# Algorithm Synthesis by Lazy Thinking: Case Study Gröbner Bases

Bruno Buchberger
Research Institute for Symbolic Computation
Johannes Kepler University, Linz, Austria

Talk at Ultra-Omega-Theorema 2005, Saarland University
November 14, 2005



**Algorithm-Supported Mathematical Theory Exploration**

**An "Algorithm" for Algorithm Synthesis**

**Synthesis of a Gröbner Bases Algorithm**

**Conclusion**

## Algorithm-Supported Mathematical Theory Exploration

## An "Algorithm" for Algorithm Synthesis

## Synthesis of a Gröbner Bases Algorithm

## Conclusion



# Algorithm-Supported ("Automated") Mathematical Theory Exploration versus Automated Theorem Proving

Trento 1999, Talk by B.B.: Mathematical Theory Exploration # Theorem Proving

Starting from some given mathematical concepts and mathematical knowledge on these concepts within a uniform logical language (predicate logic),

- invent definitions (axioms) of new concepts

- invent and prove / disprove propositions on these concepts

- invent problems

- invent and verify algorithms for problems

- store and re-use the definitions, propositions, problems, algorithms in structured knowledge libraries

⟶ "MKM" (Mathematical Knowledge Management): 2001 RISC, 2003 Bertinoro, 2004 Bialowieza

5 of 41

## The Theorema Project

The *Theorema* project: Aims at computer-supported mathematical theory exploration.

The *Theorema* group: B. B. (leader), T. Jebelean, W. Windsteiger, T. Kutsia, F. Piroi, M. Rosenkranz, G. Regensburger, M. Giese, and PhD students.

Motivation: Algorithm-support for the exploration of "real" mathematics, e.g. Gröbner bases theory.

Paradigm: (Proved) proof generators for special areas of mathematics. (See B.B. talk at TYPES Workshop in Nijmegen, Nov. 3, 2004).

6 of 41

## Emphasis of this Talk

(Partially) automated invention of algorithms from problem specifications.

The method presented ("Lazy Thinking") can be implemented in any reasoning system, which provides:

- access to the proof objects even in the case of failing proofs (for the algorithmic extraction of "sub-requirments")

- a "natural" proof style (temporary assumptions and temporary goals)

7 of 41

## Algorithm-Supported Mathematical Theory Exploration

## An "Algorithm" for Algorithm Synthesis

## Synthesis of a Gröbner Bases Algorithm

## Conclusion

8 of 41

# The Algorithm Invention ("Synthesis") Problem

Given a problem specification P (in predicate logic), find an algorithm A such that

$$\underset{x}{\forall} P[x, A[x]].$$

Examples of specifications P:

```
P[x, y] ⇔ is-greater[x, y]
P[x, y] ⇔ is-sorted-version[x, y]
P[x, y] ⇔ has-derivative[x, y]
P[x, y] ⇔ are-factors-of[x, y]
P[x, y] ⇔ is-Gröbner-basis[x, y]
 ....
```

A general algorithm S for "all" P cannot exist but ...

9 of 41

## Literature

There is a rich literature on algorithm synthesis methods, see survey

[Basin et al. 2004] D. Basin, Y. Deville, P. Flener, A. Hamfelt, J. F. Nilsson. Synthesis of Programs in Computational Logic. In: M. Bruynooghe, K. K. Lau (eds.), Program Development in Computational Logic, Lecture Notes in Computer Science, Vol. 3049, Springer, 2004, pp. 30-65.

Our method is in the class of "scheme-based" methods. Closest (but essentially different):

[Lau et al. 1999] K. K. Lau, M. Ornaghi, S. Tärnlund. Steadfast logic programs. Journal of Logic Programming, 38/3, 1999, pp. 259-294.

Work in the Group of A. Bundy, Critics.

10 of 41

## The "Lazy Thinking" Method for Algorithm Synthesis (BB 2001): Intuition

"Lazy Thinking" Method for Algorithm Synthesis = My Advice to "Students" (= "Computers") How to Invent Algorithms

Given: A problem P.

Find: An algorithm A for P.

- Completely understand the problem. ("Specification" of the problem.)
- Learn how to prove.
- Collect (prove) "complete" knowledge on the auxiliary notion appearing in the problem.
- Consider known fundamental ideas of how to structure algorithms in terms of subalgorithms ("algorithm schemes").

  Try one scheme A after the other.
- Try to prove that A solves P: From the failing proof construct specifications of the subalgorithms.

# The "Lazy Thinking" Method for Algorithm Synthesis: Sketch

Given a problem specification P

- consider various "algorithm schemes" for A, e.g.
  $\mathbf{A}[\langle\rangle] = \mathbf{c}$
  $\underset{x}{\forall}\ \mathbf{A}[\langle x \rangle] = \mathbf{s}[\langle x \rangle]$
  $\underset{x,\bar{y}}{\forall}\ (\mathbf{A}[\langle x, y, \bar{z} \rangle] = \mathbf{i}[x, y, \mathbf{A}[\langle y, \bar{z} \rangle]])$
- and try to prove (automatically) $\underset{x}{\forall}\ \mathbf{P}[x, \mathbf{A}[x]]$.
- This proof will normally fail because nothing is known on the unspecified sub-algorithms c, s, i, ... in the algorithm scheme.
- From the temporary assumptions and goals in the failing proof situation (automatically) generate such specifications for the unspecified sub-algorithms c, s, i, ... that would make the proof possible.

Now, apply the method recursively to the auxiliary functions.

