The \textit{ISAC}-Project

Promising Research for the Next Future

Walther Neuper, neuper@ist.tugraz.at
Institute for Software Technology
University of Technology, Graz, Austria

January 20, 2004

Contents

1 About this paper 2
  1.1 Research and the \textit{ISAC}-project ......................... 2
  1.2 Cooperations and future R&D .............................. 2

2 Ideas about research topics 2
  2.1 Math on the semantic web .................................. 3
  2.2 Re-engineering algebra systems ............................ 5
  2.3 A rigorous framework for applied mathematics .......... 7
  2.4 Man-machine-interaction in learning math ............... 9
  2.5 Usability of a web-based math-learning-tool ............ 10

3 Conclusions 10

4 Persons 11

5 The next sixteen diploma theses 11

Abstract

\textit{ISAC} is a successful development-project geared towards educational use. So far, \textit{ISAC} adopted significant academic know-how, but did not contribute actively to research.

This paper outlines the areas of research the development of \textit{ISAC} is concerned with the intention to attract further interest of researchers in these areas.
1 About this paper

This paper details section 2.4 ‘identification of topics open for research’ in the status report no.1 of the ISAC-project [Neu03a]. It is intended for internal use and to approach external cooperations agreed with the institute.

1.1 Research and the ISAC-project

The report shows, that ISAC — due to the great support at the institute — is successful as a development-project geared towards educational use; in spring 2003 there will be a prototype (in the sequel called 'the prototype') as planned in [Neu03a]. So far the direction of development is due to the professional background of the present project leader; the other side of the medal is, that this professional background is not helpful for ISAC being a successful research project, too.

This lack of success in research is in contrast with the original conception, which has been created under the advice of two outstanding researchers\(^1\), with a promising paper written as a co-author with the present head of the institute [NW02], and with a feedback to a FWF-proposal [Neua], wich confirmed that ‘the grand scheme is certainly a good one’ and ‘the overall ambitions of the project are grand and useful’ [Neub] p.5 and p.7 respectively.

1.2 Cooperations and future R&D

Already ISAC’s conception covered two different research areas (software technology and computer mathematics, due the two advisors), and it was clear from the beginning, that also expertise in learning theory and didactics will be required.

The fact, that the present project leader feels to be no specialist in either field, made him free to approach experts from the different fields involved. Some of these approaches already resulted in fruitful cooperation (Clemens Heuberger, Institute for Mathematics, and Klaus Schmaranz, IICM, both TU Graz), some are still pending due to lack of funding (Tobias Nipkow, Isabelle-team at TU Munich, Andrea Asperti, MoWGLI-project at Bologna, Dietmar Dorninger, math at TU Vienna, Helmut Hengl, ACDCA Austria, etc.).

The aim of this paper is to present possible contributions of ISAC to current research in several areas; such contributions can arise from the rô le of ISAC as a demonstration project, also, ISAC can be a testbed for a variety of research, and one can expect the raise of fruitful new questions out of the work on ISAC.

2 Ideas about research topics

In the following there are some ideas on research, ISAC can contribute to, and can take profit from. The ideas represent the authors limited view, given with the hope to find the interest of experts to go beyond these ideas.

\(^1\)Buno Buchberger and Peter Lucas, both are professors emeriti since two years.
We assemble the ideas around topics, which themselves are interdisciplinary. Sect. 2.1 addresses expertise in software technology with emphasis for technical intricacies of the presentation of math, 2.2 concerns symbolic computation, where unforeseen requirements for software technology may arise, 2.3 seems to address hard-core mathematics (logic) and formal languages, 2.4 advocates for cooperation between learning theory, didactics of mathematics and software technology, and 2.5 should, in addition, involve educational practitioners.

2.1 Math on the semantic web

"The World Wide Web is already the largest resource of mathematical knowledge, and its importance will be exponentiated by emerging display technologies like MathML. However, almost all mathematical documents available on the Web are marked up only for presentation, severely crippling the potentialities for automation, interoperability, sophisticated searching mechanisms, intelligent applications, transformation and processing. The goal of the project\(^2\) is to overcome these limitations, passing from a machine-readable to a machine-understandable representation of the information, and developing the technological infrastructure for its exploitation. MOWGLI builds on previous “standards” for the management and publishing of mathematical documents (MathML, OpenMath, OMDoc), integrating them with different XML technology (XSLT, RDF, ...) \[\ldots\]."

