An Introduction to the *SysteMaThEx* Project

IeAT Report, February 2005

Adrian Craciun
Institute e–Austria, Timisoara, Romania
Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria
acraciun@{ieat.ro,risc.uni–linz.ac.at}

Recently, the proposal for project *SysteMaThEx* – "Systematic Mathematical Theory Exploration with the Theorema System: Case Studies" has been evaluated favorably by the European Commission and is now (at the moment of writing the present report) in the negotiations phase. This internal report of the Institute e–Austria (IeAT) aims to provide a scientific and organizatorial view of the project and sketch a research plan for carrying out the work in the project. Thus, this report should be a useful reference for potential participants in the project, members of IeAT and whoever else is interested. The report gives an short overview of the broader context of Mathematical Knowledge Management, places the *SysteMaThEx* project in this context and links it with similar relevant work. The scientific objectives of the project – case studies in the theory of lists and in polynomial ideal theory – are then motivated and described in some detail, together with the tools and methods that will be used in the project – the *Theorema* system, the theory exploration model introduced recently by Bruno Buchberger. Another part of the report addresses concrete organizatorial aspects of the project.

1. Introduction

■ 1.1. Disclaimer

This is an internal report of the Institute e−Austria, not intended for publishing. Parts of this report can later become part of some articles. This is the first version, with possible further versions expected.

■ 1.2. The Intended Readership

Of course, anybody can read the present report. However, by aiming at an overview of *SysteMaThEx* project before its start, this should be of special interest for potential participants in the project. It should (and hopefully does) identify the project in the scientific context, give a feeling about the techniques that will be employed and developed in the project and provide a coarse plan for the project itself and its possible extensions.

■ 1.3. *SysteMaThEx* at a Glance

The *SysteMaThEx* project (proposal) aims at providing major case studies of systematic theory exploration using a model for systematic theory exploration recently proposed by Bruno Buchberger [Buchberger 2004b]. Two major case studies were proposed for this project:

− **exploration of the theory of tuples** and similar theories (strings, sets, bags, trees) to the extent described in literature, for example in [Manna Waldinger 1985], [Gries Schneider 1993],

− **exploration of the theory of Gröbner Bases** to the extent described in literature, for example in [Winkler 1996], [Buchberger 1998].

The case studies will be carried out with the help of the *Theorema* system, and will have as side effects the refinement of the *Theorema* system to support the case studies undertaken by the project.

The proposal for the project is intended for the Sixth EU Framework Programme for Research and Technological Development (FP) − Marie Curie Actions, European Reintegration Grants. The participants in this project are:

− a **researcher** that has spent at least 24 months in a European Project (Adrian Craciun, the author of this report),

− a **host institution**, in the researcher's home country (the Institute e−Austria,Timisoara).

The purpose of this specific programme is to ensure an environment that allows the reintegration of the researcher into his (in this case) home country and which also provides further prospects for the development of a future career in research.

■ 1.4. The Origins of the *SysteMaThEx* Project − *Theorema*

The origins of the *SysteMaThEx* project lay in the work carried out in the *Theorema* group at the Research Institute for Symbolic Computation under the guidance of Bruno Buchberger.

The *Theorema* system aims to provide computer support to "working mathematicians" (i.e. mathematicians who teach and/or do research in mathematics), in all the phases of the mathematical activities (see [Theorema-Web] for an overall description and documentation, papers − some of which we will refer to throughout this report):

− *proving* propositions about *concepts* (introduced by definitions);
− *computing* using algorithms;
− *solving* equations, inequations, systems thereof;
− *parsing, understanding* and further *presenting* interactive mathematical texts.

Bruno Buchberger has long advocated the "theory exploration paradigm" as opposed to "isolated theorem proving", for example see [Buchberger 2000], [Buchberger 2004b], and the *Theorema* system is driven

towards supporting the exploration paradigm. In fact Bruno Buchberger had a pivotal role in establishing the new field of Mathematical Knowledge Management (MKM), which tries to address the needs for automated support in the various aspects of doing mathematics – see [MKM 2001], [MKM 2003] (MKM will be introduced in a bit more detail in a following section).

One of the most important reasons for setting up the *Theorema* system, according to Bruno Buchberger, was his (a real working mathematician) need for an automated support tool for the exploration of the theory of Gröbner bases.

The *SysteMaThEx* project is motivated by all of the above, and it aims to make a contribution to the field of MKM by developing knowledge bases through systematic exploration in the two important case studies considered (theory of tuples and theory of Gröbner bases).

■ 1.5. About the Author of This Report

Adrian Craciun is currently (at the moment of writing this report) working on his PhD in the frame of the RISC PhD program, under the supervision of Bruno Buchberger. From 2000 until 2004 he has been working in the *Theorema* group, supported by the FP5 European Project Calculemus. From October 2004, he has moved to the IeAT institute, where he continues the work on his thesis (this work will be supported by the *SysteMaThEx* project), which he is expected to finish in the summer/autumn of 2005.

The *SysteMaThEx* project is strongly connected to the topic of the author's thesis – program synthesis in the context of systematic theory exploration; it will support the completion of this work and allow directions to naturally extend it.

■ 1.6. IeAT

The Institute e–Austria (IeAT) is a research institute in the area of computer science, that was explicitly founded for giving graduates from the universities in Timisoara new opportunities for staying in the country and pursuing careers in R&D. IeAT joins researchers from three academic institutions: two computer science departments from West University and "Politehnica'' University, in Timisoara and RISC. Fields of research expertise include software engineering, formal verification, automated theorem proving, artificial intelligence, high performance computing and computational mathematics.

While already having ties to the *Theorema* project, carrying out the *SysteMaThEx* project at the IeAT institute will allow the establishment of a research group in the fields of automated theorem proving, computer algebra, MKM, and will support young researchers who study in the Universities in Timisoara if they were to choose a career in the mentioned fields.

■ 1.7. The Contents of This Report

In the following, the next section is dedicated to describing the context in which the *SysteMaThEx* project is placed – that of Mathematical Knowledge Management – MKM. First a broad overview of this new research field, then a closer look to what represent the core of MKM – systematic generation of knowledge bases of mathematics, in particular the problem of invention in mathematics, and some approaches that tackled this aspect. The *Theorema* system as the main tool to support the implementation of the project will be introduced and described in some detail. The last part of this section will be dedicated to describing the exploration model introduced by Bruno Buchberger and which will be in the focus of the *SysteMaThEx* project.

The next section — The *SysteMaThEx* Project – Scientific Part — will identify the scientific objectives of the project. The case studies considered for this project, are briefly motivated, illustrated with a few examples. Some research directions to be followed during the project are indicated, including one of the main focus points of this project – a contribution to the problem of invention – synthesis in the context of systematic theory exploration. We will discuss some of the possible approaches here. The method of research is presented in some detail.

Then follows a section dedicated to the organizatorial aspects of the project. We discuss here the steps we will take to form the group – hiring people, seminars, etc., as well as other informations of organizatorial nature.

Summarizing the information from the previous two sections, we give a sketch of a work plan for the 2 year span of the *SysteMaThEx* project.

The last section is dedicated to conclusions and a summarizing discussion.

2. Where Do We Live and What Do We Do There?

■ 2.1. Where Do We Live? – The Big Picture: Mathematical Knowledge Management

The "world" we are "living" in is the world of mathematics. "We", in the large sense of the word means "mathematicians", and the world of mathematics (with the knowledge attached to it) spans from the caves and plains of the dawn of humanity to the many universities, laboratories, etc. where mathematics (no matter by what name it goes) is developed today. While more details about the nature of mathematics, definitions, history, etc. is not in the scope of this report (refer, for this, to encyclopedias, histories of mathematics, philosophy texts, etc.), one fact remains: throughout its history, vast volumes of mathematical knowledge was/is/will be produced, even more so with the technological advances of the last few (hundreds, tens of) years.

The support of the mathematical knowledge is diverse (stone–paper–electronic), so is the style (informal, formal) and notation, some of the mathematical texts may contain logical errors. Of course, in an ideal situation, the mathematician has access to the mathematical knowledge, can make sure that this knowledge is correct, and can contribute to it by further developing it. The development of the field and the development of technology of the last couple of centuries meant big steps towards achieving this ideal situation, in particular

– the development of automated reasoning systems – automated theorem checkers/provers – allows checking the correctness of mathematical proofs and the production of correct proofs respectively,

– the development of computer algebra systems allows difficult computations to be performed in a very short time,

– the development of editors, editor languages, etc, allows the easy production of high print quality mathematical texts,

(i.e. the development of the computer technology plays an essential part in this effort to reach the ideal situation described above).

As a remark, the sense of "we" becomes a bit more restricted, "computer mathematicians" (i.e. those who do mathematics with the support of computers, on some level).

Mathematical Knowledge Management is a new research field, established with the occasion of the First International Workshop on Mathematical Knowledge Management – [MKM 2001], which tries to bring together and focus all the efforts to reach the ideal situation described above, i.e. to provide tools that allow efficient management of mathematical knowledge.