11 of 41

# Example: Synthesis of Merge-Sort [BB et al. 2003]

Problem: Synthesize "sorted" such that

$\underset{x}{\forall}\ \textbf{is-sorted-version}[x, \textbf{sorted}[x]].$

("Correctness Theorem")

Knowledge on Problem:

$\underset{x,y}{\forall}\ \left(\textbf{is-sorted-version}[x, y] \Leftrightarrow \begin{array}{l} \textbf{is-sorted}[y] \\ \textbf{is-permuted-version}[x, y] \end{array}\right)$

$\textbf{is-sorted}[\langle\rangle]$

$\underset{x}{\forall}\ \textbf{is-sorted}[\langle x \rangle]$

$$\underset{x,y,z}{\forall}\left(\text{is-sorted}[\langle x, y, z\rangle] \Leftrightarrow \begin{matrix} x \geq y \\ \text{is-sorted}[\langle y, z\rangle]\end{matrix}\right)$$

etc.



## An Algorithm Scheme: Divide and Conquer

$$\underset{x}{\forall}\left(\text{sorted}[x] = \begin{cases} s[x] & \Leftarrow \text{is-trivial-tuple}[x] \\ m[\text{sorted}[l[x]], \text{sorted}[r[x]]] & \Leftarrow \text{otherwise}\end{cases}\right)$$

s, m, l, r are unknowns.

We Now Start Proving the Correctness Theorem and Analyze the Failing Proof: see notebooks with failing proofs.



## Automated Invention of Sufficient Specifications for the Subalgorithms

A simple (but amazingly powerful) rule ( m ... an unknown subalgorithm ):

Collect temporary assumptions T[ x0, ... A [ ], ... ]

and temporary goals G[ x0, ...m [ A [ ] ] ]

and produces specification

$$\underset{X, \ldots, Y, \ldots}{\forall}\left(T[X, \ldots Y, \ldots] \Rightarrow G[Y, \ldots m[Y]]\right).$$

Details: see papers [BB 2003] and example.

# The Result of Applying Lazy Thinking in the Sorting Example

Lazy Thinking, automatically (in approx. 2 minutes on a laptop using the *Theorema* system), finds the following specifications for the sub-algorithms that provenly guarantee the correctness of the above algorithm (scheme):

$$\underset{x}{\forall}\ (\text{is-trivial-tuple}[x] \Rightarrow s[x] = x)$$

$$\underset{y,z}{\forall}\ \left(\begin{matrix}\text{is-sorted}[y] \\ \text{is-sorted}[z]\end{matrix} \Rightarrow \begin{matrix}\text{is-sorted}[m[y, z]] \\ m[y, z] \approx (y \asymp z)\end{matrix}\right)$$

$$\underset{x}{\forall}\ (l[x] \asymp r[x] \approx x)$$

Note: the specifications generated are not only sufficient but natural !

The four proof notebooks generated automatically by Theorema that develop these specifications successively, are given in the appendix.



# What Do We Have Now?

- **Case A:** We find algorithms S, M, L, R in our knowledge base for which the properties specified above for s, m, l, r are already contained in the knowledge base or can be derived (proved) from the knowledge base.

  In this case, we are done, i.e. we have synthesized a sorting algorithm.

- **Case B:** We do not find such algorithms S, M, L, R in our knowledge base.

  In this case, we apply Lazy Thinking again in order to synthesize appropriate s, m, l, r

  until we arrive at sub-sub-...-algorithms in our knowledge base (e.g. the basic operations of tuple theory like append, prepend etc.)

Case B can be avoided, if we proceed systematically bottom-up ("complete theory exploration" in layers).

## Example: Synthesis of Insertion-Sort

Synthesize A such that

$$\underset{x}{\forall}\ \textbf{is-sorted-version}[x, A[x]].$$

Algorithm Scheme: "simple recursion"

$$A[\langle\rangle] = c$$
$$\underset{x}{\forall}\ A[\langle x\rangle] = s[\langle x\rangle]$$
$$\underset{x,\bar{y}}{\forall}\ (A[\langle x, \bar{y}\rangle] = i[x, A[\langle\bar{y}\rangle]])$$

Lazy Thinking, automatically (in approx. 2 minutes on a laptop using the *Theorema* system), finds the following specifications for the auxiliary functions

$$c = \langle\rangle$$
$$\underset{x}{\forall}\ (s[\langle x\rangle] = \langle x\rangle)$$
$$\underset{x,\bar{y}}{\forall}\ \left(\textbf{is-sorted}[\langle\bar{y}\rangle] \Rightarrow \begin{array}{l}\textbf{is-sorted}[i[x, \langle\bar{y}\rangle]]\\ i[\langle x, \bar{y}\rangle] \approx (x \smile \langle\bar{y}\rangle)\end{array}\right)$$

17 of 41

## How Far Can We Go With the Method ?

Can we automatically synthesize algorithms for non-trivial problems? What is "non-trivial"?

Example of a non-trivial problem: construction of Gröbner bases.

18 of 41

## The Problem of Constructing Gröbner Bases is Non-trivial

Dozens of fundamental problems in algebraic geometry, invariant theory, optimization, graph theory, coding theory, cryptography, statistics, symbolic summation, symbolic solution of differential equations, ... can be reduced to the construction of Gröbner bases. (Approx. 500 papers on the application of the Gröbner bases method.)

Some of these problems were open for decades.

Main algorithmic idea of Gröbner bases theory: The "S-polynomials" together with the S-polynomial theorem.

(B.B. An Algorithmical Criterion for the Solvability of Algebraic Systems of Equations. Aequationes mathematicae 4/3, 1970.)

Hence, question: Can Lazy Thinking automatically invent the notion of S-polynomial and automatically deliver the S-polynomial theorem with proof?