The MOWGLI-project builds on the solid ground already provided by previous European projects (Such as OpenMath and Euler) and several XML dialects for the management of mathematical documents (MathML, OpenMath, OMDoc, ...). All these languages cover different and orthogonal aspects of the information; our aim is not to propose a new standard, but to study and to develop the technological infrastructure required for taking advantage of the potentialities of all of them.\(^3\)

Contacts ZS4C – MoWGLI: ZS4C is designed as a web-based math tool, and thus is exposed to the web’s deficiencies mentioned above, and is committed to the upcoming standards for math on the web, which MoWGLI is expected to establish in the near future.

It was no surprising idea, that ZS4C can take profit from MowGLIs deliverables. This view could be mutually confirmed in three very positive meetings at Bologna in February 2003. [Neu03b] was a suitable preparation such that in the following we just note the outcome of the meetings. One meeting was with Andrea Asperti, Luca Padovani and Claudio Sacerdoti Coen (pts. (1.1), (1.2), (1.4) below), another with Paul Libbrecht and George Goguadze (pts. (1.3), (1.5)), and one with Hanane Naciri (pt. (1.4)).

The situation is even more interesting after the cancellation of the FWF funding in late 2003: we decided to concentrate the restricted resources on the development of the novel functionality of a stepwise interactive algebra system, and let ZS4Cs...\(^2\)

\(^2\)This text is a copy from the MoWGLI-project http://www.mowgli.cs.unibo.it/

\(^3\)This text is a copy from the MoWGLI-project http://www.mowgli.cs.unibo.it/
prototype present math formulae as strings (!) to the users. Thus there will be a highly innovative product with a stone-age front-end.

The résumé agreed on in the meetings was that ISAC will contact MoWGLI again, if there will be some funding.

The findings for possible cooperation, as discussed in the meetings, were the following (while the numbers (X.Y) refer to the sections in [Neu03b]).

**XML exportation (1.1)**, as given by MoWGLIs deliverables $D2.a \cdots d$, transforms theories, proofs and formulae from the theorem prover Coq to an ‘intermediate representation format’. This task is partly accomplished by the prototype for ISACs problems, methods, and examples.

However, the exportation of Isabelle’s theories and formulae could also be of interest for the Isabelle developers; this question has not been discussed at the meeting with Tobias Nipkow and his team at TU Munich in fall 2001. A challenge seems to be to present Isabelle’s type-information in MathML, and to re-import typed formulae from MathML-content immediately to Isabelle-terms (bypassing conversion from XML to strings followed by employment of Isabelle’s parser).

**Stylesheets and DTD’s (1.2)** are provided by MoWGLIs deliverable $D2.e/f$: These tools transform the ‘intermediate representation format’ to (MathML-) presentation format. The tools could easily be adopted for ISACs theories, problems, methods, examples, calculations and formulae; ISAC could prove the genericity of the tools developed.

**Metadata and tools (1.3)**, as given by MoWGLIs deliverables $D3.b$ and $D4.b$, could be very useful for ISAC, too. The former would give guidelines to provide ISACs problems, methods and examples with standardised meta data. It has to be noted, that OMDOC does not yet know ‘problems’; Michael Kohlhase would be the right person to address this question to.

The latter would be helpful for searching ISACs knowledge base, for indexing and interlinking the various items of ISACs knowledge base.

**MathML rendering (1.4)**, as described in MoWGLIs deliverable $4D.a$, is covered by two different tools, one written in C++ (and providing also for input of formulae), and one written in Java.⁴

The latter would be easier to integrate into ISACs (Java-) front-end; however, this tool seems not yet to provide for input of formulae. This could be concern of a cooperation with INRIA.

⁴There are commercial formula-editors – all offered by one (!) vendor [www.dessci.com/]. These editors regard math as a fixed domain, while Isabelle (and thus ISAC) live from the extensibility of the math language; thus these editors are no use for ISAC.
Education (1.5) is addressed by MoWGLIs dissemination and use plan, which announces a validation scenario in the field of education. ISAC could be the medium for parallel field studies.

With this part of MoWGLI primarily DFKI, the German Research Center for Artificial Intelligence, is concerned; this team also has experience with user-models and in metadata for describing math examples.

Alltogether, in this area ISAC seems to be most likely to be involved as a demonstration project; contributions seem possible in (1.3 - OMDOC) and (1.4).

2.2 Re-engineering algebra systems

This topic was the main concern of the proposal [Neua], while the construction of the front-end as addressed by topic 2.1 was a implied as a secondary task in the proposal. Here we repeat just the ideas.

Abstract of the FWF-proposal with headline 'ISAC - a hierarchy of problem types in applied mathematics':

At the intersection of computer mathematics and software technology scientific development goes towards clarification of formal languages and the respective interrelations based on formal logic, and towards mechanic interpretation of increasingly abstract languages.