Bruno Buchberger proposed the following break–down of the field of MKM in an early version of [Piroi Buchberger 2004], also in [Buchberger Nakagawa 2004]:

(a) the *organizatorial* aspect of MKM, which is concerned with:

– "digitization of mathematics": scanning of the large amounts of mathematical knowledge available in print into electronic formats (ps, pdf), the transformation of these into representation formats that allow editing and publishing (e.g. LaTeX);

– parsing of LaTeX documents, detecting the logical structure of documents, [Nakagawa et al 2004];

– providing standards for representation of mathematical knowledge (with its logical structure), such as MathML [MathML], OpenMath [OpenMath];

– building up tools for MKM, from tools for management of knowledge bases (such as those described in [Piroi Buchberger 2004]), tools for general work in mathematics (such as frontends to theorem provers and computer algebra systems – some of them already provided by commercial systems like Mathematica or Maple, and some of them developed from scratch or based on available editing/visualization tools –

see Proof General for the case of the Isabelle theorem prover [Isabelle], or Omega [Omega]), and tools for the exploration and development of (new) knowledge bases (**and side effects of the *SysteMaThEx* project can be the development of such tools**);

– mathematical knowledge bases, in (some) electronic format, including the full logical structure of various mathematical theories, and there are already some fairly large portions of mathematics available in the various formats corresponding to various systems, such as the Journal of Formalized Mathematics, developed with the Mizar proof checker [JFM], MBase [MBase], the Formal Digital Library project [FDL], the NIST Digital Library of Mathematical Functions [Nist], Hypertextual Electronic Library of Mathematics [Helm], the libraries of the theorem provers Isabelle [Isabelle], IMPS [Imps], Coq [Coq];

(b) the *logical* aspect of MKM, which is at the core of the field, and which is concerned with:

– formalization of mathematics, i.e. transferring mathematical knowledge into some electronic, formally checked format, which was started by the Automath project ([Automath]), continued with Mizar ([Mizar]), and which at some level or the other is found in the context of every theorem prover, proof assistant;

– methods to ensure communication between various formalisms used in various systems (e.g. logical frameworks);

– (computer supported) exploration of mathematical theories, invention (**which is the purpose of SyteMaThEx**).

Note that one of the main goals of MKM is the production of knowledge bases, reflected in both the branches of MKM mentioned above. In the case of the organizatorial branch, the focus is on technical details (formats, systems) whereas in the second case the focus is on how these knowledge bases are built. Next, we will discuss the exploration and invention aspects of MKM in more detail.

■ 2.2. Where Do We Do There? – A Closer Look: Systematic Exploration – Invention

■ A Model for Systematic Theory Exploration

Bruno Buchberger has introduced, in [Buchberger 2004a], (and later streamlined in [Buchberger 2004b]) a model for systematic theory exploration based on schemes, as concentrated mathematical knowledge. The schemes form the **ideas thread** in the four thread exploration model (as introduced in [Buchberger 2004a]):

Ideas Thread	Theory Thread	Reasoning Objects Thread	Reasoners Thread
Concentrated math knowledge – definition schemes; – theorem schemes; – problem schemes; – algorithm schemes;	– axioms; – definitions; – theorems; – algorithms;	Reasoning objects, combining – computation steps; – proof steps ; – solve steps.	Reasoners, combining – proving; – solving; – simplifying. Methods for exploration – cascade; – lazy thinking.
[1]	[2]	[3]	[4]

The language considered for this model of exploration is that of higher order predicate logic (plus sequence variables and the quotation symbol "▼", that allows the distinction between object-level and meta-level formulae, see [Buchberger 2004a] for details).

This may yet not be familiar to the reader. However, a closer look reveals that the four threads could be familiar to the reader familiar with the following contexts:

• [2]+[3] ⟶ Mathematical textbooks – that usually contain some development of a theory (with axioms, theorems, definitions), including proofs (and possibly the descriptions of the proof method and the proof of correctness);

• [2]+[3]+[4] ⟶ CA (Computer Algebra) and ATP (Automated Theorem Proving) systems;

• [1]+[2]+[3]+[4] ⟶ Scheme-Based Algorithm Synthesis, with several approaches undertaken, such as that of Smith [Smith 1999], and lazy thinking introduced by Bruno Buchberger in the context of systematic theory exploration in [Buchberger 2003] (this will be discussed in more detail in a subsequent section).

The exploration of a theory proceeds in exploration stages, cycles ("the creativity spiral in mathematics", [Buchberger 1993]). Consider an **exploration situation** (reflecting the current stage in the exploration) $\mathfrak{E}$, defined (in accordance with the four thread model) by:

$\mathfrak{E}$:

[1] $\langle \mathfrak{S}_d , \mathfrak{S}_t , \mathfrak{S}_p , \mathfrak{S}_a \rangle$ – schemes available (higher order formulae);
[2] $\langle \mathfrak{A}, \mathfrak{D}, \mathfrak{T}, \mathfrak{Al} \rangle$ – axioms, definitions, theorems;
[3] $\langle \mathfrak{O} \rangle$ – reasoning objects;
[4] $\langle \mathfrak{P}, \mathfrak{C}, \mathfrak{Lt} \rangle$ – reasoners, cascade, lazy thinking machineries.

An exploration step takes an exploration situation into another, $\mathfrak{E}_1 \longrightarrow \mathfrak{E}_2$ according to the following strategy (with focus on [2]):

[a] **Start.** Exploration situation:
[1] – libraries of definition, theorem, problem, algorithm schemes;
[2] – current theory (+ new axioms);
[3] – proofs, computation traces,
[4] – available machineries.

Repeat:

[b] add (invent) new **concepts**:
– by using definition schemes from [1];
– by using algorithm schemes (to solve some problem) and lazy thinking machinery from [1] and [4];

[c] add (invent) new **conjectures**:
– by using theorem schemes from [1];
– by syntactic analysis of possible interactions between a new concept and old concepts in the theory – using only [2];
– as a side effect of applying cascade or lazy thinking machinery from [4].

until [saturated, <bored, done, etc…>] **.**

In exploring new mathematical theories, the focus is, in general set to developing the theory thread. However, of great interest is a shifting on the focus to the development of the other threads:

- focus on [1]: development and classification of knowledge schemes;
- focus on [3]: proof simplification, focus windows;
- focus on [4]: lifting knowledge to inference and proving correctness of provers.

■ Other Approaches to Concept/Conjecture Invention

There are several efforts by various researchers and research groups towards systems of (automated) discovery of concepts and conjectures in mathematical theories, see [Colton et al 2000] for a survey. Most of the approaches surveyed are limited to certain domains (such as finite sets, graphs, plane geometry, number theory), and have been more or less successful in their domain of exploration.

One of the best automatic mathematics discovery systems is Colton's HR, see [Colton 2002]:
– it discovered 14 sequences missing from the encyclopedia of integer sequences,
– it discovered interesting (for mathematicians) theorems about sequences,
– it was the only non–human to add to the TPTP library ([TPTP]) 184 conjectures.

HR functions according to the following scheme:

$$\texttt{Axiom sets} \xrightarrow{\texttt{MACE model generator}} \texttt{Objects of Interest} \xrightarrow{\texttt{HR discovery}} \texttt{Concepts} + \texttt{Conjectures} \xrightarrow{\texttt{Otter}} \texttt{Proofs.}$$

Concepts are sets of predicates true for all objects, and examples that satisfy the predicates. New concepts and conjectures are formed by using various production rules (such as generalizations, specializations, etc.), and several measures of interestingness are defined for them.

However, the HR system is not yet deemed ready to use by mathematicians, for example in [Bundy et al 2003], S. Huczynska and R. McCasland (mathematicians): "while HR has some unique qualities which may have potential , the system as it stands is not ready for use in research mathematics" (although according to Huczynska it is good enough for setting tutorial questions in number theory).

Many of the ideas developed in HR, like interestingness measures, production rules, can be relevant to our exploration (invention/discovery) model.

- **Invention of Algorithms – Synthesis**

At the beginning of this section, we described the general strategy to carry out one exploration step. One way to add new concepts to the theory (thread [2]) is by synthesizing algorithms that meet problem specifications (logic formulae of the form $\forall_X Q[X, A[X]]$, where the predicate Q[X, Y] describes a relation between the input X and the output Y, and A is the algorithm to be synthesized).

The **lazy thinking synthesis method** (introduced by Bruno Buchberger, see [Buchberger 2003]) works in the following way:
– Start with an exploration situation containing the problem specification P, the knowledge K (the content of the theory thread, or parts of it),
– The algorithm A that satisfies the problem P is determined in the following way:
– select (if applicable) one after the other algorithm schemes from the library of algorithm schemes available in the exploration situation,
– try to prove the correctness theorem (given by the problem specification P) using available knowledge and the algorithm scheme,
– the proof is likely to fail, because the algorithm scheme introduces unknown symbols in the theory,
– when the proof fails, analyze the proof attempt and generate conjectures (specifications on the unknown symbols) that allow the proof to go through,
– when the proof succeeds, the following theorem is true: the algorithm A introduced by the algorithm scheme satisfies the specification P, provided that we can find algorithms that satisfy the specifications for the unknown symbols (introduced by the algorithm scheme), i.e. the conjectures made during the proof attempt,
– two choices are available now: either such algorithms exist already in the knowledge base (and we have a retrieval problem), or we apply the method of lazy thinking to synthesize the unknown algorithms specified by the conjectures.