19 of 41

## The Gröbner Bases Algorithm in *Mathematica*, Maple, Derive, ...

```
f1 = x y - 2 y z - z - 1;
f2 = y^2 - x^2 z + x z + 2;
f3 = z^2 - y^2 x + x + 1;

F = {f1, f2, f3};
```

```
{time, G} = GroebnerBasis[F] // Timing
```

```
{0. Second,
 {-11 - 23 z - 14 z^2 + 177 z^3 + 134 z^4 + 230 z^5 - 383 z^6 - 11 z^7 - 744 z^8 + 5 z^9 -
   348 z^10 + 176 z^11 - 64 z^12 + 64 z^13, 8370016703419 - 97318774930576 y +
   207351025151780 z - 722400840178486 z^2 + 155529319830493 z^3 -
   1676870964046611 z^4 + 2617008001482037 z^5 - 1102382418002270 z^6 +
   3982342741210737 z^7 - 760100969952625 z^8 + 2005641117585932 z^9 -
   1006208474754864 z^10 + 397148351653504 z^11 - 304335002718272 z^12,
  -7357801284994 - 12164846866322 x - 24858058005413 z -
   54253437844244 z^2 - 25982455463515 z^3 + 137954693662980 z^4 +
   82182910700209 z^5 + 274904331275643 z^6 +
   68396806233557 z^7 + 121808189089485 z^8 - 38565643382556 z^9 +
   8317571394992 z^10 - 19345137946880 z^11 - 4858008691392 z^12}}
```

20 of 41

# The S-Polynomials

```
S-polynomial[x y - 2 y z - z - 1, y^2 - x^2 z + x z + 2] =
 x z (x y - 2 y z - z - 1) + y (y^2 - x^2 z + x z + 2)
```

$y (2 + y^2 + x z - x^2 z) + x z (-1 + x y - z - 2 y z)$

```
x z (x y - 2 y z - z - 1) + y (y^2 - x^2 z + x z + 2) // Expand
```

$2 y + y^3 - x z + x y z - x z^2 - 2 x y z^2$

```
S-polynomial[g1, g2] =
 form the

least-common-multiple[leading-power-product[g1], leading-power-product[g2]]

 and then ...
```

21 of 41

Algorithm-Supported Mathematical Theory Exploration

An "Algorithm" for Algorithm Synthesis

**Synthesis of a Gröbner Bases Algorithm**

Conclusion

22 of 41

## The Problem of Constructing Gröbner Bases

Find algorithm Gb such that

$$\underset{\text{is-finite}[F]}{\forall} \begin{pmatrix} \text{is-finite}[\text{Gb}[F]] \\ \text{is-Gröbner-basis}[\text{Gb}[F]] \\ \text{ideal}[F] = \text{ideal}[\text{Gb}[F]]. \end{pmatrix}$$

Definitions [BB 1965] :

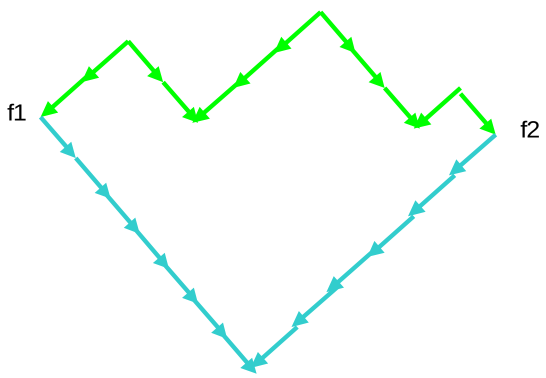
**is-Gröbner-basis[G] ⇔ is-confluent[ $\rightarrow_G$ ].**

$\rightarrow_G$ ... a division step.

23 of XXX

# Confluence of Division $\rightarrow_G$

**is-confluent[ → ] :⇔** $\forall_{f1,f2}$ **(f1 ↔* f2 ⇒ f1↓* f2)**




24 of 41

# Knowledge on the Concepts Involved

**h1** $\rightarrow_G$ **h2 ⇒ p . h1** $\rightarrow_G$ **p . h2**

etc.

25 of 41

# Algorithm Scheme "Critical Pair / Completion"

```
A[F] = A[F, pairs[F]]
A[F, ⟨⟩] = F

A[F, ⟨⟨g1, g2⟩, p̄⟩] =
 where[f = lc[g1, g2], h1 = trd[rd[f, g1], F], h2 = trd[rd[f, g2], F],
   { A[F, ⟨p̄⟩]                                              ⇐ h1 = h2
   { A[F ⌢ df[h1, h2], ⟨p̄⟩ ≍ ⟨⟨F_k, df[h1, h2]⟩ | k=1,…,|F| ⟩]  ⇐ otherwise ]
```

This scheme can be tried in any domain, in which we have a reduction operation rd that depends on sets F of objects and a Noetherian relation $\succ$ which interacts with rd in the following natural way:

$$\underset{f,g}{\forall}\ (f \succeq rd[f, g]).$$



# The Essential Problem

The problem of synthesizing a Gröbner bases algorithm can now be also stated by asking whether, starting with the proof of

$$\underset{F}{\forall}\ \begin{pmatrix} \text{is-finite}[A[F]] \\ \text{is-Gröbner-basis}[A[F]] \\ \text{ideal}[F] = \text{ideal}[A[F]] \end{pmatrix},$$

we can *automatically* produce the idea that

```
lc[g1, g2] = lcm[lp[g1], lp[g2]]
```

and

```
df[h1, h2] = h1 - h2
```

and prove that the idea is correct.

## Now Start the (Automated) Correctness Proof

With current theorem proving technology, in the *Theorema* system (and other provers), the proof attempt could be done automatically. (Ongoing PhD thesis of A. Craciun.)

28 of 41

## Details

It should be clear that, if the algorithm terminates, the final result is a finite set (of polynomials) G that has the property

$$\underset{g1,g2\in G}{\forall}\left(\text{where}\left[f = \text{lc}[g1, g2],\ h1 = \text{trd}[\text{rd}[f, g1], F],\ h2 = \text{trd}[\text{rd}[f, g2], F],\ \bigvee\begin{cases} h1 = h2 \\ \text{df}[h1, h2] \in G \end{cases}\right]\right).$$

We now try to prove that, if G has this property, then

```
is-finite[G],
ideal[F] = ideal[G],
is-Gröbner-basis[G],
   i.e. is-Church-Rosser[ →G ].
```

Here, we only deal with the third, most important, property.