For instance, the theorem prover Isabelle clearly separates three language layers: (1) the implementation language (SML), (2) the object language (math formulas) and (3) the deductive knowledge (theories, which even define the logic, the formulas are to be interpreted with).

ISAC adds another language layer of application-oriented knowledge which does not yet exist in math systems: types of problems, based on the grounds of 'formal methods' in software technology. These 'problem-types' (e.g. types of equations) ISAC can be 'matched' interactively with a formalized 'problem' (with specified input items, output items, pre- and post-condition). ISAC arranges the problem-types in a hierarchy such that they can be mechanically searched for automated problem refinement.

Construction of the root of this hierarchy concerns re-engineering the basic functions of algebra systems, of simplifiers and equation-solver: hard-coded knowledge is being extracted into separated language layers which are both, human readable and mechanically interpretable. ISAC is just in time to serve as a demonstration project for the European IST-project MoWGLI, which will establish standards for math knowledge representation.

ISACs knowledge interpreter is constructed such that it works stepwise. The atomic steps are matching and rewriting, both of which are intuitively comprehensible by naive users. In order to support this intuition as much as possible, ISAC represents simplification by stepwise rewriting even in domains, which require other methods (e.g. a generalized Euclidean algorithm for canceling multivariate rationals): this is done by a novel 'reverse rewriting', part of establishing a 'rewriting paradigm'.
ISAC is a maximally transparent system, it is transparent w.r.t. the underlying knowledge, and the knowledge-interpreter works transparently as well (with rewriting and matching as the atomic steps, which the user can interactively guide and modify). This is the point, why ISAC is supposed to provide a novel kind of math learning environment: students can go down into the details on their own pace any time, and watch the system work, and inspect the underlying knowledge for the reasons w.r.t. their actual problem solving.

**ISAC’s offers for research** in this area are:

- **ISAC** is implemented in SML [Pau91], probably the most appropriate program language for symbolic computation
- **ISAC** implements a novel structure more flexible than algebra systems:
  - the knowledge, problem-types and methods, is described in Isabelle/HOL *(not in SML)*
  - the problem-handler for automated refinement of problems (e.g. types of equations)
  - the interpreter of the script-language as a simple functional language with powerful tactics (e.g. applying term rewriting systems). However, one could expect, that tactics are required, which are not yet contained in ISAC’s language for methods (see 2.3 below).
  - a ‘single stepping’ rewriter for conditional and ordered rewriting, featuring nested term rewriting systems
- Isabelle’s powerful tools for type-inference, parsing, matching and pretty printing are made handily available
- In the release of 2003 Isabelle already implements knowledge up to limits, continuity and differentiation for complex functions etc.\(^5\)
- Some experience on how to implement math knowledge within this novel framework (hierarchy of elementary equations, ’reverse rewriting’ in the simplification of multivariate rationals)

**The subtasks** seem to be straight forward, most of them being appropriate for diploma theses of math students. The proposal gives the following list.

**TheC1:** factorization in multivariate polynomial rings, plus application for equation solving.

**TheC2:** symbolic computation on radicals is indispensable for most applications. The implementation includes an efficient canonical simplifier, and a novel application of ‘reverse rewriting’.

\(^5\)See Isabelle’s theories at http://isabelle.in.tum.de/library/index.html
TheC3: term-orders for AC-rewriting have been done already for polynomials and rationals over integers. The implementation of floating-point numbers and of complex numbers will re-call this task.

TheC4: algebraic and transcendental equations. This topic requires the design of the hierarchy (integrated into the existing hierarchy of polynomial and rational equations), i.e. appropriate normal forms, patterns and predicates for the preconditions; it includes the implementation of the respective problem types together with methods solving them.

TheC5: floating point numbers will be introduced to Isabelle as a new datatype; this topic includes the numeric computation features, and addresses the approximative nature of floating point numbers (when they represent reals) following hints given by [Har97].

TheC6: symbolic computation on complex numbers: Complex numbers are already implemented in Isabelle 2003; because ISAC employs its own ’single stepping’ rewrite engine, these numbers need to be integrated into the respective simplifiers.

TheC7: integration can build upon Isabelle-2003’s implementation of the integral on real and on complex numbers in nonstandard analysis. An idea is, not to use latest advances of symbolic computation (Rothstein-Trager-Lazard-Rioboo algorithm [Bro97]), but to decompose Risch’s algorithm into subproblems and thus resembling methods as taught in elementary math courses.