An implementation of lazy thinking was carried out in *Theorema* by the author, and used in case studies in the theory of tuples, see [Buchberger Craciun 2004] (which also contains a comparison to related methods for algorithm synthesis). Some more considerations on lazy thinking, case studies, possible improvements and relevance to the *SysteMaThEx* project will be made in the next section.

■ 2.3. What Are the Tools? The *Theorema* System

Theorema is the tool for doing theory exploration (in particular invention) as described in [Buchberger 2004b], and outlined above.

- **A Short Note on *Theorema***

To achieve this, *Theorema* integrates techniques of *automated theorem proving* and *computer algebra*, using *one language frame* – a variant of *higher order logic* (the *Theorema* language).

The *proving* aspect of *Theorema* is handled by *basic provers*:
– general purpose (containing general proof techniques, which do not depend on particular domains), such as the predicate logic prover, the propositional prover,
– special provers (containing proof techniques that are only valid in particular domains), such as the set theory prover, the Gröbner bases prover, etc.
The basic provers are combined in *user provers* (visible to the user), which implement *proof strategies*.

The system is implemented on top of the *Mathematica* computer algebra system, this choice being motivated among others by the following:

– the *Mathematica* front–end (notebook files) allow "textbook–like" syntax, e.g.:

```
Definition["limit",
    ∀f,a limit[f, a] ⟺ ∀ϵ>0 ∃N ∀n≥N |f[n] - a| < ϵ  "l1"]

Definition["product of sequences/functions", any[f, g, x],
    (f * g)[x] = f[x] * g[x]  "f*g"]

Proposition["limit of product", any[f, a, g, b],
    (limit[f, a] ∧ limit[g, b])  "l2*"
        ⟹ limit[f * g, a * b]]

Lemma["distance of product", any[x, y, z, t, δ, ϵ],
    (|(y * t) - (x * z)| < (δ * (ϵ + |z| + 1) + |x| * ϵ)) ⟸  "dist*"
        (|y - x| < δ ∧ |t - z| < ϵ)]

Algorithm["sorted", any[is-tuple[X]],
 sorted[X] =
  { special[X]                                                ⇐ is-trivial-tuple
  { merged[sorted[left-split[X]], sorted[right-split[X]]]     ⇐ otherwise
   ]
```

(note that the above are random examples, provided only to illustrate the indicated feature of the system).

– the whole range of *Mathematica*'s algorithm libraries is available to *Theorema* (on demand – in fact *Theorema* does not implicitly use these algorithms),

– the programming language of *Mathematica* is a functional style language, with powerful matching mechanisms, which allows elegant and rapid implementation of *Theorema* components (provers, simplifiers, computers).

Naturally, this choice has certain disadvantages, mostly the fact that the use of the *Mathematica* programming language makes the system slower than it would be if it were implemented in lower–level programming languages (such as C, Java).

Further information on the Theorema system can be found in [Buchberger et al 2000], [Buchberger et al 2004], [Buchberger et al 2005].

■ MKM Tools In *Theorema*

Theorema was designed as a system to support mathematical activities. Not surprisingly, features that support (the automation of) this activity are present in the current version of the system. Such mechanisms include:

– the Cascade prover that allows discovery of theorems in a top-down exploration cycle, as described in [Buchberger 2000],

– the Functor syntax, that allows building new domains, see [Tomuta 1998],

– the Focus Windows technique for exploration of reasoning objects, as described in [Piroi Buchberger 2002],

– the Label Management tools for organization of knowledge bases, see [Piroi Buchberger 2004],

– the Cascade Lazy Thinking prover, implemented by the author, and used to synthesize algorithms, see [Buchberger Craciun 2004].

More MKM tools to assist work in mathematics and the development of the system are needed, however. Some of these are:

– tools to manage libraries of knowledge schemes,

– query tools for knowledge bases, and a visual interface to handle knowledge queries,

– documentation tools,

– visual exploration tools (in the vein of Focus Windows).

3. The *SysteMaThEx* Project – Scientific Part

■ 3. 1. The Scientific Objectives of the *SysteMaThEx* Project

1. Case Studies (Main Goal):
    1.1. Tuples (and related theories)
        a. Bottom–Up Exploration
        b. Algorithm Synthesis
            – exploration of various algorithm types,
        c. Preprocessing of algorithm schemes
    1.2. Gröbner Bases
        a. Bottom–Up Exploration of Polynomial Rings, Reduction Relations
        b. Algorithm Synthesis
            – synthesis of a Gröbner Bases algorithm,
            – synthesis of other algorithms in the theory
        c. Generalization:
            – theory exploration in reduction rings, general reduction domains, axiomatization of general reduction domains,
            – synthesis in such domains (CPC algorithm scheme, others).

2. Improvements / Tools for MKM in *Theorema* (Secondary Goal – expected by–product of the case studies):
    2.1. New Provers (for the domains explored in the case studies).
    2.2. Libraries of (Definition, Theorem, Algorithm) Schemes.
    2.3. Infrastructure for Theory Exploration.

■ 3.2. The Case Studies

We will here briefly introduce the two major case studies that will be undertaken in this project, motivate them and indicate possible research directions that can be undertaken in each case.

■ Exploration in the Theory of Tuples

The theory of tuples (lists) is a well-studied theory in the frame of computer science, as represents one of its fundamental theories. Most of the basic algorithms use objects from the domain of lists (in some representation). This theory is very well suited for exploration, because it is "as simple as possible, as complicated as necessary": results should appear fast enough and it should be easy to compare them against the abundant literature available.

The theory of tuples has already been explored by the author, for testing the sequence prover developed in the frame of the *Theorema* system, see [Craciun 2002]. The theory was developed in a systematic way, "by hand", in a bottom-up manner. However, the goal of the *SysteMaThEx* project is to achieve this in a computer supported way.

As a part of his PhD work in the frame of the *Theorema* group, the author has implemented Buchberger's lazy thinking synthesis paradigm and employed it in a case study of synthesis in the theory of tuples, that of the synthesis of sorting algorithms using the divide-and-conquer algorithm type, see [Buchberger Craciun 2004]. In the **Appendix** of this report, this example is presented in big detail.

So far, only the divide-and-conquer algorithm scheme (and related schemes based on structural induction) have been investigated. Efficiency aspects have not been investigated in big detail.

- **Exploration in the Theory of Gröbner Bases**

As already mentioned in the introductory section, one of the main motivations and goals of the *Theorema* project was the need for support for exploration in the theory of Gröbner bases, for an examples of exploration related to this are [Buchberger 2003a] (focused on computation of Gröbner bases), and [Buchberger 2004c] (where Bruno Buchberger shows that the lazy thinking method for algorithm synthesis is in principle powerful enough to synthesize an algorithm deemed beyond the reach of synthesis systems).

The *SysteMaThEx* project will first undertake the task of carrying out the synthesis of an algorithm for Gröbner bases in *Theorema*, by lazy thinking, starting with some knowledge (on the reduction relation, Church-Rosser property and confluence, Newman's lemma, polynomial rings, etc. – see [Winkler 1996] or any other textbook on Gröbner bases) and with the Critical Pair Completion algorithm scheme (first identified in [Buchberger 1987]).

Having carried out the synthesis, the attention will turn to the exploration of the knowledge that in the synthesis exploration was taken for granted, i.e. reduction relations, Church-Rosser, Newman's lemma, polynomial rings.

Another research direction is the investigation whether Gröbner bases algorithms can be synthesized using algorithm schemes other than the Critical Pair Completion scheme.

- **Special Focus: Synthesis in the Context of Systematic Theory Exploration**

In both case studies, a special emphasis is put on synthesis. The main research directions here are: development of algorithm scheme libraries, and improvement of the lazy thinking method for synthesis.

So far, in *Theorema*, only divide-and-conquer (and related) and Critical–Pair Completion algorithm schemes were considered. Other algorithm schemes are well-known in the literature (e.g. backtracking, dynamic programming, local search, etc.), and they should be investigated in the context of lazy thinking.

Other problems of interest are termination of the synthesized algorithms, efficiency issues, data types.

Recently, the idea of preprocessing of lazy thinking by using problem schemes was introduced in [Buchberger Craciun 2004a], for the divide-and-conquer algorithm scheme. The essence of this approach is to apply lazy thinking at a very abstract level and use the results for domains that are instances of that abstract level (or satisfy the axioms that describe this level). This approach should be undertaken in each of the case studies. For the Gröbner bases case study a similar effort was [Buchberger 1984], the result of which was the notion of reduction rings.

