, the EU projects MONET,[6] OPENMATH,[7] and MOWGLI,[8] and the American QPQ[9] repository of deductive software tools.

1.8.2 Zusammenfassung

Im Gegensatz zur *expliziten Induktion* zwingt die mathematische Induktion mittels *Descente Infinie* nicht zur frühzeitigen Festlegung einer bestimmten Induktionsordnung und einer bestimmten Auswahl an Induktionshypothesen. Das Arbeitsprogramm des Projektes *Descente Infinie* besteht in der Integration von Dienstprogrammen zur Entwicklung von mathematischen Induktionsbeweisen in das Assistenzsystem ΩMEGA, der Evaluation der Umsetzbarkeit unseres Ansatzes zur Behandlung der *Descente Infinie* und der Erforschung der in der Praxis erforderlichen Erweiterungen. Bezüglich der Entwicklung und Repräsentation komplexer mathematischer Induktionsbeweise soll diese Integration schließlich nachweisen, dass eine Computer-Unterstützung nicht nur die Anforderungen an Brauchbarkeit, Flexibilität und Modularität im mathematischen Alltag erfüllen kann, sondern sogar in der Lage ist, die gegenwärtige mathematische Praxis in einigen Punkten zu verbessern.

[1] http://www.coli.uni-sb.de/sfb378/projects.phtml?action=2&w=14
[2] Cf.. Wirth (2004a)
[3] http://www.ags.uni-sb.de/~omega/
[4] http://www.calculemus.org
[5] http://monet.nag.co.uk/mkm/index.html
[6] http://monet.nag.co.uk/cocoon/monet/index.html
[7] http://www.openmath.org/cocoon/openmath/index.html
[8] http://www.mowgli.cs.unibo.it/
[9] http://www.qpq.org

2 State of the art, Own preliminary work

Building on the ΩMEGA proof assistant, currently under development at Saarland University, the objective of the project is to investigate the realizability of *computer-assistance* (Section 2.1.1, Section 2.2.1) for complex mathematical induction proofs (Section 2.1.2, Section 2.1.3, Section 2.2.2).

2.1 State of the art

2.1.1 Computer-Assisted Mathematical Theorem Proving

Many mathematicians use their computers just as improved typewriters, with the following exceptions, roughly arranged from successful to less successful computer support: typesetting, electronic publishing, information retrieval, numerical and computer algebra computation, mathematical databases, theory development, and theorem proving.

The low acceptance of computer assistance in theorem proving among mathematicians suggests to look at the situation from a mathematician's point of view before we describe the state of the art in more detail.

Theory Development[10] Some mathematicians use typesetting systems to develop their theories and proofs in more detail after a first raw pen and paper sketch, because they can replace names and symbols systematically and document the development process—even if this may only mean to keep a failed proof attempt or a previous version of a definition or theorem as a comment in the LaTeX source file. This may, however, be very time consuming when parts of the typesetting turn out to be irrelevant for the final presentation. More crucially, however, 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, resulting in the famous *eureka* steps, which puzzle the freshmen in mathematics. This is not only the case for 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 whole development process can be improved. It should be possible to generate different presentations for proof search, education, and succinct documentation for different purposes from a general-purpose internal deep representation, from which the surface representation is computed.

[10] Note that here we use the term "theory development" to refer to the development of new theories by a small group of mathematicians and not to the encoding of existing theories for joint use of the whole mathematics community in huge mathematical databases such as MBASE, cf. `http://www.mathweb.org/mbase`.

Theorem Proving Very few mathematicians use proof checkers or interactive theorem provers (with some degree of automation) for checking those parts of their proofs of important theorems where soundness is a problem. Typically, these are the parts which are error-prone but rather *boring and complex*, involving lengthy argumentation and complicated case analysis.

However, today's theorem provers are not well integrated into the mathematician's daily work. Instead of supporting the mathematician, the mathematician has to support the system: Strong automation support is inherently sensitive to syntactic presentation style. For the more interactive large-scale organization of the proofs this situation has to be improved and our goal is to satisfy the following requirements on computerized proofs:[11]

Modularization via small lemmas Instead of a single huge proof graph for a theorem, a working mathematician needs a forest where each lemma has its own proof tree and the trees are connected by lemma applications. As mathematical practice shows, it should be irrelevant for applying a lemma whether it already has a proof or whether it is still open and to be proved later. Application of induction hypotheses must be possible in a similarly unrestricted fashion.

Overcoming Bourbakism Besides standard requirements on accessibility, the systems' functionality has to use the language of the working mathematician.[12] While the *vagueness* of the natural language of mathematical communication is an essential quality,[13] the implementational flavor[14] of formal logic puts most mathematicians off.

A mathematical proof is usually designed top-down and the refinement stops at different levels, typically before reaching the level of a basic calculus. Such a design is not possible with most theorem provers because they require some bottom-up over-formalized development in the style of Bourbaki (1954). Obviously, a more flexible top-down proof development at abstract levels would be more appropriate. Only for checking a proof with an external proof checker do we have to go down to the level of a basic calculus and deal with all the tiny logical details when automation fails.[15]

[11] I could not find information on the subject published by mathematicians who are not professionally occupied with computers. Chang (2004), however, nicely describes the confusion about computer proofs in the mathematics community. The listed requirements are based on my own experience and private discussions with many mathematicians, theoretical computer scientists, logicians, system developers, and students.

[12] This language, for example, includes higher-order logic but does not apply the special logic jargon and notation of the higher-order logic community.

[13] For example, mathematicians successfully use the language of set theory without specifying which axiomatization of set theory they actually refer to.

[14] Formal logic and set theory were designed for formalization and consistency proofs. Thus, besides the structure, the concrete implementation matters and isomorphic structures must not be identified in general, contrary to most other areas of mathematics.

[15] In general, however, this is most expensive and too tedious to be accepted by mathematicians. To go down to the level of a basic calculus for external proof checking may be appropriate in software verification. Even there, we should be aware of the fact that epistemologically this procedure does not protect us from a possible inconsistency of logical systems (cf. Gödel (1931), Wittgenstein (1939)) and practically the translation of the theorem and the integration of the parts into a whole physical system is still a sufficient source for errors! Thus, the goal should be to find as many errors as possible with the given resources. In mathematics, this has always been the objective and the experienced mathematician knows very well how formal he has to get in which parts of his proofs. Actually, this is one of the most astonishing abilities: to realize the point where the semantic view on the problem gains a gestalt which makes further formalization superfluous. Thus, the inherent problem of high costs of stepping down

Implicitness of calculi and proper notions of their completeness As logical calculi were originally designed for defining theories, the standard notions of completeness are not sufficient for proof search: It is not just the existence of *some* proof we are interested in, but exactly the proof the mathematician has in mind must be realizable within the system in an appropriate and recognizable form.

Moreover, we do not consider it to be appropriate that a mathematics assistance system coerces working mathematicians into thinking explicitly in terms of formal logic calculi during their meta-level proof constructions.

Systems Overview Already in the 1970s, Benthem Jutting (1977) formalized Landau (1930) (a supplement to Calculus textbooks) and passed the code through the non-interactive proof checker AUTOMATH.[16] The main interactive systems in which substantial theories have been represented are the following: The higher-order interactive theorem proving environment HOL[17] in the logical framework Isabelle.[18] However, just like Mizar,[19] they suffer from Bourbakism as described above. PVS[20] is very useful for program verification, but it is still too much oriented toward implementation and its type system fascinates more the logicians than the mathematicians. The intuitionistic logic systems Coq[21] and Nuprl[22] are not in the scope of two-valued logics used exclusively by the vast majority of working mathematicians on the meta level. For a more detailed discussion of these systems cf. Barendregt (2003). In any case, these systems are poor in supporting top-down proof development and *descente infinie*.

2.1.2 *Descente Infinie*

In everyday mathematical practice of an advanced theoretical journal the frequent inductive arguments are hardly ever carried out explicitly, but 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* (Hilbert & Bernays (1968/70), Vol. II) or Gentzen's *Hauptsatz* (Gentzen (1935)), or confluence theorems like the one in Gramlich & Wirth (1996) still require considerable ingenuity and *will* be carried out! The experienced mathematician engineers his proof roughly according to the following pattern:

---

to the level of a basic calculus can be overcome for our project by relying on the mathematician's abilities and not enforcing a complete low-level logic formalization.