29 of 41

## Using Available Knowledge

Using Newman's lemma and some elementary properties it can be shown that it is sufficient to prove

$$\text{is-Church-Rosser}[\to_G] \Leftrightarrow \underset{p}{\forall}\ \underset{f1,f2}{\forall}\left(\left(\begin{cases} p \to f1 \\ p \to f2 \end{cases}\right) \Rightarrow f1 \downarrow^* f2\right).$$

30 of 41

# The (Automated) Proof Attempt

Let now the power product p and the polynomials f1, f2 be arbitary but fixed and assume

$$\begin{cases} p \to_G f1 \\ p \to_G f2. \end{cases}$$

We have to find a polyonomial g such that

$$f1 \to_G^* g,$$
$$f2 \to_G^* g.$$

From the assumption we know that there exist polynomials g1 and g2 in G such that

```
lp[g1] | p,
f1 = rd[p, g1],
lp[g2] | p,
f2 = rd[p, g2].
```

From the final situation in the algorithm scheme we know that for these g1 and g2

$$\bigvee \begin{cases} h1 = h2 \\ df[h1, h2] \in G, \end{cases}$$

where

```
h1 := trd[f1', G], f1' := rd[lc[g1, g2], g1],
h2 := trd[f2', G], f2' := rd[lc[g1, g2], g2].
```



# Case h1=h2

$$lc[g1, g2] \to_{g1} rd[lc[g1, g2], g1] \to_G^* trd[rd[lc[g1, g2], g1], G] =$$
$$trd[rd[lc[g1, g2], g2], G] \leftarrow_G^* rd[lc[g1, g2], g2] \leftarrow_{g2} lc[g1, g2].$$

(Note that here we used the requirements rd[lc[g1,g2],g1]$\prec$lc[g1,g2] and rd[lc[g1,g2],g2]$\prec$lc[g1,g2].)

Hence, by elementary properties of polynomial reduction,

$$\underset{a,q}{\forall}\ (a\,q\,lc[g1, g2] \to_{g1} a\,q\,rd[lc[g1, g2], g1] \to_G^* a\,q\,trd[rd[lc[g1, g2], g1], G] = a\,q\,trd[rd[lc[g1, g2], g2], G] \leftarrow_G^* a\,q\,rd[lc[g1, g2], g2] \leftarrow_{g2} a\,q\,lc[g1, g2]).$$

Now we are stuck in the proof.

32 of 41

## Now Use the Specification Generation Algorithm

Using the above specification generation rule, we see that we could proceed successfully with the proof if lc[g1,g2] satisfied the following requirement

$$\underset{p,g1,g2}{\forall}\left(\left(\begin{cases} lp[g1] \mid p \\ lp[g2] \mid p \end{cases}\right) \Rightarrow \left(\underset{a,q}{\exists}\ (p = a\,q\,lc[g1, g2])\right)\right), \quad (\text{lc requirement})$$

With such an lc, we then would have

$$p \to_{g1} rd[p, g1] = a\,q\,rd[lc[g1, g2], g1] \to_G^* a\,q\,trd[rd[lc[g1, g2], g1], G] = a\,q\,trd[rd[lc[g1, g2], g2], G] \leftarrow_G^* a\,q\,rd[lc[g1, g2], g2] = rd[p, g2] \leftarrow_{g2} p$$

and, hence,

$$f1 \to_G^* a\,q\,trd[rd[lc[g1, g2], g1], G],$$

$$f2 \to_G^* a\,q\,trd[rd[lc[g1, g2], g1], G],$$

i.e. we would have found a suitable g.

33 of 41

## Summarize the (Automatically Generated) Specifications of the Subalgorithm lc

(lc requirement), which also could be written in the form:

$$\underset{p,g1,g2}{\forall}\left(\left(\begin{cases} lp[g1] \mid p \\ lp[g2] \mid p \end{cases}\right) \Rightarrow (lc[g1, g2] \mid p)\right),$$

and

```
lp[g1] | lc[g1, g2],
lp[g2] | lc[g1, g2],
```

wich is a consequence of

```
rd[lc[g1, g2], g1] < lc[g1, g2],
rd[lc[g1, g2], g2] < lc[g1, g2].
```

## Summarize Again

For synthesizing an algorithm for the Gröbner bases problem it suffices to find an lc satisfying

$$\underset{p,g1,g2}{\forall}\left(\left(\begin{cases}lp[g1] \mid p\\ lp[g2] \mid p\end{cases}\right) \Rightarrow (lc[g1, g2] \mid p)\right),$$

and

```
lp[g1] | lc[g1, g2],
lp[g2] | lc[g1, g2].
```

This problem can be solved by any high-school student (or university professor)! No knowledge on Gröbner bases theory necessary!

34 of 41

## A Suitable lc

```
lcp[g1, g2] = lcm[lp[g1], lp[g2]]
```

is a suitable function that satisfies the above requirements.

Eureka! The crucial function lc (the "critical pair" function) in the critical pair / completion algorithm scheme has been synthesized automatically!

35 of 41

## Case h1≠h2

In this case, df[h1,h2]∈G:

In this part of the proof (wich is much easier) we are basically stuck right at the beginning. By the requirement generation algorithm we obtain the following requirement for df:

$\underset{h1,h2}{\forall} (h1 \downarrow_{\{df[h1,h2]\}}{}^{*} h2)$ (df requirement).