■ 3.3. Method of Research

As described in [Craciun et al 2004], in the investigation of both major case studies we will employ the following:

– **"pen and paper"** (or "screen and keyboard") **case studies**, both in the theory of tuples and the theory of Gröbner bases. This way the ideas of the 4 thread will be tried out in selected examples. The aim of this phase is to clarify what needs to be done for the implementation of support for the 4–model thread, and in particular for the support of exploration in the chosen case studies;

– **implementation phase**, in which the ideas coming from the case studies ("pen and paper" initially) are implemented in the system;

– **computer supported exploration** of the theories, in which the implementation is used in the concrete cases chosen for exploration.

In fact, this is not a linear process. In our project we will move many times between the 3 stages, in cycles. This is a natural process in mathematics – the creativity spiral in mathematics of Bruno Buchberger [Buchberger 1993], applied to **the level of developing methods** (for theory exploration).

As mentioned in the previous section, in the *SysteMaThEx* project, the focus of exploration is on the (automated or, at least, computer–supported) construction of the theory thread, in each case. For this, there are two approaches:

The bottom-up approach: Start from some knowledge (e.g. axioms initially), then (invent and) introduce new concepts by the use of definition schemes (from the ideas thread); ask questions about the properties of the new notions (theorem schemes), thus obtaining new conjectures. A second source of conjectures is the investigation of the interaction of the new notions with the already known notions. Once conjectures were generated, try to prove them by employing reasoners (4th thread). Similarly, problems can be generated systematically by the use of problem schemes and algorithms for their solution can be proposed by algorithms schemes. By proving theorems (including correctness theorems for algorithms) using (automated) provers, reasoning objects are produced (3rd thread), which may later be post−processed, for example, by proof simplifiers. All this represents one exploration cycle.

The top-down approach: This approach is used to solve a problem (introduced in the theory by the use of a problem scheme), or to prove a theorem (either a conjecture generated in the bottom-up construction of the theory, or a conjecture generated by the use of a theorem scheme). We will apply the **lazy thinking method for algorithm synthesis** (see [Buchberger 2003], [Buchberger Craciun 2004]) to synthesize solutions to problems, or the lazy thinking method for invention of theorems ([Buchberger 2000]) to prove a theorem (theorems that are missing from the knowledge base are invented and proved). The lazy thinking method for algorithm synthesis makes use of algorithm schemes.

In fact, systematic theory exploration is an interplay between the bottom-up and top-down approaches.

Each of the 4 threads will be addressed in our exploration. However, exploration of the other threads will be addressed.

Building up libraries of schemes requires a careful analysis of the types of mathematical knowledge, from the general mathematical knowledge as found in textbooks, and also the use of the observations from the case studies (abstract away the details from the concrete examples):

• In the case of **definition schemes**:

− analyze the possible types of definitions: explicit definitions of predicate symbols, explicit definitions of function symbols, implicit non-unique definitions of function symbols, implicit unique definitions of function symbols;

− construct by syntactic analysis, definition schemes. For example this is possible in the case of recursive definitions. Examples:

$$\forall_{f,\,is,\,s,\,g,\,h} \; \texttt{is-simple-recursion[f, is, s, g, h]} \Leftrightarrow \forall_{x} \; \texttt{f[x]} = \begin{cases} \texttt{s[x]} & \Leftarrow \; \texttt{is[x]} \\ \texttt{g[f[h[x]]]} & \Leftarrow \; \texttt{otherwise} \end{cases}'$$

$$\forall_{f,\,is,\,s,\,g,\,h_1,\ldots,\,h_k} \; \texttt{is-divide-and-conquer[f, is, s, g, } h_1\texttt{, ..., } h_k\texttt{]} \Leftrightarrow$$

$$\forall_{x} \; \texttt{f[x]} = \begin{cases} \texttt{s[x]} & \Leftarrow \; \texttt{is[x]} \\ \texttt{g[x, f[}h_1\texttt{[x]], ..., f[}h_k\texttt{[x]]]} & \Leftarrow \; \texttt{otherwise} \end{cases}'$$

where "f","is", "s", "g", "h", "h_1", …, "h_k" are second order variables, "is-simple-recursion", "is−divide−and−conquer" are predicate constants (from a set of fixed constants − the names of the definition schemes), "x" is a first order variable. The formulae are expressed in the *Theorema* syntax, and represent increasingly complex recursive schemes.

• In the case of **theorem schemes**:

Examples of possible schemes (details omitted) are "is−equivalence−relation", "is−partial−ordering", "is−commutative−ring", and they can be applied to notions (objects) in the knowledge base.

• In the case of **problem schemes**:

Examples of problem schemes are:

− $\forall_{F}$ `is-finite-Gröbner-basis[F, Gb[F]]`, where F ranges over sets of multivariate polynomials in a fixed number of indeterminates over a fixed coefficient field, "Gb" is an unknown function (algorithm); this is the **explicit problem** specification for the Gröbner bases algorithm, "is−finite−Gröbner−basis" is a predicate constant;

− the problem of canonical simplification is an **implicit problem**:

$\forall_{x}\ (x \sim f[x]),$

$\forall_{x,y}\ (x \sim y \Rightarrow (f[x] = f[y])),$

where "~" is an equivalence, and "f " is an unknown function, the canonical simplifier.

• In the case of **algorithm schemes**:

Algorithm schemes are very similar to definition schemes (in fact they represent abstract definitions of algorithms), they are only used in a different way (not to introduce explicitly notions in the theory, but as solutions to problems, using the lazy thinking method for algorithm synthesis). That is why approaches similar to the ones for the definition schemes will be used. Algorithm libraries thus built will be compared/completed with results in scheme–based synthesis literature, such as [Smith 1996].

For each of the case studies, **the theory thread** will be developed in exploration cycles, starting from the axioms bottom-up, and solving problems in a top–down manner, as a side–effect filling up the **reasoning objects thread**.

The theory of tuples (and related theories) will be developed starting with the axioms. We will follow the general structure of the development of the theories in [Manna Waldinger 1985], although,this will not be a formalization of the Manna and Waldinger book, which will merely be used for comparison. For the exploration of the theory of Gröbner bases, the following will also be explored: reduction relations, polynomial ideal theory. Polynomials will be encoded as tuples.

For the theory of tuples we will use as reasoners the *Theorema* sequence basic prover (induction on tuples) combined with other basic provers such as predicate logic, simplifier, case distinction provers. In particular we will aim to integrate the new sequence equational provers developed by Temur Kutsia, see [Kutsia 2003], in the frame of *Theorema*.

For the theory of Gröbner bases we need to develop proof methods based on the sequence prover, the set theory prover and the sequence equational provers. The development of these provers will be carried out in parallel with the exploration of the theories.

The exploration of the reasoners thread is technically more delicate. The reasoners used from the **reasoners thread** could be proved correct using the method described by Bruno Buchberger in his 4 threads model, formulating the correctness of the prover as an explicit correctness problem in the *Theorema* language and using the system to prove this (employing "weaker" provers that were already proved correct), and this research direction will be undertaken in the *SysteMaThEx* project.

One other important aspect in our proposal is the **documentation** of exploration process and of the implementation of the codes. The documentation of the exploration process will come in the form of internal reports of working group. The documentation of code will be done in a systematic way and will be insisted upon during the implementation process.

4. The *SysteMaThEx* Project/Group – Organizational Aspects

According with the goals of the specific European Commission programme, this project supports in fact, the establishment of a research group in the area of automated reasoning,

This section outlines the main aspects concerned with the organization of the project: hiring personnel, purchasing equipment and books, organization of the seminar and working group, other administrative tasks.

- **Hiring Personnel**

The *SysteMaThEx* proposal budget contains resources to support the hiring of 3 people to work in the project. As outlined in the proposal, this project aims to support 3 master–level people to carry out their work and write a master thesis in the field of MKM.

The candidates for these positions should meet the following requirements:

– they should be last-year students of mathematics, computer mathematics or computer science, or first year master students in a computer science / mathematics program, preferably (*) at the West University of Timisoara, or the "Politehnica" University of Timisoara;

– posses good knowledge background in mathematics and logic (of particular interest: logic, (computer) algebra, set theory etc.);

– good background in computer science (of particular interest: functional programming – *Mathematica; Java*, web design/implementation, text editing – LaTex, *Linux* represent advantages);

– good knowledge of the English language (reading, writing, speaking), as the official language of the project is English.

(*) **Note:** The reason for the apparent restriction of the candidates to being students of the two universities from Timisoara, is motivated by both the objective of the European Program (reintegration of the researcher at the host institution) and the objectives of the Institute e–Austria (opportunities for local researchers). In case these positions will not be filled, the above restriction will be relaxed.

The positions represent a half-time job and will offer the following:

– training in the fields of automated theorem proving (*Theorema*), logic, mathematics, MKM,
– a chance to live at the edge of a new research field,
– a chance to be in contact with the newest developments in the field, and meet leading scientists,
– further career prospects (PhD studies).

Note: In filling the positions, the equal opportunity policy (promotion of women in research) of the European Commission will be observed.