[16] `http://www.cs.kun.nl/~freek/aut/`
[17] `http://www.cl.cam.ac.uk/Research/HVG/HOL/`
[18] `http://www.cl.cam.ac.uk/Research/HVG/Isabelle/`
[19] `http://www.mizar.org`
[20] `http://pvs.csl.sri.com`
[21] `http://coq.inria.fr/`
[22] `http://www.cs.cornell.edu/Info/Projects/NuPrl/nuprl.html`

> 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 wellfounded ordering in which all the instances of the conjecture he has applied as an induction hypothesis are smaller than the original conjecture itself.

The hard problems in these induction proofs are

(i) to find the numerous induction hypotheses (as, e.g., to eliminate the Cut in the proof of Gentzen's Hauptsatz) and

(ii) to construct a wellfounded ordering that satisfies the ordering constraints of all these induction hypotheses in parallel (which was, e.g., the hard part for Wilhelm Ackermann in the elimination of the ε-formulas in the proof of the first ε-theorem).

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. He called it *"descente infinie ou indéfinie"*. Here it is in modern language: A proposition Γ can be proved by *descente infinie* as follows:

> *Find a wellfounded ordering* $<$ *and show that for each assumed counterexample of* Γ *there is another counterexample of* Γ *that is smaller in* $<$.

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 Hippasos of Metapont (Italy) in the middle of the 5^{th} century B.C., cf. Fritz (1945). Moreover, we find many occurrences of *descente infinie* in the famous collection "Elements" of Euclid of Alexandria, cf. Euclid (ca. 300 B.C.). 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 17^{th} century, it remained the standard induction method of working mathematicians until today.

2.1.3 Automated Theorem Proving by 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 (Robinson (1965)), 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 reduce the second-order Theorem of Nötherian Induction and the inductive Method of Descente Infinie to first-order *induction axioms* and purely deductive first-order reasoning. The so-called "waterfall"-method of the pioneers of this approach (Boyer & Moore (1979)) refines this process into a fascinating heuristic, and the powerful inductive theorem proving system NQTHM (Boyer & Moore (1988)) shows the success of this reduction approach. For comprehensive surveys on explicit induction cf. Walther (1994) and Bundy (2001).

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 syntactical induction orderings. While we are not going to discuss implicit induction here (cf., however, Wirth (2005) for a survey), it seems to be necessary to distinguish *descente infinie* from the mainstream work on explicit induction.

Modern explicit induction systems such as INKA (Autexier &al. (1999)) or ACL2 (Kaufmann &al. (2000)) easily outperform even a good mathematician on the typical inductive proof problems that arise in his daily work or as subtasks in software verification. However, although explicit induction has become a standard in education (cf. the ✓ERIFUN project,[23] Walther & Schweizer (2003)) and there is still evidence for considerable improvements over the years (cf. Hutter & Bundy (1999)), these methods and systems do not seem to scale up to hard mathematical problems, and we believe that there are *principle reasons* for this shortcoming.

Explicit induction unfortunately must solve the hard problems (i) and (ii) mentioned above 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 very difficult proofs in advance. Although the techniques for guessing the right induction axiom by an analysis of the syntax of the conjecture and of the recursive definitions are perhaps the most developed and fascinating applications of heuristic knowledge in artificial intelligence and computer science, even the disciples of explicit induction admit the limitations of this *recursion analysis*. In Protzen (1994), p. 43, we find not only small verification examples already showing these limits, but also the conclusion:

> "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" and the label "lazy induction" that was coined in this context are nothing but a reinvention of Fermat's *descente infinie* by the explicit-induction community.

Descente infinie and explicit induction do not differ in the task (establishing inductive validity) 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. While explicit induction remains the method of choice for routine tasks, it is an obstacle to progress in the automation of difficult proofs, where the proper induction axioms cannot be guessed in advance.

[23] `http://www.inferenzsysteme.informatik.tu-darmstadt.de/verifun/`

2.2 Own preliminary work

2.2.1 Computer-Assisted Mathematical Theorem Proving

Theory Development This is the subject of collaboration between the $\TeX_{\text{MACS}}$ project[24] of Joris van der Hoeven and the ΩMEGA group, cf. Fiedler (2004). Although this collaboration is not an immediate part of our project *Descente Infinie*, it will easily provide a powerful graphical user interface to our system and, vice versa, our system is to provide data on proof states and proof histories for each of the required presentation styles, namely for proof search, education, and succinct presentation.

Modularization via small lemmas This modularization—as described in Section 2.1.1—is already implemented in the QUODLIBET system,[25] cf. Avenhaus &al. (2003), but restricted to first-order clausal logic and tactic-based automation. For the more general proof-planning framework of the ΩMEGA system, cf. Siekmann &al. (2002), the theoretical problems are solved in Wirth (2004a) and for the practical and heuristic problems a *task interface* has been proposed in the ΩMEGA group, cf. Wirth &al. (2004).

Overcoming Bourbakism Last winter we gave an introduction to human-oriented theorem proving in a lecture course (cf. Wirth &al. (2003)) with the explicit goal to overcome the logic jargons of the various research communities and their traditional formalist presentation. This has attracted excellent master's students to the ΩMEGA group and the experience and material will serve as a source for tutorials and guidelines for system interfaces.

Proof development in the ΩMEGA system, cf. Siekmann &al. (2002), follows the top-down development approach of the working mathematician and does not enforce a complete low-level logic formalization.

Implicitness of calculi and proper notions of their completeness The design of human-oriented calculi for proof search is a more recent development, cf. e.g. Wirth (1997), Giese (1998), Kühler (2000), Autexier (2003), Wirth (2004a), Wirth (2004e). Cf. Section 3.2.2 for more on this.

[24] `http://www.texmacs.org/index.php3`

[25] The current version of QUODLIBET can be obtained from `http://agent.informatik.uni-kl.de.quodlibet.html`

2.2.2 *Descente Infinie*

Looking for a formal inductive calculus for *descente infinie*, the "implicit induction" of Bachmair (1988) (as implemented in UNICOM, Gramlich & Lindner (1991)) was a starting point for us, because it includes an explicit step for lazy induction-hypothesis application; but its calculus is machine-oriented and restricted to first-order universally quantified unconditional equations. These limitations were so severe in practice that we decided to stop the development of UNICOM and use this experience for the development of the QUODLIBET system based on a human-oriented inductive calculus for first-order universally quantified clausal logic, cf. Wirth (1997), Kühler (2000), Avenhaus &al. (2003). We have used QUODLIBET for proofs on natural numbers (non-terminating specification of division), sorting algorithms on lists (bubble-sort, insertion-sort, merge-sort, quick-sort), binary search trees (cf. Kühler (2000)), orderings (lexicographic path ordering), and the irrationality of $\sqrt{2}$ in the style of Hippasos of Metapont. Note that the latter two examples cannot be appropriately handled with systems based on explicit induction, such as INKA and ACL2; indeed, the first involves complex mutual inductions over an ambiguous and non-terminating defining case analysis, and the second applies a *descente infinie* which does not rely on recursion analysis at all. In Kühler (2000), the concepts and the implementation of the QUODLIBET system are presented in detail. Very recently, the underlying standard tactics were again considerably improved, cf. Schmidt-Samoa (2004). An intelligent and flexible combination with decision procedures is in progress at the University of Kaiserslautern. Disregarding run times and decision procedures, the QUODLIBET system has about the same level of automation by recursion analysis as the leading explicit induction systems; and indeed, all the heuristic knowledge and automatization of the field of explicit induction are still applicable and indispensable in our framework of *descente infinie*. The possibility to be lazy, however, simplifies things when different induction schemes are in conflict (compare Walther (1992) with Kühler (2000), Section 8.3). Another advantage is that—when automation fails—QUODLIBET stops early and presents the state of the proof attempt in a human-oriented form. For the project *Descente Infinie*, it is important that the whole set of standard tactics for automation and recursion analysis is available in the higher-level tactic programming language QML (QUODLIBET Meta Language).