36 of 41

## Looking to the Knowledge Base for a Suitable df

(Looking to the knowledge base of elementary properties of polynomial reduction, it is now easy to find a function df that satifies (df requirement), namely

df[h1, h2] = h1 - h2,

because, in fact,

$\underset{f,g}{\forall} (f \downarrow_{\{f-g\}}{}^{*} g)$.

Eureka! The function df (the "completion" function) in the critical pair / completion algorithm scheme has been "automatically" synthesized!)



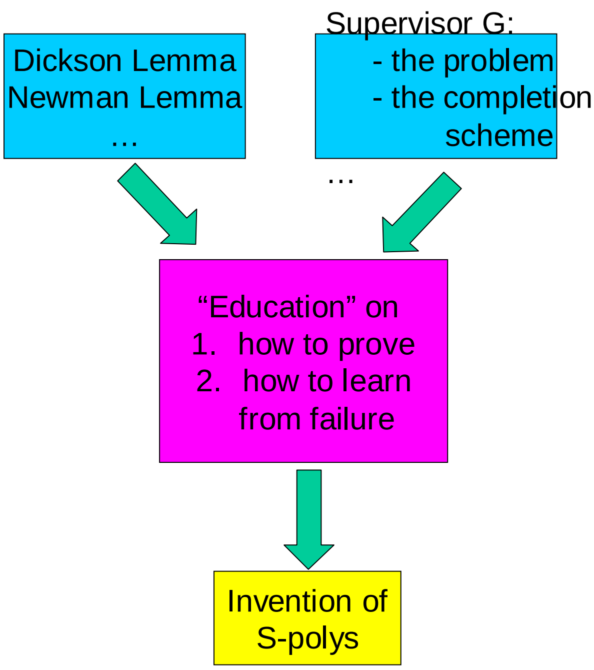
## Algorithm-Supported Mathematical Theory Exploration

## An "Algorithm" for Algorithm Synthesis

## Synthesis of a Gröbner Bases Algorithm

## Conclusion

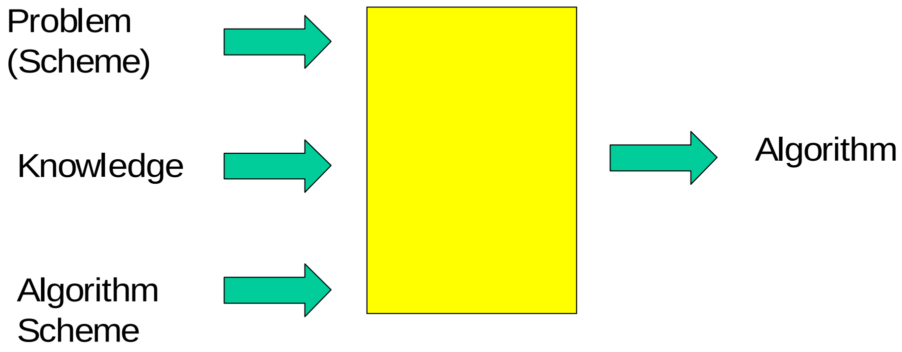
## A way of looking at it ("what would have happened if ..."):




Pairs idea? (The CPC algorithm scheme did not really exist at that time.)

```
A[F, ⟨⟨g1, g2⟩, p⟩] =
 where[f = lc[g1, g2], h1 = trd[rd[f, g1], F], h2 = trd[rd[f, g2], F],
   { A[F, ⟨p⟩]                                                              ⇐ h1 = h2
   { A[F ⌢ df[h1, h2], ⟨p⟩ ≍ ⟨⟨F_k, df[h1, h2]⟩ | k=1,…,|F| ⟩]             ⇐ otherwise   ]
```

## Research Topics




- **Libraries** of algorithm **schemes**.

  More generally, libraries of formulae schemes for definitions, propositions, problems, and algorithms.

- **Case studies** of problem (schemes), knowledge, algorithm schemes and how they produce algorithms.

- How well are **current reasoning systems suited** for supporting this approach to algorithm (definition, theorem, problem, ...) synthesis?

- Improved **algorithms for generating problem specifications** from failing proofs.

40 of 41

## Special Semester on Gröbner Bases at RICAM and RISC (Feb - July 2006)

8 Workshops on various special topics in GB theory:  One workshop on

"Formal Gröbner Bases Theory",  March 6 - March 10, 2006.

41 of 41

# References

## ■ On Gröbner Bases

[Buchberger 1970]

B. Buchberger. Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems (An Algorithmical Criterion for the Solvability of Algebraic Systems of Equations). Aequationes mathematicae 4/3, 1970, pp. 374-383. (English translation in: [Buchberger, Winkler 1998], pp. 535 -545.) Published version of the PhD Thesis of B. Buchberger, University of Innsbruck, Austria, 1965.

[Buchberger 1998]

B. Buchberger. Introduction to Gröbner Bases. In: [Buchberger, Winkler 1998], pp.3-31.

[Buchberger, Winkler, 1998]

B. Buchberger, F. Winkler (eds.). Gröbner Bases and Applications, Proceedings of the International Conference "33 Years of Gröbner Bases", 1998, RISC, Austria, London Mathematical Society Lecture Note Series, Vol. 251, Cambridge University Press, 1998.

[Becker, Weispfenning 1993]

T. Becker, V. Weispfenning. Gröbner Bases: A Computational Approach to Commutative Algebra, Springer, New York, 1993.

## ■ On Mathematical Knowledge Management

B. Buchberger, G. Gonnet, M. Hazewinkel (eds.)

Mathematical Knowledge Management.

Special Issue of Annals of Mathematics and Artificial Intelligence, Vol. 38, No. 1-3, May 2003, Kluwer Academic Publisher, 232 pages.

A.Asperti, B. Buchberger, J.H.Davenport (eds.)

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-NewYork, 2003, 223 pages.

A.Asperti, G.Bancerek, A.Trybulec (eds.).