TheC8: problem-types for calculus provide for a knowledge widely used application domain in engineering. This knowledge will comprise the stuff taught at (technical) high-schools, and if possible, the application selected in Appl.

This list is not at all exhaustive; the great number of elements indicates, that more than one institute should be involved. In particular, the implementation of specific topics (differential equations etc.) would require specific expertise.

One obstacle for students choice of such a diploma thesis is, that usually some code must be written in SML.

2.3 A rigorous framework for applied mathematics

Surprisingly, in academic education, too, math students learn the formal language of mathematics like their mother language, like a natural language, just by imitating and exercising. There are few exceptions to this, i.e. courses on formal logics in the entry phase of studies, which usually can be found in math-biased courses in

\[ \text{ISAC}. \]

\[ \text{Our dream is, that mathematicians will publish new concepts by implementation in Isabelle, and accordingly will publish new algorithms by implementation in ISAC.} \]
computer science. This is according to the fact, that mathematicians (particularly in academia) don’t take computers serious in the sense, that they use them for conjecturing, graphing etc, but if they seriously start their professional work and write down a proof, they take paper and pencil (and a computer only for \LaTeXing).

This is in spite of the two facts, that the foundations of mathematics in formal logic have been clarified thoroughly during the last century and that computer science has established a successful handling of formal languages. Since Bourbaki has made the presentation of math consistent, math is it’s own specification in the line of ‘formal methods’ [FJL97], actually a specification unsurpassed in consistency and completeness. Thus deriving math programs should be a straight forward and almost mechanical process\(^7\), leading to programs proven correct w.r.t. the specification.

So far there is some successful research in this direction: computer theorem provers like PVS, HOL, Coq, Isabelle, implement their respective mathematical logic, formally define math objects, and mechanically prove theorems on these objects — und thus create fast growing bodies of math knoweldge, which is proven correct mechanically. The Foc-project\(^8\) [Fec01] aims at building an environment to develop certified computer algebra libraries. In the Foc language, any implementation must come with a proof of its correctness. And there are already considerations about integrated systems [Buc96, FvM03], which ’will provide a uniform (logic and software) framework in which a working mathematician, without leaving the system, can get computer-support while looping through all phases of the mathematical problem solving cycle (specifying a problem, exploring a given problem, proving or disproving conjectures, programming, and writing up one’s findings)\(^9\).

\(\text{ZSAC}\) is interested in applying knowledge, which has been proven in Isabelle [Pau94] in order to solve problems in engineering and science — but there is still no 'logic for solving'! Such a logic will be different from the logics used in theorem proving: the latter, given a theorem, tries to construct a proof-tree with certain properties according to a specific logic, and then considers the respective theorem ’true’. In contrary in applied math, given some input data (meeting some pre-condition) to a problem, one tries to solve the problem by constructing some output data, which are related to the input by a so-called post-condition: what will be the certain properties of a calc-tree?\(^??\).

The clarification of such a 'logic for solving' and an appropriate implementation of this logic in a 'transparent system' as described in sect.2.2 will give completely new approaches to the foundations of math as cited above. Actually, even high-school students are interested in questions of decidability, soundness, completeness and incompleteness (e.g. Gödels theorems), if they interactively experience ‘mechanical thinking’ in the formal framework of a clearly designed mathematics engine.

---

\(^7\)Of course, this conclusion is too simplistic. For instance, \(\exists\) in a specification cannot lead to an obvious solution in a constructive program, \(\forall\) may be even more harmful in program synthesis.

\(^8\)http://www-spi.lip6.fr/foc/

\(^9\)cited from http://www.theorema.org/
**ISACs offers for research** in this area are:

- **ISAC** is implemented in SML which has a formally defined semantics [MTHM97]
- **ISAC** has a preliminary language for methods, which is a (marginal) extension of Isabelle/HOL. Thus an ISAC-method is an Isabelle-term, ready to be fetched to Isabelle’s proof environment.
- The language preliminarily implements the idea to separate the aspects of denotational (? algebraic) semantics and of operational semantics by hiding the construction of the ’calc-tree’ from the programmer of methods.
- The methods comprise novel kinds of high-level ’tactics’: the appropriateness and completeness of the set of tactics is an urgent research issue.
- **ISAC** has a very simple (and thus easily extensible) **interpreter** for the method’s language, which not only (1) determines the next tactic to promote a calculation, but also (2) tries to locate a user-input tactic in the method, and (3) tries to deduce a user-input formula from the current ’calc-state’ by use of ”appropriate’ tactics from the method in use.\(^{10}\)
- The tactics for rewriting (in principle) only use theorems proven in Isabelle/HOL.