In Wirth (2004a) we have shown how to integrate *descente infinie* into state-of-the-art free-variable sequent and tableau calculi which are well-suited for an efficient interplay of human interaction and automation. The semantical requirements are satisfied for a variety of two-valued logics, such as clausal logic, classical first-order logic, and higher-order modal logic. This new formal basis can provide a flexible support for a mathematician searching for hard induction proofs.[26]

[26] For example, our comprehensive integration of *descente infinie* differs from the "lean" calculus of Baaz &al. (1997) in the following aspects: We can have mutual induction and variable induction orderings. Our induction hypotheses can be arbitrary sequents instead of a single preset literal. Finally, we can also generate induction hypotheses eagerly in the style of explicit induction, which enables goal-directedness w.r.t. induction hypotheses.

3 Goals and Work Schedule

The work schedule of the project *Descente Infinie* is to integrate utilities for proof development by mathematical induction into the assistance system ΩMEGA,[27] to evaluate the feasibility of our approach to *descente infinie*,[28] and to find the extensions necessary in practice. Regarding the development and presentation of complex induction proofs, the ultimate goal of this integration is to demonstrate that computer assistance can not only satisfy the requirements of a working mathematician on usability, flexibility, and modularity, but even improve the current mathematical practice in certain aspects.

3.1 Goals

To achieve a practically useful mathematics assistance system we aim at an *interactive* system with a high degree of automated support. To combine interaction and automation into a synergetic interplay is a very difficult and complex task. It requires sophisticated achievements from logic, tactics programming, proof planning, agent-based approaches from artificial intelligence, graphical user interfaces, and connection to external reasoning tools on the one hand, and a deeper experience in informal and formal human proof development on the other hand. Obviously, the development of a state-of-the-art mathematics assistance system is a huge enterprise, requiring expertise from many different fields. This enterprise is approached by a functional extension of the current ΩMEGA system and by a re-implementation of its logic engine within the OMEGA project of the Collaborative Research Center SFB 378 "Resource-Adaptive Cognitive Processes", cf. Wirth (2004c). Very recently, this project has been evaluated successfully for a final funding period.

Although *encoding* of proofs by mathematical induction is possible in any higher-order logic system such as the simply-typed λ-calculus of Alonzo Church used in the ΩMEGA system, neither the current ΩMEGA system nor its planned extension include any convenient support for the actual *development* of proofs by mathematical induction.

Why does a higher-order theorem prover (such as the ΩMEGA system) need this additional support for mathematical induction?

Inductive theorem proving (typically in the form of explicit induction) has been a standard extension to first-order deductive theorem proving for a third of a century now. Obviously, the notions of wellfoundedness and inductive definition, Peano's axioms and similar axioms for structural induction, and the Theorem of Nötherian induction are all of second order. This seems to be the origin of the popular belief that higher-order theorem proving does not need to be augmented with special features for inductive theorem proving. It is already a motivation for higher-order theorem proving in addition to first-order theorem proving, however, that the existence of *some encoding* for any possible proof [29] is insufficient. Instead, the two following essential requirements on proof development are to be respected:

[27] `http://www.coli.uni-sb.de/sfb378/projects.phtml?action=2&w=14`
[28] Cf.. Wirth (2004a)
[29] Note that this does not mean completeness w.r.t. a semantics but only completeness w.r.t. other calculi.

(A) For the human-assisted actual construction of proofs, the proofs have to exist in the special informal structure *intended* by the users of the system.

(B) For a high level of automation support, the proofs have to exist in the special syntactical forms needed to *represent* the data structures of the heuristic procedures.

In spite of the more expressive language, the requirements (A) and (B) apply to proof construction in higher-order logic just as in first-order logic. Therefore—as induction proofs are ubiquitous in mathematics—a special support for induction in higher-order theorem-proving systems is indispensable for their successful application to mathematical practice.

Luckily, our approach to *inductive* theorem proving by *descente infinie* is mostly orthogonal to the approach to *deductive* theorem proving pursued in the new logic engine of the ΩMEGA system; and the overlapping parts are closely matching each other.[30] Moreover, it is advantageous that COMMON LISP is the common implementation language of ΩMEGA, CORE, and QUODLIBET.

3.2 Work Schedule

From the discussion in the previous section it should have become clear that the features for adding *descente infinie* to the logic engine of the ΩMEGA system can and have to be realized within our project *Descente Infinie* in close cooperation with the specification (Work Package I, ca. 2 months) and the re-implementation (Work Package II, ca. 18 months) of the logic engine for the deductive system parts of the ΩMEGA system in the OMEGA project. The inclusion into the graphical user interface (Work Package III, ca. 4 months) is to be supported by the collaboration between the OMEGA and the $\mathrm{T_EX_{MACS}}$ project, where the project *Descente Infinie* has to provide the different presentation forms of induction proofs in the form of a logical history. During the second year of the application period (2 years) we want to start modeling complex induction proofs with our system (Work Package IV) and to integrate natural language into our formulas, profiting from the DIALOG project of the Collaborative Research Center SFB 378 "Resource-Adaptive Cognitive Processes", cf. Wirth (2004c). Very recently, also this project has been evaluated successfully for a final funding period. In the second period (another 2 years), the feasibility of our approach to *descente infinie* is to be evaluated and the extensions necessary in practice have to be found and realized. Finally, proof planning methods for *descente infinie* have to capture the experience and the knowledge gained by the system developers.

[30] Note that it would be very difficult to realize our modeling of *descente infinie* within the current ΩMEGA system due to its natural deduction calculus, which does not provide the flexibility we need and whose concept of assumption is in conflict with our concept of induction hypothesis. The reason that the planned new logic engine fits *descente infinie* much better than the old system is partly due to a very close collaboration during the last three years. Moreover, comparing Wallen (1990), Giese (1998), Wirth (1999), Kühler (2000), and Autexier (2003), it seems that the experience with the design of theorem provers that are capable of realizing non-automatable complex proofs evokes similar concepts and solutions in researchers with different background and different focus.

3.2.1 Work Package I: Task Interface

A *task* can be seen as a proof obligation, a proof line with logical context, a sequent, or a lemma we want to prove from a set of assumptions. A task may contain different kinds of free variables to be instantiated appropriately during the proof construction. It is only via the rigid instantiation of these free variables that a task interacts with or threatens other parallel tasks. Thus, a task can be seen as a somewhat independent proof goal.

The *task interface* is going to be a major interface in the running re-implementation of the logic engine of ΩMEGA system, cf. Wirth &al. (2004). Roughly speaking, it is to separate the heuristic part of the theorem prover from its high-level *logic engine* in the broader sense. The heuristic part consists of user interaction, reasoning agents, tacticals, and proof planning. The logic engine is the part that executes proof steps in the proof forest and reports and keeps track of the justifications and soundness conditions of these steps. Thus, all actions in proof construction and presentation have to communicate with the logic engine via this interface.

The task interface is a non-trivial software engineering problem: The central role of this interface in the new ΩMEGA system requires a high degree of stability and, for its definition, the expertise of the whole ΩMEGA group.

Descente Infinie For proof construction and presentation by *descente infinie* the planned task interface of the ΩMEGA system has to be extended with the possibility to apply induction hypotheses. This basically means the addition of

(1) a weight term to each task,

(2) a new action to apply tasks as induction hypotheses to other tasks, and

(3) a decision algorithm for the inductive closedness of a proof tree in a proof forest.

For (1) we have to extend the task data structure with a higher-order free γ-variable (or "meta-variable") of variable result type which gets the free δ^--variables of the task as arguments. The action for (2) is similar to lemma application but generates additional subgoals that guarantee the termination of the inductive reasoning cycles. Finally, the realization of (3) in Work Package II will be quite complex as our proof trees are and-or trees and we have to check for the existence of an acyclic justification for closedness. The decision procedure is described in Wirth &al. (2004) and its run-time behavior turned out to be uncritical in the QUODLIBET system. For the general details cf. Wirth (2004a).

Liberalized δ-rule and Hilbert's epsilon Another extension of ΩMEGA's task interface to be realized within our project *Descente Infinie* is the addition of free δ^+-variables to the logic language with the following two objectives.

On the one hand, these free δ^+-variables will enable the ΩMEGA system to apply liberalized δ-steps (δ^+-steps, cf. Hähnle & Schmitt (1994)) according to the classification of Smullyan (1968). Contrary to black-box automated theorem proving, our motivation for δ^+-steps is not so much the existence of non-elementarily shorter proofs due to smaller variable-conditions and a more careful consideration of the dependency of variables. Instead, our motivation is that

δ^+-steps frequently occur in the proofs of working mathematicians: Humans typically do not increase multiplicity or apply a lemma more often than necessary just to compensate for a sloppy dependency consideration. Indeed, sophisticated lazy dependency considerations are a standard element of mathematical proofs, typically starting with a sentence such as "Finally, we have to show that it is possible to choose *a* independently from *b* and *c*."