- **Purchasing Equipment**

This will be done in accordance with the rules of the host institution.

- **Project Seminar**

A weekly project seminar will be established, as a venue for presentation of work in progress, literature reports, talks by guests, conference reports, etc. Given the major scientific objectives of the project, special sessions – **theory exploration parties** – where work on the case studies will be carried out by the whole group. The seminar, as well as the theory exploration parties, will be open to all people interested. Moreover, students (and faculty) of mathematics, computer science and related departments will be encouraged to participate.

As well as the seminar, individual meetings of the project leader with the participants in the project will be organized periodically, that will allow discussion and monitoring of progress of work, etc.

- **General Administrative Tasks**

In addition to the scientific work in the project, some other tasks will be assigned to each participant in the project. Such additional tasks include:

– setting up and maintaining the equipment,
– development of a web–page for the project,
– writing reports (in addition to scientific reports such as literature surveys – "where do we live?" and work progress reports "what do we do there?", write milestone reports for the European Commission, etc.).

5. The *SysteMaThEx* Project – Work Plan

Time line	Scientific Tasks	Administrative Tasks
0 – 6	1. Training in Theorema. 2. Pen-and-paper case studies : – theory of tuples, – synthesis in the theory of tuples using divide and conquer algorithms. 3. Synthesis of Gröbner bases algorithms.	Hiring personnel. Purchasing equipment. Purchasing books. Web page for the project. Research visits to RISC. Periodic activity reports. Management report.
6 – 12	1. General exploration in the theory of Gröbner bases. 2. Development / improvement of the Theorema provers employed in the exploration. 3. Specification of concrete master theses : – bottom-up exploration in the theory of tuples, – automated synthesis in the theory of tuples – general exploration in polynomial rings. 4. Dissemination of results (2 – 3 conference papers).	Periodic activity reports. Mid-term review. Management report.
12 – 24	1. Further exploration in the two theories. 2. Investigation of non-CPC approaches to Gröbner Bases algorithm synthesis. 3. Dissemination of results (3 masters thesis, conference papers).	Purchasing equipment. Purchasing software. Purchasing books. Final management report. Final activity report.

The project is expected to start at the beginning of March 2005, and it will be 24 months long.

6. Conclusions

We have attempted to give an (rather detailed) overview of the *SysteMaThEx* project, due to begin shortly (beginning of March 2005) at the e–Austria Institute in Timisoara. We hope that this manages to offer both a clear image of the objectives of this project, and a clear localization of this project in the field of MKM.

Undoubtedly, there will be some disappointment for the reader who wanted the promises made at the beginning of this report fulfilled (a detailed motivation for the case studies, with more examples). This, however goes beyond the scope of this report (from practical reasons). Plenty of literature pointers have been provided, so the disappointed reader should have a solution to address this apparent shortcoming and unfulfilled promises.

Some of the examples might be too technical and gaps may be present, that are not motivated (such as *Theorema* commands, options, syntax). The author hopes that this too is not considered a shortcoming, but rather a teaser, inciting the reader (and possible candidate for a position in the project) to take up studying the problems introduced here.

SysteMaThEx is an exciting undertaking, with lots of interesting directions to be explored. It is the hope of this author that this point will be clear to those interested, and that soon enough, with the start of the project, results that will meet these hopes and expectations will appear.

References

[Automath]
N.G. de Brujin. The mathematical language Auutomath, its usage, and some of its expressions. In M. Laudet, D. Lacombe, L. Nolin, and M. Schützenberger (eds.), *Proc. of Symposium on Automatic Demonstration*, vol 125 of *LN in Mathematics*, pp. 29–61. Springer 1970.

[Buchberger 1984]
B. Buchberger. A Critical–Pair/Completion Algorithm in Reduction Rings. *Proc. of "Rekursive Kombinatorik"*, Munster, Germany, May 1983. LNCS 171, Springer 1984, pp. 137–161.

[Buchberger 1987]
B. Buchberger. History and Basic Features of the Critical–Pair / Completion Procedure. *Journal of Symbolic Computation* (1987) **3**, pp. 3–38, 1987.

[Buchberger 1993]
B. Buchberger. Mathematica: A system for doing mathematics by computer?. Technical Report **93–50**, RISC, Johannes Kepler University, 1993.

[Buchberger 1998]
B. Buchberger. Introduction to Gröbner Bases. In: *Gröbner Bases and Applications* (B. Buchberger, F. Winkler, eds.), London Mathematical Society Lecture Notes Series 251, Cambridge University Press, 1998, pp.3–31.

[Buchberger 2000]
B. Buchberger. Theory Exploration with Theorema*,* Analele Universitatii din Timisoara, Seria Matematica–Informatica, **Vol. XXXVIII**, Fasc. **2**, 2000, (*Proceedings of SYNASC 2000*, 2nd International Workshop on Symbolic and Numeric Algorithms In Scientific Computing, Oct. 4–6, 2000, Timisoara, Romania T. Jebelean, V. Negru, A. Popovici, eds.), pp.9–32.

[Buchberger 2003]
B.Buchberger. Algorithm Invention and Verification by Lazy Thinking. In: D. Petcu, V. Negru, D. Zaharie, T. Jebelean (eds.), *Proceedings of SYNASC 2003*, 5th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, October 1–4, 2003, Mirton Publishing, pp.2–26.

[Buchberger 2003a]
B. Buchberger. Groebner Rings in TH∃OREM∀: A Case Study in Functors and Categories. SFB Technical Report 2003–49, SFB "Scientific Computing", Mathematical Institutes, Johannes Kepler University, Linz, Austria, Version 2003–11–15.

[Buchberger 2004a]
B.Buchberger. Computer–Supported Mathematical Theory Exploration: Schemes, Failing Proof Analysis, and Metaprogramming. RISC Technical Report, 4 June, 2004.

[Buchberger 2004b]
B. Buchberger. Algorithm Supported Mathematical Theory Exploration. In: *Proceedings of AISC 2004 (7 th International Conference on Artificial Intelligence and Symbolic Computation)*, B. Buchberger, John Campbell (ed.), Springer Lecture Notes in Artificial Intelligence Vol. **3249**, pp. 236–250. 22–24 September 2004. Copyright: Springer, Berlin–Heidelberg, RISC, Johannes Kepler University, Austria.

[Buchberger 2004c]
B.Buchberger. Towards the Automated Synthesis of a Gröbner Bases Algorithm. RACSAM (Rev.R.Acad.-Cien.Serie A.Mat), 2004, to appear.

[Buchberger Craciun 2004]
B. Buchberger, A. Craciun. Algorithm Synthesis by Lazy Thinking: Examples and Implementation in Theorema. *Proceedings of the Mathematical Knowledge Management Symposium*, Electronic Notes in Theoretical Computer Science,Volume **93**, 18 February 2004, pp 24–59.

[Buchberger Craciun 2004a]
B. Buchberger, A. Craciun. Algorithm Synthesis by Lazy Thinking: Using Problem Schemes. In: D. Petcu, V. Negru, D. Zaharie, T. Jebelean (eds.), *Proceedings of SYNASC 2004, 6th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing*, Timisoara, Romania, 2004, Mirton Publishing, ISB.

[Buchberger Nakagawa 2004]
Bruno Buchberger, Koji Nakagawa. Mathematical Knowledge Editor: A Research Plan, Sept. 2004.

[Buchberger et al 2000]
B.Buchberger, C.Dupre, T.Jebelean, F.Kriftner, K.Nakagawa, D.Vasaru, W.Windsteiger. The Theorema Project: A Progress Report. In: M.Kerber and M.Kohlhase (eds.), *Symbolic Computation and Automated Reasoning (Proceedings of CALCULEMUS 2000, Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, August 6−7, 2000, St.Andrews, Scotland)*, A.K.Peters, Natick, Massachusetts, ISBN 1−56881−145−4, pp.98−113.

[Buchberger et al 2004]
B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, N. Popov, J. Robu, W. Windsteiger, F. Piroi, M. Rosenkranz. Theorema: Towards Systematic Mathematical Theory Exploration, Submitted to the Journal of Applied Logic.

[Buchberger et al 2005]
B. Buchberger and the Theorema group. The Th∃orem∀ Project @ RISC and RICAM, a presentation brochure for the Theorema Project.

[Bundy et al 2003]
A. Bundy, S. Colton, S. Huczynska, R. McCasland. New Directions in Automated Conjecture Making. In *Proceedings of the Automated Reasoning Workshop*, 2003.

[Colton 2002]
S. Colton. *Automated Theory Formation in Pure Mathematics (Distinguished Dissertations Series)*. Springer Verlag, 2002.

[Colton et al 2000]
S. Colton, A. Bundy, T. Walsh. On the Notion of Interestingness in Automated Mathematical Discovery. International Journal of Human Computer Studies, Vol. **53**, No. 3, pp. 351−375, 2000.

[Coq]
Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions* Series: Texts in Theoretical Computer Science. An EATCS Series 2004, XXV, 469 p., Hardcover.