**Ideas on a procedere** for research within this topic are the following:

- Iterate the following points until the set of ’tactics’ and other program operators seems to be ’complete’:
  1. select a certain set of problems from applied math (one example may be the one given in [Neu03a] on p.8/9)
  2. identify a set of ’tactics’ and other program operators necessary for describing the method which calculates the result of the problem
  3. prove the methods correct w.r.t. the post-conditions of the respective problems
- establish an environment specialized for such proofs within Isabelle
- relate the above proof procedures to the respective math domains and abstract them towards a ”calculus of solving” (this work essentially would amount to push the tactics and the program operators into the alphabet of the respective proof-language)
- investigate the above interpreters’ features (2) and (3) w.r.t. ’applicability’ of these events caused by the user to the analogon of a proof-state (‘calc-state’); this will lead to work on an operational semantics of these methods.

### 2.4 Man-machine-interaction in learning math

**TODO**

\(^{10}\)Feature (1) of the interpreter is the usual one, whereas the features (2) und (3) seem to be novel generalizations of debuggers’ features; both of the latter are essential for ISACs novel kind of support for user-interaction.
2.5 Usability of a web-based math-learning-tool

TODO

3 Conclusions

This paper pointed out that the development of ISAC is concerned with front-of-the-wave research in several vivid academic disciplines.

With respect to this point we express the hope to attract further interest of researchers in these disciplines and highly welcome decisive strengthening and amplification of the ISAC-project!

We have the prospect of fruitful research results and of a useful educational tool for applied mathematics.
4 Persons

mentioned in this report, in alphabetical order:

- Andrea Asperti http://www.cs.unibo.it/~asperti/
- Bruno Buchberger http://www.risc.uni-linz.ac.at/people/buchberg/
- Dietmar Dorninger http://www.algebra.tuwien.ac.at/dorninger/
- George Goguadze http://www.activemath.org/~george/
- Clemens Heuberger http://finanz.math.tu-graz.ac.at/~cheub/
- Helmut Heugl http://www.acdca.ac.at/
- Michael Kohlhase http://www.ags.uni-sb.de/~kohlhase/
- Paul Libbrecht http://www.activemath.org/~paul/
- Peter Lucas, Professor emeritus at IST, TU Graz.
- Walther Neuper http://www.ist.tu-graz.ac.at/neuper/
- Tobias Nipkow http://www4.informatik.tu-muenchen.de/~nipkow/
- Luca Padovani http://www.cs.unibo.it/~lpadovan
- Claudio Sacerdoti Coen http://www.cs.unibo.it/~sacerdot
- Klaus Schmaranz http://www.iicm.edu/iicm/staff/university506a/kschmar
- Franz Wotawa http://www.ist.tugraz.at/staff/wotawa/

5 The next sixteen diploma theses

The following list presents topics for diploma theses which are required most urgently for the forthcoming development of ISAC. Not all of them are covered by the topics of research as sketched in sect.2.

Topics on math in the semantic web related to sect.2.1; these involve co-operation with MoWGLI and are addressed to students of software technology.

- XSLT for $\mathcal{S}4\mathcal{C}$ problems, methods, examples, and calculations, see p.4 (1.2)
- transformation MathML-content to MathML-presentation, see p.4 (1.2)
- extend the Java formula-renderer FIGUE with formula-input, see p.4 (1.4)
- tool for semi-automatical interlinking of items in the database, and unidirectional from calculations, see p.4 (1.3).
Topics on the distributed multi-user architecture involving cooperation with Dinopolis; addressed to students of software technology. The architectural design of ISAC has been completed already for the prototype. This has been done looking forward to the availability of the Dinopolis-components for Java expected by the end of the year.

- port ISAC’s object-handling to Dinopolis
- implement the multi-course features (specific explanations, access-rights etc. for specific courses)
- implement authoring tools for lecturers of courses
- implement admin-tools for lecturers of courses

Topics for SML-Gurus involving work on ISAC’s knowledge-interpreter; addressed to students of software technology.

- re-engineering and extending the method-interpreter, see p.9
- multiple environments for applied mathematics
- profiling and optimization of the mathematics engine
- specific syntax-tools for solving transcendental equations.

Topics about extensions of ISAC’s math-knowledge concerning Isabele’s HOL-language; addressed to students of mathematics.

- TheC2: symbolic computation on radicals, see p.6
- TheC5: floating point numbers, see p.7
- TheC6: symbolic computation on complex numbers, see p.7
- TheC7: integration, see p.7

References