On the other hand, the free δ^+-variables will supply the ΩMEGA system with a powerful choice operator, namely our new variant of Hilbert's ε-operator, cf. Wirth (2002). Of all known variants, this one is best-suited for proof search because a free δ^+-variable representing a *committed choice* for an ε-term can be globally replaced with *any* term satisfying the choice-condition. In this aspect of proof search, the free δ^+-variables behave similarly to the free γ-variables (or "meta"-variables). Moreover, our ε-operator comes for free with our approach to *descente infinie* and the liberalized δ-rule, its descriptive power offers a multitude of additional applications, and its operationalization is closely integrated into the logical system.

For more details cf. Wirth (2002) and Wirth (2004a).

3.2.2 Work Package II: Logic Engine

The *logic engine* is to be a high-level human-oriented inference engine in the wider sense, realizing the task interface of Work Package I. It executes proof steps in the proof forest and reports and keeps track of the justifications and conditions on soundness and safeness of these steps. Moreover, it provides the information on the current state, focus, and context of the proofs and the possible proof steps to the calling programs and human users.

The basis of the logic engine is a high-level human-oriented calculus for proof search with strong automation support. The call for and development of such calculi has occurred only quite recently, cf. e.g. Wirth (1997), Giese (1998), Kühler (2000), Autexier (2003), Wirth (2004a), Wirth (2004e). These calculi are to overcome the following two problems:

- A low-level standard calculus which is not optimized for proof search tends to export low-level tasks to higher levels of abstraction. These low-level tasks have turned out to be problematic in practice because they can neither be ignored nor properly treated on the higher levels. For example, the proof planning (high level) in the old ΩMEGA system severely suffers from its commonplace natural deduction calculus (low level) in many aspects.

- If automation fails on the lowest level, the human user must be given a fair chance to deal with the basic calculus and to understand what the problem is.

In the field of human-oriented calculi for proof search the design goals are the following:

1. compliance with natural human proof techniques,
2. support of the user in stating his proof ideas, and
3. no difficulties in understanding and searching for proofs represented with the system at any level.

Automation will fail from time to time even on the deepest logic level. Thus, even the deepest level within the system has to be based on a *human-oriented* calculus, such as the ones in Autexier (2003), Wirth (2004a). Note that—contrary to popular belief that *nomen est omen*—we do not consider *natural deduction* calculi for classical logic to be human-oriented. As explained in Wirth (2004a), they are not particularly well-designed w.r.t. the design goals listed above and not well-suited for the application of induction hypotheses.

The human-oriented calculi for *descente infinie* in two-valued logics of Wirth (2004a) provide inference rules for all natural proof steps and combine a homogeneous representation of all proof tasks with 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. The calculus complies with natural human proof techniques. It supports the human users in stating their proof ideas and in understanding and searching for proofs represented in the calculus.

The CoRe system of Autexier (2003) (developed for the running re-implementation of the logic engine of the ΩMEGA system) goes one crucial step ahead of this in the following aspect: It basically frees the users from thinking in terms of any calculus at all: A semantic understanding of the language together with the idea of rewriting equals by equals is sufficient to control the proof search interactively. A more sequent-based solution to interaction without awareness of a calculus is sketched in Wirth (2004e) and has the advantage that it might simplify the migration of heuristic knowledge from the sequent-based QUODLIBET system.

Technically, the CoRe system combines a matrix calculus for two-valued logics based on indexed formula trees in the style of Wallen (1990) with window-based focusing techniques à la Monk (1988), Robinson & Staples (1993). The human user can set a focus on the crucial subformula and the logical context is provided in the form of a set of rewrite rules. Thereby it is—at least interactively—possible to model the experienced working mathematician's practice to jump immediately *in medias res* (i.e. to the crucial middle part of a typical calculus proof) and to defer the less crucial decisions to later refinement. For example, the lim+-example (limit of sums is sum of limits) in Wirth &al. (2003) shows how indexed formula trees admit to start with the crucial estimate by the inequation *before* choosing the only successful β-step sequencing. In any tableau calculus this has to be done in the reverse order, which means blind guessing and typically wrong commitments, cf. Wirth &al. (2003). While means to find the proper β-sequencing and -downfolding are still lacking in the ΩMEGA system, the proof plan it currently constructs for lim+ concentrates on the estimate and—just as a working mathematician—defers the β-sequencing and -downfolding, which is the second main part of the high-level mathematical problem besides the estimate.

In close cooperation and collaboration with Serge Autexier regarding the implementation of the new ΩMEGA system in the OMEGA project, we want to change the current implementation of the CoRe system in the following aspects:

(1) Add the weights required for *descente infinie* as indicated in the previous section.

(2) Replace the current representation of the proof state in *two* indexed formula trees with a representation in a *single* indexed formula tree. This is to be achieved by switching from the variable-conditions of Wallen (1990) to the state-of-the-art ones of Wirth (2004a).

(3) Split this huge single indexed formula tree into several small ones, one for each single task.

(4) Connect these small indexed formula trees with links for application of axioms, lemmas, and induction hypotheses.

(5) Enable the *bound* δ^-- and γ-variables to act as *free* δ^-- and γ-variables for inter-task and external systems communication.

(6) Improve the facilities to exploit the advantage of matrix over sequent or tableau calculi to find the proper order of β-steps and the proper β-downfolding directions.

(7) Add the free δ^+-variables with the functionality described in the previous section by a general mechanism enabling the addition of further conditioned variables for capturing the semantics and operationalization of reference constructs in natural language.

(8) Improve the rewrite-rule presentation of the context information provided for the subformula focused on.

(9) Add an interface realizing the QUODLIBET inference machine.

Contrary to Work Package I, most of these items are research problems and no engineering tasks. Thus, it may well be that we fail to solve them in time and have to look for alternative solutions. While it is not possible to discuss the planned alternative solutions in detail here, let us have a closer look at the most problematic item:

Although the considerations in Wirth (2004e) may help, it is unclear how to simulate navigation in sequent-calculus proof trees in indexed formula trees. This problem has to be solved for (9), the realization of the QUODLIBET inference machine. As its roughly forty functions provide the whole functionality required by the library of tactics in QUODLIBET meta language after compilation, this realization would immediately provide us with a state-of-the-art automation support based on advanced recursion analysis. Moreover, we believe that this is also the right level of integration: Naïvely, one would realize the tactics for automation by recursion analysis on ΩMEGA's proof planning level to take advantage from a declarative description and a synergetic interplay with other methods for proof planning. The experience gained from investigations to improve recursion analysis such as Walther (1992), Schmidt-Samoa (2004) is, however, that the practical testing finally shows that the procedural character of this heuristic knowledge cannot be successfully captured by abstraction and declarative description. It is also for this reason, that the proof planning approach of the OYSTER-CLAM system (cf. Bundy &al. (1990)) fundamentally differs by its more procedural character from the knowledge-based proof planning of the ΩMEGA system (cf. Melis & Siekmann (1999)).

If we do not succeed in realizing the QUODLIBET inference machine as an interface to the CORE system, the best alternative is to write a new compiler for QUODLIBET meta language based on the refined CORE system. Although—exactly for this reason—the QUODLIBET meta language has been designed to be very simple, the procedural character of the heuristic knowledge

may not admit of such a compiler. In that case, we can either complete the rudimental integration of the explicit-induction theorem prover INKA (cf. Autexier &al. (1999)) into CORE, or otherwise loosely integrate QUODLIBET or INKA into the new ΩMEGA system using ΩMEGA's standard integration mechanisms for external systems.

3.2.3 Work Package III: Integration into User Interfaces

The modeling of complex induction proofs requires a sophisticated graphical user interface. This is not only the case for the development of proof planning methods and the evaluation of our approach to *descente infinie* by confronting working mathematicians with the system in a second period, but already for the realization of the system developer's paradigmatic examples of Work Package IV. The example proof of the generalized Newman Lemma in Section 3.4 of Wirth (2004a) is nearly the limit of complexity we can handle without a system support besides LaTeX.[31] Nevertheless, experience gained with the QUODLIBET system is that the possibility of modeling proofs with a command or task interface is hardly possible and even the developers of the interface themselves get hopelessly lost in counting term positions of bigger terms.