[Craciun 2002]
A. Craciun. The Sequence Provers in Theorema. *Calculemus 2002, Work in Progress Papers*, Seki−Report Nr. SR−02−04, Universität des Saarlandes.

[Craciun et al 2004]
A. Craciun, D. Petcu for IeAT: "SysteMaThEx" Systematic Mathematical Theory Exploration in the Theorema System: Case Studies. Project proposal.

[FDL]
S.F. Allen, M. Bickford, R. L. Constable, R. Eaton, C. Kreitz, and L. Lorigo. FDL: A Prototype Formal Digital Library. Unpublished manuscript, Cornell University, 2002.

[Gries Schneider 1993]
David Gries, Fred B. Schneider. *A Logical Approach to Discrete Math*. Springer−Verlag New York 1993.

[Helm]
A. Asperti, L. Padovani, C. Sacerdoti Coen, F. Guidi , I Schena. Mathematic knowledge management in Helm. *Annals of Mathematics and Artificial Intelligence,* 38 (1): 27-46, 2003.

[Imps]
W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: an Interactive Mathematical Proof System. Journal of Automated Reasoning, **11**: 213−248, 1993.

[Isabelle]
L. Paulson. Isabelle: the next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pp 361−386. Academic Press 1990.

[JFM]
The Journal of Formalized Mathematics, http://mizar.org/JFM/.

[Kutsia 2003]
T. Kutsia. The Equational Prover of THEOREMA. In R. Nieuwenhuis (editor), *Rewriting Techniques and Applications, 14th International Conference, RTA−03,* Valencia, Spain, Jun 9−11, 2003, Springer Verlag LNCS 2706, pp 367−379.

[Manna Waldinger 1985]
Z. Manna, R. Waldinger, *The Logical Basis for Computer Programming − Volume 1 − Deductive Reasoning*, Addison−Wesley Publishing Company, ISBN 0−201−18260−2 (v. 1), 1985.

[MathML]
W3C Math Home:What is MathML?(http://www.w3.org/Math/)

[MKM 2001]
B. Buchberger, O. Caprotti (eds.). *Mathematical Knowledge Management.* Electronic Proceedings of the 1st International Workshop on Mathematical Knowledge Management, RISC, Schloss Hagenberg, Austria, Sep.24−26, 2001, http://www.risc.uni−linz.ac.at/institute/conferences/MKM2001/Proceedings/

[MKM 2003]
A. Asperti, B. Buchberger, J. Davenport. *Mathematical Knowledge Management. Proceedings of the Second International Conference on Mathematical Knowledge Management (MKM 2003)*, Bertinoro, Italy, Feb.16−18, 2003, Lecture Notes in Computer Science,Vol.2594, Springer, Berlin−Heidelberg−New York, 2003, 223 pages.

[Mizar]
A. Trybulec, H.A. Blair. Computer aided reasoning. In R. Parikh, ed, *Proc. of the conference Logic of Programs*, vol. 193 of *LNCS*, pp. 406−412. Springer 1985.

[Nakagawa et al 2004]
K. Nakagawa, A. Nomuro, M. Suzuki. Extraction of Logical Structure from Articles in Mathematics. In *Proceedings of the Third International Conference on Mathematical Knowledge Management (MKM 2004)*, Bialowieza, Poland, September 19−21, LNCS 3119, Springer Heidelberg-Berlin-New York, 2004.

[Nist]
D.W.Lozier. NIST Digital Library of Mathematical Functions. In Annals of Mathematics and Artificial Intelligence −− Special Issue on Mathematical Knowledge Management.

[Omega]
J. Siekmann and C. Benzmüller. Omega: Computer supported mathematics. In *Proc. of the 27th German Conference on Artificial Intelligence, KI'04*, Ulm, Germany, 2004. To appear.

[OpenMath]
O. Caprotti, D. Carlisle, OpenMath and MathML: Semantic Mark Up for Mathematics. In ACM Crossroads, ACM Press, 1999.

[Piroi Buchberger 2002]
F. Piroi, B. Buchberger. Focus Windows: A New Technique for Proof Presentation. SFB Report No. 02−32, Johannes Kepler University Linz, Spezialforschungsbereich F013. Technical report, December 2002.

[Piroi Buchberger 2004]
Florina Piroi, Bruno Buchberger. An Environment for Building Mathematical Knowledge Libraries, in

Proceedings of the Third International Conference on Mathematical Knowledge Management (MKM 2004), Bialowieza, Poland, September 19−21, LNCS 3119, Springer Heidelberg-Berlin-New York, 2004.

[Smith 1996]
D. R. Smith. Toward a Classification Approach to Design. *Proceedings of the Fifth International Conference on Algebraic Methodology and Software Technology, AMAST'96*, LNCS, Springer Verlag, 1996, 62−84.

[Smith 1999]
D.R.Smith. Mechanizing the Development of Software. In M. Broy and R. Steinbrueggen (eds.), *Calculational System Design, Proceedings of the NATO Advanced Study Institute*, IOS press, Amsterdam, 1999, pp.251−292.

[TheoremaWeb]
Theorema Web Page: http://www.theorema.org

[Tomuta 1998]
E. Tomuta. Functor Proving in *Theorema*. PhD Thesis, Research Institute for Symbolic Computation, Johannes Kepler University, A4040 Linz, Austria, 1998.

[TPTP]
G. Sutcliffe, C. Suttner, The TPTP Problem Library for Automated Theorem Proving, http://www.cs.miami.edu/~tptp/ .

[Winkler 1996]
F. Winkler, *Polynomial algorithms in computer algebra*, Springer Verlag Wien 1996.

Appendix

- **Specification of the Problem of sorting**

  We want to sort tuples. Thus, we are looking for a unary function

```
sorted[X]
```

  which satisfies the following (specification) property:

```
Theorem["correctness of sort", any[is-tuple[X]],
 is-sorted-version[X, sorted[X]]
]
```

  Of course, the predicate **is-sorted-version** is known. In the next section, we will make explicit the theory in which we work (tuples expressed recursively –from the left – in terms of sequence variables).

- **Algorithm Type: Divide and Conquer for Tuples**

```
Algorithm["sorted", any[is-tuple[X]],
 sorted[X] =
  { special[X]                                                  ⇐ is-trivial-tuple
  { merged[sorted[left-split[X]], sorted[right-split[X]]]       ⇐ otherwise
    ]
```

  Some (type conservation mainly) specifications are available on the auxiliary functions:

```
Lemma["closure of special", any[X],
 with[is-tuple[X] ∧ is-trivial-tuple[X]],
 is-tuple[special[X]]]

Lemma["splits are tuples", any[X],
 with[is-tuple[X] ∧ ¬ is-trivial-tuple[X]],
 is-tuple[left-split[X]]
 is-tuple[right-split[X]] ]

Lemma["splits are shorter",
 any[is-tuple[X]], with[¬ is-trivial-tuple[X]],
 X > left-split[X]
 X > right-split[X] ]

Lemma["closure of merge", any[is-tuple[X, Y]],
 is-tuple[merged[X, Y]]]
```

- **The Prove command for applying the Lazy Thinking method to synthesize a sorting algorithm and the trace:**

```
Prove[Theorem["correctness of sort"],
  using → Theory["lazy thinking correctness of sort"],
  by → CascadeLT[SqnsEqCasePC, GenerateConjectures,
    nS, aNS, kConjectures, nConjectures],
  built-in → {Property[ ≈ → {Commutative}]},
  ProverOptions → {SqnsProofTechniques → {"WFInd"},
    SqnsWFOrdering → •[ > ], GRWTarget → {"kb", "goal"},
    AllowIntroduceQuantifiers → True}, SearchDepth → 50] // Timing
```

```
LAZY THINKING::::: The proof fails.
 After analysing the failing proof, the
   following conjecture is added to the knowledge base:
 •lf[Lemma (conjecture15): conjecture15,
     ∀_{X1, is-tuple[X1]}  (is-trivial-tuple[X1] ∧ is-sorted[X1] ⇒ (special[X1] = X1)),
  •finfo[]]
 Now attempt the proof with the updated knowledge base.

LAZY THINKING::::: The proof fails.
 After analysing the failing proof, the
   following conjecture is added to the knowledge base:
 •lf[Lemma (conjecture44): conjecture44,
     ∀_{X2, X3, X4, is-tuple[X4]}  (is-tuple[X2] ∧ left-split[X4] ≈ X2 ∧ is-sorted[X2] ∧
      is-tuple[X3] ∧ right-split[X4] ≈ X3 ∧ is-sorted[X3] ∧
      ¬ is-trivial-tuple[X4] ⇒ merged[X2, X3] ≈ X4), •finfo[]]
 Now attempt the proof with the updated knowledge base.

LAZY THINKING::::: The proof fails.
 After analysing the failing proof, the
   following conjecture is added to the knowledge base:
 •lf[Lemma (conjecture46): conjecture46,
     ∀_{X5, X6, X7, is-tuple[X7]}  (is-tuple[X5] ∧ left-split[X7] ≈ X5 ∧ is-sorted[X5] ∧ is-tuple[X6] ∧
      right-split[X7] ≈ X6 ∧ is-sorted[X6] ∧ ¬ is-trivial-tuple[X7] ⇒
     is-sorted[merged[X5, X6]]), •finfo[]]
 Now attempt the proof with the updated knowledge base.


 LAZY THINKING ::::: The proof is completed!!!!
```