Proceedings of the Third International Conference on Mathematical Knowledge Management, MKM 2004,

Bialowieza, Poland, September 19-21, 2004, Lecture Notes in Computer Science, Vol. 3119, Springer, Berlin-Heidelberg-NewYork, 2004

## ■ On Theorema

[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.

## ■ On Theory Exploration and Algorithm Synthesis

[Buchberger 2000]

B. Buchberger. Theory Exploration with *Theorema*.

Analele Universitatii Din Timisoara, Ser. 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, Rumania, T. Jebelean, V. Negru, A. Popovici eds.), ISSN 1124-970X, 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 (Symbolic and

Numeric Algorithms for Scientific Computing, Timisoara, Romania, October 1–4, 2003), Mirton Publishing, ISBN 973–661–104–3, pp. 2–26.

[Buchberger, Craciun 2003]

B. Buchberger, A. Craciun. Algorithm Synthesis by Lazy Thinking: Examples and Implementation in Theorema. in: Fairouz Kamareddine (ed.), Proc. of the Mathematical Knowledge Management Workshop, Edinburgh, Nov. 25, 2003, Electronic Notes on Theoretical Computer Science, volume dedicated to the MKM 03 Symposium, Elsevier, ISBN 044451290X, to appear.

[Buchberger 2005]

B. Buchberger.

Towards the Automated Synthesis of a Gröbner Bases Algorithm.

RACSAM (Review of the Royal Spanish Academy of Science), Vol. 98/1, pp. 65-75, 2005.

# Appendix: The Proof Notebooks Automatically Generated by Theorema for the Synthesis of the Merge-Sort Algorithm

## Notebook 1:

## Notebook 2

Comment on Notebook 2: Note that in the knowledge base (i.e. the formulae listed under "assumptions"), the specification (Lemma (conjecture15): conjecture15) is now contained, which describes the specification automatically generated (from the failing proof in Notebook 1) for the function 'special'. The proof then proceeds as in Notebook 1 but succeeds to get over the point at which the first proof was stuck.

Prove:

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

under the assumptions:

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

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

(Definition (is sorted): 3) $\underset{x,y,z}{\forall}$ (is-sorted[⟨x, y, z̄⟩] ⇔ x ≥ y ∧ is-sorted[⟨y, z̄⟩]),

.... and all the formulae in the assumptions of Notebook 1,

(Lemma (closure of merge)) $\underset{\substack{\text{is-tuple}[X]\\ \text{is-tuple}[Y]}}{\forall}$ is-tuple[merged[X, Y]], (Lemma (conjecture15): conjecture15) $\underset{\text{is-tuple}[X1]}{\underset{X1}{\forall}}$ (is-trivial-tuple[$X1$] ⇒ (special[$X1$] = $X1$)).

We try to prove (Theorem (correctness of sort)) by applying several proof methods for sequences.

We try to prove (Theorem (correctness of sort)) by well–founded induction on $X$.

Well–founded induction:

Assume:

(1) is-tuple[⟨$\overline{X_0}$⟩].

Well–Founded Induction Hypothesis:

(2) $\underset{\text{is-tuple}[x2]}{\forall}$ (⟨$\overline{X_0}$⟩ > $x2$ ⇒ is-sorted-version[$x2$, sorted[$x2$]])

We have to show:

(3) is-sorted-version[⟨$\overline{X_0}$⟩, sorted[⟨$\overline{X_0}$⟩]].

We try to prove (3) by case distinction using (Algorithm (sorted)). However, the proof fails in at least one of the cases.

Case 1:

(4) is-trivial-tuple[⟨$\overline{X_0}$⟩].

Hence, we have to prove

(5) is-sorted-version[⟨$\overline{X_0}$⟩, special[⟨$\overline{X_0}$⟩]].

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

(9) is-sorted[⟨$\overline{X_0}$⟩].

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

(10) $\underset{Y}{\forall}$ (($Y$ = ⟨$\overline{X_0}$⟩) ⇒ $Y$ ≈ ⟨$\overline{X_0}$⟩).

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

(11) is-tuple[special[⟨$\overline{X_0}$⟩]].

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

(13) special[⟨$\overline{X_0}$⟩] = ⟨$\overline{X_0}$⟩.

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

(21) is-sorted-version[⟨$\overline{X_0}$⟩, ⟨$\overline{X_0}$⟩].

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

(22) is-tuple[⟨$\overline{X_0}$⟩] ⋀ ⟨$\overline{X_0}$⟩ ≈ ⟨$\overline{X_0}$⟩ ⋀ is-sorted[⟨$\overline{X_0}$⟩].

We prove the individual conjunctive parts of (22):

Proof of (22.1) is-tuple[⟨$\overline{X_0}$⟩]:

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

Proof of (22.2) ⟨$\overline{X_0}$⟩ ≈ ⟨$\overline{X_0}$⟩:

Formula (22.2) is true by (10).

Proof of (22.3) is-sorted[⟨$\overline{X_0}$⟩]:

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

Case 2:

(6) ¬ is-trivial-tuple[⟨$X_0$⟩].

Hence, we have to prove

(8) is-sorted-version[⟨$X_0$⟩,
merged[sorted[left-split[⟨$X_0$⟩]], sorted[right-split[⟨$X_0$⟩]]]] .

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) is-sorted-version[left-split[⟨$X_0$⟩], sorted[left-split[⟨$X_0$⟩]]],

(24) is-sorted-version[right-split[⟨$X_0$⟩], sorted[right-split[⟨$X_0$⟩]]],

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

(25)

is-tuple[sorted[left-split[⟨$X_0$⟩]]] ∧
left-split[⟨$X_0$⟩] ≈ sorted[left-split[⟨$X_0$⟩]] ∧ is-sorted[sorted[left-split[⟨$X_0$⟩]]]
.