The current graphical user interfaces of the ΩMEGA (LΩUI) and the QUODLIBET system are a great help in this context. But from our experience with them it has become clear that they still fall short of what is needed for a system to be really useful in practice. The development of a better graphical user interface is the subject of collaboration between the TeXmacs project[32] of Joris van der Hoeven and the ΩMEGA group, cf. Fiedler (2004). The TeXmacs system—soon to be adapted by major publishing companies such as Springer—improves the already very high typesetting standard achieved in TeX. Moreover, the user communicates with the TeXmacs system over a WYSIWYG graphical user interface. Important in our context is that, while TeX carefully models the human typesetting process, TeXmacs focuses on the logical structure of information, no matter whether this information comes from the users or from a program. After adding the missing features for sophisticated communication with mathematics assistance systems TeXmacs system, the ΩMEGA system will have an excellent graphical user interface realized with the TeXmacs system by the end of next year. Although the construction of this graphical user interface is not part of our project *Descente Infinie*, it will be a simple standard programming task to include the basic features for *descente infinie* into this new interface.

The structured proof presentation achieved by this will be sufficient for inspecting states of proof attempts in proof search and for interactive teaching. For the succinct presentation of proofs in mathematical proceedings and journals (which is the ultimate goal of the TeXmacs / OMEGA cooperation), this form of presentation is inappropriate. For example, the structure of the final presentation of the complex induction proof of Gramlich & Wirth (1996) does not at all reflect its development history. The presentation in the proceedings not only omits irrelevant trials and easily reconstructible parts, but also starts with the presentation of the induction ordering in an *eureka* step, which was the very last step in the development history of the proof. Contrary to the encoding of mathematical induction in higher-order logic formulas, with our direct approach to model the *descente infinie* of a working mathematician we can easily perform such proof history

[31] Note, however, that in Section 3.4 of Wirth (2004a) we had to go down to the level of the basic calculus to show its human-orientedness. For the modeling of the proofs within our project *Descente Infinie* this will not be a must.

[32] `http://www.texmacs.org/index.php3`

transformations and will present the results to the $\TeX_{\text{MACS}}$ system in the format of the successor to the *P.rex* system of Armin Fiedler, cf. Fiedler (2001a), Fiedler (2001b), Fiedler (2004).

3.2.4 Work Package IV: Modeling Complex Induction Proofs

Proof development in the ΩMEGA system, cf. Siekmann &al. (2002), follows the *top-down* development approach of the working mathematician and does not enforce a complete low-level logic formalization. The *vagueness* of mathematical argumentation (stated in Section 2.1.1 on page 4 as an essential property of high-level mathematical communication) is to be approached with the following two means:

(1) The ΩMEGA system admits arbitrary COMMON LISP expressions as justifications for tasks (or proof steps). These justifications can span a large variation in reliability, ranging from "verified by the logic engine" over "computed by an external system" to "conjectured by a mathematician". Indeed—at the highest level of abstraction—the latter justification can be a meaningful representation of knowledge, even if we have to take the task to which it is assigned as a black box at first because we fail to analyze its semantics.

(2) A lazy approach to the task structure proposed in Wirth &al. (2004) will enable the working mathematician to start with a restricted form of natural language and—if intended—end up in sequents of strongly typed higher-order formulas. To be able to mix formal with informal elements in a mutual recursion, we will initially represent parts marked as natural language (possibly containing parts marked as formal language) as undefined functors.

During the second year of the application period (2 years) we will *start* modeling complex induction proofs such as Hilbert's first ε-theorem and the confluence theorem of Gramlich & Wirth (1996).[33]

In the second funding period—provided that the first phase is successfully evaluated—we want to complete the modeling of the above theorems and then go on with the ones of Wirth (1995), which fall into the class of important[34] inductive theorems with rather *boring and complex* proofs. As already indicated in Section 2.1.1 on page 4, this class constitutes a typical area of application of computer-assisted inductive theorem proving.

To improve the pragmatic treatment of natural language as a black box with holes, we shall profit from the semantic analysis of a mixture of formulas and natural language as proposed in

[33] Contrary to a partial analysis of natural language, there is no way to defer this to a second funding period. It is even not adequate to wait for the completion of Work Packages II and III before starting the modeling of complex induction proofs: Although the modeling will be very troublesome and awkward without the interfaces being completed, it already has to accompany the realization of Work Package II because a practical solution to its research tasks is not possible without some paradigmatic examples. Moreover, note that these paradigmatic examples must not be simple: Otherwise the advantage of *descente infinie* over explicit induction cannot be realized. As our example proofs already exist in a well-written but highly abstract form, it will be possible to have the coding executed by students of mathematics. This has the additional advantage to give us a first early feedback on our goal of human-orientedness. Nevertheless—just as the implementation tasks of Work Packages II and III—the experiments for Work Package IV will result in a very high workload on the students' side of our project *Descente Infinie*.

[34] The theorems in Wirth (1995) are the basis for the very weak admissibility conditions for specifications in the QUODLIBET system, admitting partial, ambiguous, and non-terminating function definitions without sacrificing powerful recursion analysis.

Wirth (2004c) for the very recently successfully evaluated DIALOG project in the Collaborative Research Center SFB 378 “Resource-Adaptive Cognitive Processes”. Moreover, our approach to *descente infinie* (cf. Wirth (2004a)) includes a choice operator, namely a version of Hilbert’s ε-operator, cf. Wirth (2002). Together with its extensible technique of realization by free variables associated with conditions (as planned in step (7) of Work Package II), we hope it to be useful in capturing the referential interpretation of articles and anaphoric pronouns, cf. Heusinger (1997).[35]

3.3 Human Experiments

None.

3.4 Animal Experiments

None.

3.5 Experiments with Recombinant DNA

None.

[35] As our setting, however, is less ambiguous than the DIALOG project’s tutorial setting and includes neither dialogs nor proofs but only tasks, applying some solutions of Zinn (2003) may be more appropriate in our context. Note we are not going to do research in natural language understanding, but only want to apply the results pragmatically in a very restrictive setting, possibly contributing our versions of Hilbert’s ε.

4 Beantragte Mittel / Funds Requested

While the scientific parts of this proposal are written in English for communication with our international partners, the administrative sections are in German (according to private communication with Dr. Gerit P. Sonntag of DFG).

4.1 Personalkosten / Staff

1. *Ein* wissenschaftlicher Mitarbeiter BAT IIa über die gesamte Förderperiode.
2. *Eine* wissenschaftliche Hilfskraft (mit Abschluss) (19 h) über die gesamte Förderperiode zur Unterstützung der schwierigen Implementierungsarbeiten des Work Package II.
3. *Eine* studentische Hilfskraft (ohne Abschluss) (19 h) (Informatikstudent) über die gesamte Förderperiode zur Implementation, zum Testen, zur Dokumentation und zur organisatorischen Unterstützung.
4. *Drei* studentische Hilfskräfte (ohne Abschluss) (19 h) (Mathematikstudenten) *für das zweite Jahr der Förderperiode* zur Unterstützung der Implementierungsarbeiten von Work Package III und zur Beweismodellierung innerhalb des Work Package IV. Zur Begründung dieses besonderen Bedarfs siehe Fußnote 33.

4.2 Wissenschaftliche Geräte / Scientific Instrumentation

Keine.

4.3 Verbrauchsmaterial / Consumables

Keine. (Grundausstattung Prof. Dr. Jörg H. Siekmann)

4.4 Reisen / Travel Expenses

Die folgenden Zahlen beziehen sich auf zwei Jahre *insgesamt*.

Teilnahme an int. wissenschaftlichen Konferenzen (CADE, IJCAI, IJCAR, TABLEAU, TPHOLs, &c.) (4 x 1300 €)	5200 €
Konferenzgebühren (4 x 600 €)	2400 €
Besuche bei Kooperationspartnern	2500 €
Einladung Gastwissenschaftler	1000 €
Summe 4.4	**11100 €**

4.5 Publikationskosten / Publication Costs

Keine.

4.6 Sonstige Kosten / Other Costs

Keine.