- **The proof of correctness of merge-sort, produced automatically by *Theorema:***

  Note that although it is long, it is readable – even more so in electronic form, since the Mathematica cell structure makes parsing the notebook very easy:

Prove:

(Theorem (correctness of sort)) $\forall_{\text{is-tuple}[X]}$ is-sorted-version[X, sorted[X]],

under the assumptions:

(Definition (is sorted): 1) is-sorted[⟨⟩],

(Definition (is sorted): 2) $\forall_{x}$ is-sorted[⟨x⟩],

(Definition (is sorted): 3)

$\forall_{x, y, \bar{z}}$ (is-sorted[⟨x, y, $\bar{z}$⟩] ⇔ x ≥ y ∧ is-sorted[⟨y, $\bar{z}$⟩]),

(Definition (is permuted version): 1) ⟨⟩ ≈ ⟨⟩,

(Definition (is permuted version): 2) $\underset{y,\bar{y}}{\forall}\ (\langle\rangle \not\approx \langle y, \bar{y}\rangle)$,

(Definition (is permuted version): 3)

$\underset{x,\bar{x},\bar{y}}{\forall}\ (\langle\bar{y}\rangle \approx \langle x, \bar{x}\rangle \Leftrightarrow x \in \langle\bar{y}\rangle \wedge \mathtt{dfo}[x, \langle\bar{y}\rangle] \approx \langle\bar{x}\rangle)$,

(Definition (is sorted version)) $\underset{\substack{X,Y \\ \mathtt{is\text{-}tuple}[X]}}{\forall}\ (\mathtt{is\text{-}sorted\text{-}version}[X, Y] \Leftrightarrow\ \mathtt{is\text{-}tuple}[Y] \wedge X \approx Y \wedge \mathtt{is\text{-}sorted}[Y])$,

(Proposition (is tuple tuple)) $\underset{\bar{x}}{\forall}\ \mathtt{is\text{-}tuple}[\langle\bar{x}\rangle]$,

(Definition (prepend): $\frown$) $\underset{x,\bar{y}}{\forall}\ (x \frown \langle\bar{y}\rangle = \langle x, \bar{y}\rangle)$,

(Proposition (singleton tuple is singleton tuple)) $\underset{x}{\forall}\ \mathtt{is\text{-}singleton\text{-}tuple}[\langle x\rangle]$,

(Definition (is trivial tuple)) $\underset{\mathtt{is\text{-}tuple}[X]}{\forall}\ (\mathtt{is\text{-}trivial\text{-}tuple}[X] \Leftrightarrow\ \mathtt{is\text{-}empty\text{-}tuple}[X] \vee \mathtt{is\text{-}singleton\text{-}tuple}[X])$,

(Definition (is element): 1) $\underset{x}{\forall}\ (x \notin \langle\rangle)$,

(Definition (is element): 2) $\underset{x,y,\bar{y}}{\forall}\ (x \in \langle y, \bar{y}\rangle \Leftrightarrow (x = y) \vee x \in \langle\bar{y}\rangle)$,

(Definition (deletion of the first occurrence): 1) $\underset{a}{\forall}\ (\mathtt{dfo}[a, \langle\rangle] = \langle\rangle)$,

(Definition (deletion of the first occurrence): 2)

$\underset{a,x,\bar{x}}{\forall}\ (\mathtt{dfo}[a, \langle x, \bar{x}\rangle] = \|\langle\bar{x}\rangle \Leftarrow x = a,\ x \frown \mathtt{dfo}[a, \langle\bar{x}\rangle] \Leftarrow \mathtt{otherwise}\|)$,

(Definition (is longer than): 1) $\underset{\bar{y}}{\forall}\ (\langle\rangle \not> \langle\bar{y}\rangle)$,

(Definition (is longer than): 2) $\underset{x,\bar{x}}{\forall}\ (\langle x, \bar{x}\rangle > \langle\rangle)$,

(Definition (is longer than): 3) $\underset{x,\bar{x},y,\bar{y}}{\forall}\ (\langle x, \bar{x}\rangle > \langle y, \bar{y}\rangle \Leftrightarrow \langle\bar{x}\rangle > \langle\bar{y}\rangle)$,

(Proposition (trivial tuples are sorted)) $\underset{\substack{\bar{x} \\ \mathtt{is\text{-}trivial\text{-}tuple}[\langle\bar{x}\rangle]}}{\forall}\ \mathtt{is\text{-}sorted}[\langle\bar{x}\rangle]$,

(Proposition (only trivial tuple permuted version of itself))

$\underset{\substack{\bar{x},Y \\ \mathtt{is\text{-}trivial\text{-}tuple}[\langle\bar{x}\rangle]}}{\forall}\ ((Y = \langle\bar{x}\rangle) \Rightarrow Y \approx \langle\bar{x}\rangle)$,

(Proposition (reflexivity of permuted version)) $\underset{\bar{x}}{\forall}\ (\langle\bar{x}\rangle \approx \langle\bar{x}\rangle)$,

(Algorithm (sorted))

$$\underset{\mathtt{is\text{-}tuple}[X]}{\forall}\ (\mathtt{sorted}[X] = \|\mathtt{special}[X] \Leftarrow \mathtt{is\text{-}trivial\text{-}tuple}[X],\ \mathtt{merged}[\mathtt{sorted}[\mathtt{left\text{-}split}[X]], \mathtt{sorted}[\mathtt{right\text{-}split}[X]]] \Leftarrow \mathtt{otherwise}\|)$$

,

(Lemma (closure of special)) $\forall_{\substack{X \\ \text{is-tuple}[X] \wedge \text{is-trivial-tuple}[X]}} \text{is-tuple}[\text{special}[X]]$,

(Lemma (splits are tuples): 1) $\forall_{\substack{X \\ \text{is-tuple}[X] \wedge \neg \text{is-trivial-tuple}[X]}} \text{is-tuple}[\text{left-split}[X]]$

,

(Lemma (splits are tuples): 2)

$$\forall_{\substack{X \\ \text{is-tuple}[X] \wedge \neg \text{is-trivial-tuple}[X]}} \text{is-tuple}[\text{right-split}[X]],$$

(Lemma (splits are shorter): 1) $\forall_{\substack{\text{is-tuple}[X] \\ \neg \text{is-trivial-tuple}[X]}} (X > \text{left-split}[X])$,

(Lemma (splits are shorter): 2) $\forall_{\substack{\text{is-tuple}[X] \\ \neg \text{is-trivial-tuple}[X]}} (X > \text{right-split}[X])$,

(Lemma (closure of merge)) $\forall_{\substack{\text{is-tuple}[X] \\ \text{is-tuple}[Y]}} \text{is-tuple}[\text{merged}[X, Y]]$,

(Lemma (conjecture15): conjecture15)

$$\forall_{\substack{X1 \\ \text{is-tuple}[X1]}} (\text{is-trivial-tuple}[X1] \wedge \text{is-sorted}[X1] \Rightarrow (\text{special}[X1] = X1)),$$

(Lemma (conjecture44): conjecture44)

$$\forall_{\substack{X2, X3, X4 \\ \text{is-tuple}[X4]}} (\text{is-tuple}[X2] \wedge \text{left-split}[X4] \approx X2 \wedge \text{is-sorted}[X2] \wedge \text{is-tuple}[X3] \wedge \text{right-split}[X4] \approx X3 \wedge \text{is-sorted}[X3] \wedge \neg \text{is-trivial-tuple}[X4] \Rightarrow \text{merged}[X2, X3] \approx X4),$$

(Lemma (conjecture46): conjecture46)

$$\forall_{\substack{X5, X6, X7 \\ \text{is-tuple}[X7]}} (\text{is-tuple}[X5] \wedge \text{left-split}[X7] \approx X5 \wedge \text{is-sorted}[X5] \wedge. \text{is-tuple}[X6] \wedge \text{right-split}[X7] \approx X6 \wedge \text{is-sorted}[X6] \wedge \neg \text{is-trivial-tuple}[X7] \Rightarrow \text{is-sorted}[\text{merged}[X5, X6]])$$

We prove (Theorem (correctness of sort)) by well-founded induction on X.

Well-founded induction:

Assume:

(1) $\text{is-tuple}[\langle \overline{X_0} \rangle]$.

Well-Founded Induction Hypothesis:

(2) $\forall_{\text{is-tuple}[x4]} (\langle \overline{X_0} \rangle > x4 \Rightarrow \text{is-sorted-version}[x4, \text{sorted}[x4]])$

We have to show:

(3) $\texttt{is-sorted-version}[\langle \overline{X_0} \rangle, \texttt{sorted}[\langle \overline{X_0} \rangle]]$.