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

(26) is-tuple[sorted[right-split[⟨$X_0$⟩]]] ∧ .
right-split[⟨$X_0$⟩] ≈ sorted[right-split[⟨$X_0$⟩]] ∧
is-sorted[sorted[right-split[⟨$X_0$⟩]]]

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

(41) is-tuple[merged[sorted[left-split[⟨$X_0$⟩]], sorted[right-split[⟨$X_0$⟩]]]] ∧ .
merged[sorted[left-split[⟨$X_0$⟩]], sorted[right-split[⟨$X_0$⟩]]] ≈ ⟨$X_0$⟩ ∧
is-sorted[merged[sorted[left-split[⟨$X_0$⟩]], sorted[right-split[⟨$X_0$⟩]]]]

Not all the conjunctive parts of (41)can be proved.

Proof of (41.1) is-tuple[merged[sorted[left-split[⟨$X_0$⟩]], sorted[right-split[⟨$X_0$⟩]]]]:

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

(42) is-tuple[sorted[left-split[⟨$X_0$⟩]]] ∧ is-tuple[sorted[right-split[⟨$X_0$⟩]]].

We prove the individual conjunctive parts of (42):

Proof of (42.1) is-tuple[sorted[left-split[⟨$X_0$⟩]]]:

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

Proof of (42.2) is-tuple[sorted[right-split[⟨$X_0$⟩]]]:

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

Proof of (41.3) merged[sorted[left-split[⟨$X_0$⟩]], sorted[right-split[⟨$X_0$⟩]]] ≈ ⟨$X_0$⟩:

The proof of (41.3)fails. (The prover "QR" was unable to transform the proof situation.)

Proof of (41.4)
is-sorted[merged[sorted[left-split[⟨$X_0$⟩]], sorted[right-split[⟨$X_0$⟩]]]]:

Pending proof of (41.4).

□

## Notebook 3

Comment on Notebook 3: Note that in the knowledge base (i.e. the formulae listed under "assumptions"), the additional specification (Lemma (conjecture44): conjecture44) is now contained, which describes part of the specification automatically generated (from the failing proof in Notebook 2) for the functions 'left', 'right', and 'merge'. The proof then proceeds as in Notebook 2 but succeeds to get over the point at which the second proof was stuck.

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

... and all the assumptions of Notebook 1 ...

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

(Lemma (conjecture15): conjecture15)

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

(Lemma (conjecture44): conjecture44)

$\forall_{\substack{X2,X3,X4\\ \text{is-tuple}[X4]}}$ (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[$X2$, $X3$] ≈ $X4$) .

We try to prove (Theorem (correctness of sort)) by well–founded induction on $X$.

Well–founded induction:

Assume:

(1) is-tuple[⟨$\overline{X_0}$⟩].

Well–Founded Induction Hypothesis:

(2) $\forall_{\text{is-tuple}[x3]}$ (⟨$\overline{X_0}$⟩ ≻ $x3$ ⇒ is-sorted-version[$x3$, sorted[$x3$]])

We have to show:

(3) is-sorted-version[⟨$\overline{X_0}$⟩, sorted[⟨$\overline{X_0}$⟩]].

We try to prove (3) by case distinction using (Algorithm (sorted)). However, the proof fails in at least one of the cases.

Case 1:

(4) is-trivial-tuple[⟨$\overline{X_0}$⟩].

Hence, we have to prove

(5) is-sorted-version[⟨$\overline{X_0}$⟩, special[⟨$\overline{X_0}$⟩]].

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

(9) is-sorted[⟨$\overline{X_0}$⟩].

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

(10) $\forall_{Y}$ ((Y = ⟨$\overline{X_0}$⟩) ⇒ Y ≈ ⟨$\overline{X_0}$⟩).

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

(11) is-tuple[special[⟨$\overline{X_0}$⟩]].

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

(13) special[⟨$\overline{X_0}$⟩] = ⟨$\overline{X_0}$⟩.

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

(21) is-sorted-version[⟨$\overline{X_0}$⟩, ⟨$\overline{X_0}$⟩].

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

(22) is-tuple[⟨$\overline{X_0}$⟩] ⋀ ⟨$\overline{X_0}$⟩ ≈ ⟨$\overline{X_0}$⟩ ⋀ is-sorted[⟨$\overline{X_0}$⟩].

We prove the individual conjunctive parts of (22):

Proof of (22.1) is-tuple[⟨$\overline{X_0}$⟩]:

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

Proof of (22.2) ⟨$\overline{X_0}$⟩ ≈ ⟨$\overline{X_0}$⟩:

Formula (22.2) is true by (10).

Proof of (22.3) is-sorted[⟨$\overline{X_0}$⟩]:

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

Case 2:

(6) ¬ is-trivial-tuple[⟨$\overline{X_0}$⟩].

Hence, we have to prove

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

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) is-sorted-version[left-split[⟨$\overline{X_0}$⟩], sorted[left-split[⟨$\overline{X_0}$⟩]]],

(24) is-sorted-version[right-split[⟨$\overline{X_0}$⟩], sorted[right-split[⟨$\overline{X_0}$⟩]]],

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

(25)

is-tuple[sorted[left-split[$\langle \overline{X_0} \rangle$]]] ∧
left-split[$\langle \overline{X_0} \rangle$] ≈ sorted[left-split[$\langle \overline{X_0} \rangle$]] ∧ is-sorted[sorted[left-split[$\langle \overline{X_0} \rangle$]]]
.

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

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

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

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

Not all the conjunctive parts of (41)can be proved.