5 Preconditions of the Project

5.1 Team

Dr. Serge Autexier	DFKI and Universität des Saarlandes
Dr. Christoph Benzmüller	assistant professor (C2), Universität des Saarlandes
Dr. Henri Lesourd	wissenschaftliche Hilfskraft, Universität des Saarlandes
Prof. Dr. Jörg H. Siekmann	full professor (C4), Universität des Saarlandes
Dr. Claus-Peter Wirth	project leader, Universität des Saarlandes

5.2 Cooperation with other Scientists

- Prof. Dr. Jürgen Avenhaus, Prof. Dr. Klaus Madlener, Tobias Schmidt-Samoa, Fachbereich Informatik, Universität Kaiserslautern, D–67663 Kaiserslautern, Germany, w.r.t. QUODLIBET.
- Prof. Dr. Alan R. Bundy, Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh, Appleton Tower, Crichton St, Edinburgh EH8 9LE, Scotland, w.r.t. proof planning in inductive theorem proving and analysis of mathematical discourse.
- Dr. Joris van der Hoeven, Département de Mathématiques, Université Paris-Sud, 91405 Orsay Cedex, France, w.r.t. $\TeX_{\text{MACS}}$.
- Dr. Dieter Hutter, DFKI GmbH Deduction and Multi-Agent-Systems Stuhlsatzenhausweg 3, D–66123 Saarbrücken, Germany, w.r.t. INKA and induction in general.
- Dr. Ulrich Kühler, sd&m AG, Am Schimmersfeld 7a, D-40880 Ratingen, Germany, w.r.t. QUODLIBET.

5.3 Foreign Contacts and Cooperations

See the previous section. Besides these, the project *Descente Infinie* will profit from the excellent and manifold international connections of the ΩMEGA group.

5.4 Available Equipment

The hardware equipment will be covered by the *Grundausstattung* of Prof. Dr. Jörg H. Siekmann.

5.5 Saarland University's General Contribution

The general contribution for consumables from the Saarland University via the *Grundausstattung* of Prof. Dr. Jörg H. Siekmann will be approximately 1000 €.

5.6 Other Requirements

None.

6 Erklärungen / Declarations

6.1

Ein Antrag auf Finanzierung dieses Vorhabens wurde bei keiner anderen Stelle eingereicht. Wenn ich einen solchen Antrag stelle, werde ich die Deutsche Forschungsgemeinschaft unverzüglich benachrichtigen.

6.2

Der DFG-Vertrauensdozent der Universität des Saarlandes, Prof. Dr. Hartmut Janocha, wurde von dieser Antragstellung unterrichtet.

7 Unterschrift(en) / Signature(s)

Saarbrücken, den 30. Juni 2004

(Dr. Claus-Peter Wirth)

8 Verzeichnis der Anlagen / List of Enclosures

1. CV des Antragstellers inkl. Publikationsliste
2. Aktuelle Publikation des Antragstellers: Wirth (2004a)

Alle Anlagen zum Verbleib.

Acknowledgments: I would like to thank Serge Autexier, Armin Fiedler, Jörg H. Siekmann and Magdalena A. M. Wolska for most useful advice, and Tobias Schmidt-Samoa for some hints.

References

Serge Autexier (2003). *Hierarchical Contextual Reasoning.* Ph.D. thesis, FR Informatik, Saarland Univ..

Serge Autexier, Dieter Hutter, Heiko Mantel, Axel Schairer (1999). INKA *5.0 — A Logical Voyager.* 16th CADE 1999, LNAI 1632, pp. 207–211, Springer.

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.

Matthias Baaz, Uwe Egly, Christian G. Fermüller (1997). *Lean Induction Principles for Tableaus.* 6th TABLEAU 1997, LNAI 1227, pp. 62–75, Springer.

Leo Bachmair (1988). *Proof By Consistency in Equational Theories.* 3rd IEEE Symposium on Logic In Computer Sci., pp. 228–233, IEEE Press.

Henk Barendregt (2003). *Foundations of Mathematics from the perspective of Computer Verification.* Draft. `http://www.cs.kun.nl/ henk/papers.html (May 23, 2004).`

L. S. van Benthem Jutting (1977). *Checking Landau's "Grundlagen" in the* AUTOMATH *system.* Ph.D. thesis, Eindhoven Univ. of Technology. Also as: Mathematical Centre Tract 83, Math. Centrum, Amsterdam, 1979.

Nicolas Bourbaki (1954). *Livre I — Théorie des Ensembles.* Hermann, Paris.

Robert S. Boyer, J S. Moore (1979). *A Computational Logic.* Academic Press.

Robert S. Boyer, J S. Moore (1988). *A Computational Logic Handbook.* Academic Press.

Alan R. Bundy (2001). *The Automation of Proof by Mathematical Induction.* In: Robinson & Voronkov (2001), Vol. 1, pp. 845–911.

Alan R. Bundy, Frank van Harmelen, C. Horn, Alan Smaill (1990). *The Oyster-Clam System.* 10th CADE 1990, LNAI 449, pp. 647–648, Springer.

Kenneth Chang (2004). *In Math, Computers Don't Lie. Or Do They?.* The New York Times, April 6, 2004.

Euclid of Alexandria (ca. 300 B.C.). *Elements.* D. E. Joyce, Dept. Math. & Comp. Sci., Clark Univ., Worcester, MA. `http://aleph0.clarku.edu/~djoyce/java/elements/Euclid.html (March 24, 2003).`

Armin Fiedler (2001a). *Dialog-Driven Adaptation of Explanations of Proofs.* In: Bernhard Nebel (ed.). *Proc. 17th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1295–1300, Morgan Kaufman (Elsevier). `http://www.ags.uni-sb.de/~afiedler/pubs/complete.html (June 30, 2004).`

Armin Fiedler (2001b). *User-Adaptive Proof Explanation.* Ph.D. thesis, Saarland Univ.. `http://www.ags.uni-sb.de/~afiedler/pubs/complete.html (June 30, 2004).`

Armin Fiedler (2004). *Benutzerorientierte, interaktive Erzeugung verifizierbarer mathematischer Dokumente.* In prep., Saarland Univ..

Kurt von Fritz (1945). *Die Entdeckung der Inkommensurabilität durch Hippasos von Metapont.* Annals of Mathematics **46**, pp. 242–264. Also in: Oscar Becker. *Zur Geschichte der griechischen Mathematik*, pp. 271–308.

Dov M. Gabbay, C. J. Hogger, J. Alan Robinson (eds.) (1993 ff.). *Handbook of Logic in Artificial Intelligence and Logic Programming.* Clarendon Press.

Gerhard Gentzen (1935). *Untersuchungen über das logische Schließen.* Mathematische Zeitschrift **39**, pp. 176-210, 405–431.

Martin Giese (1998). *Integriertes automatisches und interaktives Beweisen: die Kalkülebene.* Master's thesis, Univ. Karlsruhe. `http://i11www.ira.uka.de/~giese/da.ps.gz (May 09, 2000).`

Kurt Gödel (1931). *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.* Monatshefte für Mathematik und Physik **38**, pp. 173–198.

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).`

Bernhard Gramlich, Claus-Peter Wirth (1996). *Confluence of Terminating Conditional Term Rewriting Systems Revisited.* 7th RTA 1996, LNCS 1103, pp. 245–259, Springer.

Reiner Hähnle, Peter H. Schmitt (1994). *The Liberalized δ-Rule in Free-Variable Semantic Tableaus.* J. Automated Reasoning **13**, pp. 211–221, Kluwer.

Klaus von Heusinger (1997). *Salienz und Referenz — Der Epsilonoperator in der Semantik der Nominalphrase und anaphorischer Pronomen.* Studia grammatica 43, Akademie Verlag, Berlin.

David Hilbert, Paul Bernays (1968/70). *Grundlagen der Mathematik.* 2nd ed., Springer.

Dieter Hutter, Alan R. Bundy (1999). *The Design of the CADE-16 Inductive Theorem Prover Contest.* 16th CADE 1999, LNAI 1632, pp. 374–377, Springer.

Dieter Hutter, Werner Stephan (eds.) (2005). *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday.* LNAI 2605, Springer.

Matt Kaufmann, Panagiotis Manolios, J S. Moore (2000). *Computer-Aided Reasoning: An Approach.* Kluwer.