We prove (3) by case distinction using (Algorithm (sorted)).

Case 1:

(4) $\texttt{is-trivial-tuple}[\langle \overline{X_0} \rangle]$.

Hence, we have to prove

(5) $\texttt{is-sorted-version}[\langle \overline{X_0} \rangle, \texttt{special}[\langle \overline{X_0} \rangle]]$.

Formula (4), by (Proposition (trivial tuples are sorted)), implies:

(9) $\texttt{is-sorted}[\langle \overline{X_0} \rangle]$.

Formula (4), by (Proposition (only trivial tuple permuted version of itself)), implies:

(10) $\forall_{\boldsymbol{Y}} \; (\boldsymbol{Y} = \langle \overline{X_0} \rangle) \Rightarrow \boldsymbol{Y} \approx \langle \overline{X_0} \rangle$.

Formula (1) and (4), by (Lemma (closure of special)), implies:

(11) $\texttt{is-tuple}[\texttt{special}[\langle \overline{X_0} \rangle]]$.

Formula (1) and (4), by (Lemma (conjecture15): conjecture15), implies:

(13) $\texttt{special}[\langle \overline{X_0} \rangle] = \langle \overline{X_0} \rangle$.

Formula (5), using (13), is implied by:

(21) $\texttt{is-sorted-version}[\langle \overline{X_0} \rangle, \langle \overline{X_0} \rangle]$.

Formula (21), using (Definition (is sorted version)), is implied by:

(22) $\texttt{is-tuple}[\langle \overline{X_0} \rangle] \wedge \langle \overline{X_0} \rangle \approx \langle \overline{X_0} \rangle \wedge \texttt{is-sorted}[\langle \overline{X_0} \rangle]$.

We prove the individual conjunctive parts of (22):

Proof of (22.1) $\texttt{is-tuple}[\langle \overline{X_0} \rangle]$:

Formula (22.1) is true because it is identical to (1).

Proof of (22.2) $\langle \overline{X_0} \rangle \approx \langle \overline{X_0} \rangle$:

Formula (22.2) is true by (10).

Proof of (22.3) $\texttt{is-sorted}[\langle \overline{X_0} \rangle]$:

Formula (22.3) is true because it is identical to (9).

Case 2:

(6) $\neg\, \texttt{is-trivial-tuple}[\langle \overline{X_0} \rangle]$.

Hence, we have to prove

(8) $\texttt{is-sorted-version}[\langle \overline{X_0} \rangle, \texttt{merged}[\texttt{sorted}[\texttt{left-split}[\langle \overline{X_0} \rangle]], \texttt{sorted}[\texttt{right-split}[\langle \overline{X_0} \rangle]]]]$.

From (6) , by (2), (Lemma (splits are tuples): 1), (Lemma (splits are tuples): 2), (Lemma (splits are shorter): 1), (Lemma (splits are shorter): 1) and (Lemma (splits are shorter): 2), we obtain:

(23) $\texttt{is-sorted-version}[\texttt{left-split}[\langle \overline{X_0} \rangle], \texttt{sorted}[\texttt{left-split}[\langle \overline{X_0} \rangle]]]$,

(24) $\texttt{is-sorted-version}[\texttt{right-split}[\langle \overline{X_0} \rangle], \texttt{sorted}[\texttt{right-split}[\langle \overline{X_0} \rangle]]]$,

From (23), by (Definition (is sorted version)), we obtain:

(25) $\texttt{is-tuple[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]]]} \bigwedge$.
$\texttt{left-split[}\langle \overline{X_0} \rangle\texttt{]} \approx \texttt{sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]]} \bigwedge$
$\texttt{is-sorted[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]]]}$

From (24), by (Definition (is sorted version)), we obtain:

(26) $\texttt{is-tuple[sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]} \bigwedge$.
$\texttt{right-split[}\langle \overline{X_0} \rangle\texttt{]} \approx \texttt{sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]} \bigwedge$
$\texttt{is-sorted[sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]}$

From (1) and (8), using (Definition (is sorted version)), is implied by:

(41) $\texttt{is-tuple[}$.
$\texttt{merged[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]], sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]]} \bigwedge$
$\texttt{merged[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]], sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]} \approx$
$\langle \overline{X_0} \rangle \bigwedge \texttt{is-sorted[}$
$\texttt{merged[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]], sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]]}$

We prove the individual conjunctive parts of (41):

Proof of (41.1)
$\texttt{is-tuple[}$:
$\texttt{merged[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]], sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]]}$

(41.1), by (Lemma (closure of merge)) is implied by:

(42) $\texttt{is-tuple[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]]]} \bigwedge$.
$\texttt{is-tuple[sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]}$

We prove the individual conjunctive parts of (42):

Proof of (42.1) $\texttt{is-tuple[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]]]}$:

Formula (42.1) is true because it is identical to (25.1).

Proof of (42.2) $\texttt{is-tuple[sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]}$:

Formula (42.2) is true because it is identical to (26.1).

Proof of (41.2)
$\texttt{merged[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]], sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]} \approx \langle \overline{X_0} \rangle$:

Formula (41.2), using (Lemma (conjecture44): conjecture44), is implied by:

(44) $\texttt{is-tuple[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]]]} \bigwedge$.
$\texttt{left-split[}\langle \overline{X_0} \rangle\texttt{]} \approx \texttt{sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]]} \bigwedge$
$\texttt{is-sorted[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]]]} \bigwedge$
$\texttt{is-tuple[sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]} \bigwedge$
$\texttt{right-split[}\langle \overline{X_0} \rangle\texttt{]} \approx \texttt{sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]} \bigwedge$
$\texttt{is-sorted[sorted[right-split[}\langle \overline{X_0} \rangle\texttt{]]]} \bigwedge \neg\, \texttt{is-trivial-tuple[}\langle \overline{X_0} \rangle\texttt{]}$

We prove the individual conjunctive parts of (44):

Proof of (44.1) $\texttt{is-tuple[sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]]]}$:

Formula (44.1) is true because it is identical to (25.1).

Proof of (44.2) $\texttt{left-split[}\langle \overline{X_0} \rangle\texttt{]} \approx \texttt{sorted[left-split[}\langle \overline{X_0} \rangle\texttt{]]}$:

Formula (44.2) is true because it is identical to (25.1).

Proof of (44.3) `is-sorted[sorted[left-split[⟨X0⟩]]]`:

Formula (44.3) is true because it is identical to (25.3).

Proof of (44.4) `is-tuple[sorted[right-split[⟨X0⟩]]]`:

Formula (44.4) is true because it is identical to (26.1).

Proof of (44.5) `right-split[⟨X0⟩] ≈ sorted[right-split[⟨X0⟩]]`:

Formula (44.5) is true because it is identical to (26.2).

Proof of (44.6) `is-sorted[sorted[right-split[⟨X0⟩]]]`:

Formula (44.6) is true because it is identical to (26.2).

Proof of (44.7) `¬ is-trivial-tuple[⟨X0⟩]`:

Formula (44.7) is true because it is identical to (6).

Proof of (41.3)
`is-sorted[ merged[sorted[left-split[⟨X0⟩]], sorted[right-split[⟨X0⟩]]]]`:

Formula (41.3), using (Lemma (conjecture46): conjecture46), is implied by:

(52) `is-tuple[sorted[left-split[⟨X0⟩]]] ⋀ left-split[⟨X0⟩] ≈ sorted[left-split[⟨X0⟩]] ⋀ is-sorted[sorted[left-split[⟨X0⟩]]] ⋀ is-tuple[sorted[right-split[⟨X0⟩]]] ⋀ right-split[⟨X0⟩] ≈ sorted[right-split[⟨X0⟩]] ⋀ is-sorted[sorted[right-split[⟨X0⟩]]] ⋀ ¬ is-trivial-tuple[⟨X0⟩]`.

We prove the individual conjunctive parts of (52):

Proof of (52.1) `is-tuple[sorted[left-split[⟨X0⟩]]]`:

Formula (52.1) is true because it is identical to (25.1).

Proof of (52.2) `left-split[⟨X0⟩] ≈ sorted[left-split[⟨X0⟩]]`:

Formula (52.2) is true because it is identical to (25..2).

Proof of (52.3) `is-sorted[sorted[left-split[⟨X0⟩]]]`:

Formula (52.3) is true because it is identical to (25.3).

Proof of (52.4) `is-tuple[sorted[right-split[⟨X0⟩]]]`:

Formula (52.4) is true because it is identical to (26.1).

Proof of (52.5) `right-split[⟨X0⟩] ≈ sorted[right-split[⟨X0⟩]]`:

Formula (52.5) is true because it is identical to (26.2).

Proof of (52.6) `is-sorted[sorted[right-split[⟨X0⟩]]]`:

Formula (52.6) is true because it is identical to (26.3).

Proof of (52.7) `¬ is-trivial-tuple[⟨X0⟩]`:

Formula (52.7) is true because it is identical to (6).

□