Ulrich Kühler (2000). *A Tactic-Based Inductive Theorem Prover for Data Types with Partial Operations.* Ph.D. thesis, Infix, Akademische Verlagsgesellschaft Aka GmbH, Sankt Augustin, Berlin.

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/welcome.html (Aug. 05, 2001).`

Edmund G. H. Landau (1930). *Grundlagen der Analysis (Das Rechnen mit ganzen, rationalen, irrationalen, komplexen Zahlen). Ergänzung zu den Lehrbüchern der Differential- und Integralrechnung.* Leipzig.

Erica Melis, Jörg H. Siekmann (1999). *Knowledge-Based Proof Planning.* Artificial Intelligence **115**, pp. 65–105.

Leonard G. Monk (1988). *Inference Rules Using Local Contexts.* J. Automated Reasoning **4**, pp. 445–462, Kluwer.

Martin Protzen (1994). *Lazy Generation of Induction Hypotheses.* 12th CADE 1994, LNAI 814, pp. 42–56, Springer. Long version in: Protzen (1995).

Martin Protzen (1995). *Lazy Generation of Induction Hypotheses and Patching Faulty Conjectures.* Ph.D. thesis, Infix, Akademische Verlagsgesellschaft Aka GmbH, Sankt Augustin, Berlin.

J. Alan Robinson (1965). *A Machine-Oriented Logic based on the Resolution Principle.* In: Siekmann & Wrightson (1983), Vol. 1, pp. 397–415.

Peter J. Robinson, John Staples (1993). *Formalizing a Hierarchical Structure of Practical Mathematical Reasoning.* J. Logic Computat. **3**, pp. 47–61, Oxford Univ. Press.

J. Alan Robinson, Andrei Voronkov (eds.) (2001). *Handbook of Automated Reasoning.* Elsevier.

Tobias Schmidt-Samoa (2004). *The New Standard Tactics of the Inductive Theorem Prover* QUODLIBET. SEKI-Report SR–2004–01, FR Informatik, Saarland Univ.. `http://www.ags.uni-sb.de/~veire/SEKI/welcome.html (May 23, 2004).`

Jörg H. Siekmann, Graham Wrightson (eds.) (1983). *Automation of Reasoning.* Springer.

Jörg H. 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.

Raymond M. Smullyan (1968). *First-Order Logic*. Springer.

Lincoln A. Wallen (1990). *Automated Proof Search in Non-Classical Logics.* MIT Press. Cf., however, `http://www.ags.uni-sb.de/~cp/p/wallen/all.txt` for some obsolete aspects of this fascinating book.

Christoph Walther (1992). *Computing Induction Axioms.* 3rd LPAR 1992, LNAI 624, pp. 381–392, Springer.

Christoph Walther (1994). *Mathematical Induction.* In: Gabbay &al. (1993 ff.), Vol. 2, pp. 127–228.

Christoph Walther, Stephan Schweizer (2003). *About* ✓ERIFUN. 19th CADE 2003, LNAI 2741, pp. 322–327, Springer.

Claus-Peter Wirth (1995). *Syntactic Confluence Criteria for Positive/Negative-Conditional Term Rewriting Systems.* SEKI-Report SR–95–09 (SFB), FB Informatik, Univ. Kaiserslautern.

Claus-Peter Wirth (1997). *Positive/Negative-Conditional Equations: A Constructor-Based Framework for Specification and Inductive Theorem Proving.* Ph.D. thesis, Verlag Dr. Kovač, Hamburg.

Claus-Peter Wirth (1999). *Full First-Order Free-Variable Sequents and Tableaus in Implicit Induction.* 8th TABLEAU 1999, LNAI 1617, pp. 293–307, Springer. `http://www.ags.uni-sb.de/~cp/p/tab99/welcome.html (Aug. 05, 2001).`

Claus-Peter Wirth (2002). *A New Indefinite Semantics for Hilbert's epsilon.* 11th TABLEAU 2002, LNAI 2381, pp. 298–314, Springer. `http://www.ags.uni-sb.de/~cp/p/epsi/welcome.html (Feb. 04, 2002).`

Claus-Peter Wirth (2004a). *Descente Infinie + Deduction.* Logic J. of the IGPL **12**, pp. 1–96, Oxford Univ. Press. `http://www.ags.uni-sb.de/~cp/p/d/welcome.html (Sept. 12, 2003).`

Claus-Peter Wirth (2005). *History and Future of Implicit and Inductionless Induction: Beware the old jade and the zombie!*. In: Hutter & Stephan (2005), pp. 192–203. `http://www.ags.uni-sb.de/~cp/p/zombie/welcome.html (Dec. 02, 2002).`

Claus-Peter Wirth (ed.) (2004c). *Proposal: January 2005 – December 2007.* Collaborative Research Center SFB 378 "Resource-Adaptive Cognitive Processes", Universität des Saarlandes.

Claus-Peter Wirth (2004e). *Proof Trees as Formulas.* To appear as SEKI report. `http://www.ags.uni-sb.de/~cp/p/formulas/all.ps.gz (May 23, 2004).`

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 Universität des Saarlandes, winter semester 2003/4. `http://www.ags.uni-sb.de/~cp/teaching/hotp (Sept. 12, 2003).`

Claus-Peter Wirth, Serge Autexier, Christoph Benzmüller, Andreas Meier (2004). *Proposing a Task Interface.* To appear as SEKI report. `http://www.ags.uni-sb.de/~cp/p/tasks/all.ps.gz (May 23, 2004).`

Ludwig Wittgenstein (1939). *Lectures on the Foundations of Mathematics, Cambridge.* Cora Diamond (ed.), from the notes of R. G. Bosanquet, Norman Malcolm, Rush Rhees, and Yorick Smythies, Univ. of Chicago Press, 1989.

Claus Zinn (2003). *A Computational Framework for Understanding Mathematical Discourse.* Logic J. of the IGPL **11**, pp. 457–484, Oxford Univ. Press.

Curriculum Vitae

Name	Dr. Claus-Peter WIRTH
Date of Birth	April 18, 1963
Place of Birth	Diez an der Lahn (Germany)
Sex	Male
Citizenship	German
Marital Status	Unmarried
Address	Brandenburger Str. 42 D–65582 Diez Germany

Telephone	`++49 681  302 4608 office`
	`++49 681  302 5076 FAX office`
	`++49 6432     5961 private`
	`++49 6432 924 9115 FAX private`
	`++49 175  809 0413 mobile`

E-mail	`wirth@logic.at`
WWW	`http://www.ags.uni-sb.de/~cp/welcome.html`

Degrees

1997 Ph.D. (Promotion, Dr. rer. nat.)
(promoters: Prof. Dr. Jürgen Avenhaus, Prof. Dr. Klaus Madlener),
Dept. of Computer Science, Universität Kaiserslautern, Germany

1991 M.Sc. in Computer Science (Diplom-Informatiker) (minor: Economics),
Universität Kaiserslautern, Germany

1987 Bachelor in Computer Science (Vordiplom), Universität Kaiserslautern, Germany

1982 A-levels (Abitur) (majors: mathematics, chemistry, English),
Sophie Hedwig Gymnasium, Diez, Germany

Education

Oct. 1984 – Sept. 1991	Studies in Computer Science, Universität Kaiserslautern
Oct. 1983 – Sept. 1984	Studies in Chemistry, Johannes Gutenberg Universität Mainz
Oct. 1982 – Dec. 1983	Compulsory military service
Aug. 1973 – June 1982	Sophie Hedwig Gymnasium, Diez (high school)

Employment

Since May 2001
Research and Teaching Fellow,
ΩMEGA Research Group of Prof. Dr. Jörg H. Siekmann,
Collaborative Research Center SFB 378
"Resource-Adaptive Cognitive Processes", Project MI-4 "ΩMEGA",
Dept. of Computer Science, Universität des Saarlandes, Germany

April 1998 – June 2000
Research Fellow,
Group "Functional-Logic Development and Implementation Techniques"
of Prof. Dr. Peter Padawitz,
Project "Formal specification for design automation in engineering",
Le 442/10-1,
Dept. of Computer Science 5, Universität Dortmund, Germany

Aug. 1997 – Nov. 1997
Research Fellow,
Group "Artificial Intelligence" of Prof. Dr. Ulrich Furbach,
Dept. of Computer Science, Universität Koblenz, Germany

April 1996 – March 1997
Research and Teaching Fellow,
Group "Foundations of Computer Science" of Prof. Dr. Klaus Madlener,
Dept. of Computer Science, Universität Kaiserslautern, Germany

Nov. 1991 – March 1996
Research Fellow,
Collaborative Research Center SFB 314 "AI — Knowledge-Based Systems",
Project D4 "Computation and Inference in Equational Specifications" of
Prof. Dr. Jürgen Avenhaus and Prof. Dr. Klaus Madlener,
Dept. of Computer Science, Universität Kaiserslautern, Germany

Aug. 1988 – Sept. 1988
Working Student, Functional extension (ε-rules) and
re-implementation (in C) of a parser generator,
IDAS GmbH, Holzheimer Str. 96, D-65549 Limburg, Germany

Specialization

Theoretical Computer Science, Logic, and Artificial Intelligence.
Sub-fields: Inductive Theorem Proving in First-Order and Higher-Order Logics, Rewriting, Proof Planning, Human-Oriented Interactive Automated Theorem Proving, Mathematics on the Computer, Efficient Algorithms, Formal Specification, Formal Methods, Hypertext Web Models

Awards

Ph.D. on *Positive/Negative-Conditional Equations: A Constructor-Based Framework for Specification and Inductive Theorem Proving. Magna cum laude*

Languages

German (native speaker)
English (fluent; with experience in talks, lectures, &c.)

Hobbies

Contemplative: philosophy, history, music, literature, arts
Active: swimming, bicycling, hiking

Saarbrücken, June 30, 2004

References

Books

[1] Claus-Peter Wirth (1997). *Positive/Negative-Conditional Equations: A Constructor-Based Framework for Specification and Inductive Theorem Proving.* Ph.D. thesis, 256 pp., Verlag Dr. Kovač, Hamburg.

Articles in Scientific Journals and Books

[2] Claus-Peter Wirth (2005). *History and Future of Implicit and Inductionless Induction: Beware the old jade and the zombie!*. In: Dieter Hutter, Werner Stephan (eds.). Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday, LNAI 2605, pp. 192–203, Springer.

[3] Claus-Peter Wirth (2004). *Descente Infinie + Deduction.* Logic J. of the IGPL **12**, pp. 1–96, Oxford Univ. Press.

[4] Claus-Peter Wirth (2000). *Full First-Order Sequent and Tableau Calculi With Preservation of Solutions and the Liberalized δ-Rule but Without Skolemization.* In: Ricardo Caferra, Gernot Salzer (eds.). Automated Deduction in Classical and Non-Classical Logics, LNAI 1761, pp. 283–298, Springer.

[5] Claus-Peter Wirth, Bernhard Gramlich (1994). *A Constructor-Based Approach for Positive/Negative-Conditional Equational Specifications.* J. Symbolic Computation **17**, pp. 51–90, Academic Press.

Reviewed Articles in Conference Proceedings

[6] Serge Autexier, Christoph Benzmüller, Dominik Dietrich, Andreas Meier, Claus-Peter Wirth (2005). *A Generic Modular Data Structure for Proof Attempts Alternating on Ideas and Granularity.* Accepted for presentation at MKM 2005.

[7] 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.

[8] Claus-Peter Wirth (2002). *A New Indefinite Semantics for Hilbert's epsilon.* 11th TABLEAU 2002, LNAI 2381, pp. 298–314, Springer.

[9] Jörg H. 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.

[10] Claus-Peter Wirth (1999). *Full First-Order Free Variable Sequents and Tableaus in Implicit Induction.* 8th TABLEAU 1999, LNAI 1617, pp. 293–307, Springer.

[11] Ulrich Kühler, Claus-Peter Wirth (1997). *Conditional Equational Specifications of Data Types with Partial Operations for Inductive Theorem Proving.* 8th RTA 1997, LNCS 1232, pp. 38–52, Springer.

[12] Bernhard Gramlich, Claus-Peter Wirth (1996). *Confluence of Terminating Conditional Term Rewriting Systems Revisited.* 7th RTA 1996, LNCS 1103, pp. 245–259, Springer.

[13] Claus-Peter Wirth, Klaus Becker (1995). *Abstract Notions and Inference Systems for Proofs by Mathematical Induction.* 4th CTRS 1994, LNCS 968, pp. 353–373, Springer.

[14] 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.

[15] Claus-Peter Wirth, Bernhard Gramlich (1993). *A Constructor-Based Approach for Positive/Negative-Conditional Equational Specifications.* 3rd CTRS 1992, LNCS 656, pp. 198–212, Springer.

SEKI-Reports (ISSN 1437–4447) and
SEKI-Working-Papers (ISSN 1860–5931)

[16] Claus-Peter Wirth (2005). $\lim +$, δ^+, *and Non-Permutability of* β*-Steps.* SEKI-Report SR–2005–01, 30 pp..

[17] Ulrich Kühler, Claus-Peter Wirth (1996). *Conditional Equational Specifications of Data Types with Partial Operations for Inductive Theorem Proving.* SEKI-Report SR–96–11 (SFB), 24 pp..

[18] Claus-Peter Wirth (1995). *Syntactic Confluence Criteria for Positive/Negative-Conditional Term Rewriting Systems.* SEKI-Report SR–95–09 (SFB), 188 pp..

[19] Claus-Peter Wirth, Ulrich Kühler (1995). *Inductive Theorem Proving in Theories Specified by Positive/Negative-Conditional Equations.* SEKI-Report SR–95–15 (SFB), 126 pp..

[20] Rüdiger Lunde, Claus-Peter Wirth (1994). *ASF+ — eine ASF-ähnliche Spezifikationssprache.* SEKI-Working-Paper SWP–94–05 (SFB), 64 pp..

[21] Claus-Peter Wirth, Rüdiger Lunde (1994). *Writing Positive/Negative-Conditional Equations Conveniently.* SEKI-Working-Paper SWP–94–04 (SFB), 22 pp..

[22] Claus-Peter Wirth, Bernhard Gramlich, Ulrich Kühler, Horst Prote (1993). *Constructor-Based Inductive Validity in Positive/Negative-Conditional Equational Specifications.* SEKI-Report SR–93–05 (SFB), 58 pp..

[23] Claus-Peter Wirth, Bernhard Gramlich (1992). *A Constructor-Based Approach for Positive/Negative-Conditional Equational Specifications.* SEKI-Report SR–92–10 (SFB), 50 pp..

Reports of the Dept. of Computer Sci., Univ. Dortmund

[24] Claus-Peter Wirth (2000). *Descente Infinie + Deduction.* Report 737, 86 pp., Extd. version, Feb. 1, 2003, 101 pp., `http://www.ags.uni-sb.de/~cp/p/tab99/new.html`.

[25] Volker Mattick, Claus-Peter Wirth (1999). *An Algebraic Dexter-Based Hypertext Reference Model.* Report 1999/719, FB Informatik, Univ. Dortmund, 48 pp..

[26] 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, 40 pp..

Publications at the Dept. of Computer Sci., Saarland Univ.

[27] Claus-Peter Wirth (ed.) (2004). *Proposal: January 2005 – December 2007.* Collaborative Research Center SFB 378 “Resource-Adaptive Cognitive Processes”.

[28] Claus-Peter Wirth (ed.) (2004). *Report: January 2002 – December 2004.* Collaborative Research Center SFB 378 “Resource-Adaptive Cognitive Processes”.

[29] Claus-Peter Wirth, Christoph Benzmüller, Armin Fiedler, Andreas Meier, Serge Autexier, Martin Pollet, Carsten Schürmann (2004). *Human-Oriented Theorem Proving — Foundations and Applications.* Lecture at Universität des Saarlandes, winter semester 2003/4, `http://www.ags.uni-sb.de/~cp/teaching/hotp/`.

Publications at the Dept. of Computer Sci., Univ. Kaiserslautern

[30] Claus-Peter Wirth (1997). *Efficient Algorithms.* Exercises with solutions for the lecture in winter term 1996/7.

[31] Claus-Peter Wirth (1991). *Inductive Theorem Proving in Theories Specified by Positive/Negative-Conditional Equations.* Diplomarbeit (Master’s thesis).

[32] Claus-Peter Wirth (1989). *Darstellung von Datei-Versionen über deren Differenzen.* Projektarbeit.