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

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

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

We prove the individual conjunctive parts of (42):

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

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

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

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

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

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

(44)

is-tuple[sorted[left-split[$\langle \overline{X_0} \rangle$]]] ∧ left-split[$\langle \overline{X_0} \rangle$] ≈ sorted[left-split[$\langle \overline{X_0} \rangle$]] ∧.
is-sorted[sorted[left-split[$\langle \overline{X_0} \rangle$]]] ∧ is-tuple[sorted[right-split[$\langle \overline{X_0} \rangle$]]] ∧
right-split[$\langle \overline{X_0} \rangle$] ≈ sorted[right-split[$\langle \overline{X_0} \rangle$]] ∧
is-sorted[sorted[right-split[$\langle \overline{X_0} \rangle$]]] ∧ ¬ is-trivial-tuple[$\langle \overline{X_0} \rangle$]

We prove the individual conjunctive parts of (44):

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

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

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

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

Proof of (44.3) is-sorted[sorted[left-split[$\langle \overline{X_0} \rangle$]]]:

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

Proof of (44.4) is-tuple[sorted[right-split[$\langle \overline{X_0} \rangle$]]]:

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

Proof of (44.5) right-split[$\langle \overline{X_0} \rangle$] ≈ sorted[right-split[$\langle \overline{X_0} \rangle$]]:

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

Proof of (44.6) is-sorted[sorted[right-split[$\langle \overline{X_0} \rangle$]]]:

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

Proof of (44.7) ¬ is-trivial-tuple[$\langle \overline{X_0} \rangle$]:

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

Proof of (41.3)
is-sorted[merged[sorted[left-split[$\langle \overline{X_0} \rangle$]], sorted[right-split[$\langle \overline{X_0} \rangle$]]]]:

The proof of (41.3) fails. (The prover "QR" was unable to transform the proof situation.)

□

## Notebook 4

Comment on Notebook 4: Note that in the knowledge base (i.e. the formulae listed under "assumptions"), the additional specification (Lemma (conjecture46): conjecture46) is now contained, which describes the second part of the specification automatically generated (from the failing proof in Notebook 3) for the functions 'left', 'right', and 'merge'. The proof then proceeds as in Notebook 3 but succeeds to get over the point at which the third proof was stuck and, actually, proceeds until the successful end.

Prove:

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

under the assumptions:

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

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

(Definition (is sorted): 3) $\underset{x,y,\bar{z}}{\forall}$ (is-sorted[⟨$x$, $y$, $\bar{z}$⟩] ⇔ $x \geq y$ ∧ is-sorted[⟨$y$, $\bar{z}$⟩]),

.... and all the assumptions appearing in Notebook 1 ...

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

(Lemma (conjecture15): conjecture15)

$\underset{\substack{X1 \\ \text{is-tuple}[X1]}}{\forall}$ (is-trivial-tuple[$X1$] ∧ is-sorted[$X1$] ⇒ (special[$X1$] = $X1$)), (Lemma (conjecture44): conjecture44)

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

(Lemma (conjecture46): conjecture46)

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

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) $\underset{\text{is-tuple}[x4]}{\forall} (\langle \overline{X_0} \rangle \succ x4 \Rightarrow \text{is-sorted-version}[x4, \text{sorted}[x4]])$

We have to show:

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

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

Case 1:

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

Hence, we have to prove

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

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

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

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

(10) $\underset{Y}{\forall} ((Y = \langle \overline{X_0} \rangle) \Rightarrow Y \approx \langle \overline{X_0} \rangle)$.

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

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

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

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

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

(21)is-sorted-version[⟨$\overline{X_0}$⟩, ⟨$\overline{X_0}$⟩].

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

(22)is-tuple[⟨$\overline{X_0}$⟩] ∧ ⟨$\overline{X_0}$⟩ ≈ ⟨$\overline{X_0}$⟩ ∧ is-sorted[⟨$\overline{X_0}$⟩].

We prove the individual conjunctive parts of (22):

Proof of (22.1) is-tuple[⟨$\overline{X_0}$⟩]:

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

Proof of (22.2) ⟨$\overline{X_0}$⟩ ≈ ⟨$\overline{X_0}$⟩:

Formula (22.2) is true by (10).

Proof of (22.3) is-sorted[⟨$\overline{X_0}$⟩]:

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

Case 2:

(6)¬ is-trivial-tuple[⟨$\overline{X_0}$⟩].

Hence, we have to prove

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

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)is-sorted-version[left-split[⟨$\overline{X_0}$⟩], sorted[left-split[⟨$\overline{X_0}$⟩]]],

(24)is-sorted-version[right-split[⟨$\overline{X_0}$⟩], sorted[right-split[⟨$\overline{X_0}$⟩]]],

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

(25)

is-tuple[sorted[left-split[⟨$\overline{X_0}$⟩]]] ∧
left-split[⟨$\overline{X_0}$⟩] ≈ sorted[left-split[⟨$\overline{X_0}$⟩]] ∧ is-sorted[sorted[left-split[⟨$\overline{X_0}$⟩]]]
.

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

(26)is-tuple[sorted[right-split[⟨$\overline{X_0}$⟩]]] ∧ .
right-split[⟨$\overline{X_0}$⟩] ≈ sorted[right-split[⟨$\overline{X_0}$⟩]] ∧
is-sorted[sorted[right-split[⟨$\overline{X_0}$⟩]]]

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

(41)is-tuple[merged[sorted[left-split[⟨$\overline{X_0}$⟩]], sorted[right-split[⟨$\overline{X_0}$⟩]]]] ∧ .
merged[sorted[left-split[⟨$\overline{X_0}$⟩]], sorted[right-split[⟨$\overline{X_0}$⟩]]] ≈ ⟨$\overline{X_0}$⟩ ∧
is-sorted[merged[sorted[left-split[⟨$\overline{X_0}$⟩]], sorted[right-split[⟨$\overline{X_0}$⟩]]]]

We prove the individual conjunctive parts of (41):

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

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

(42)is-tuple[sorted[left-split[⟨$\overline{X_0}$⟩]]] ∧ is-tuple[sorted[right-split[⟨$\overline{X_0}$⟩]]].

We prove the individual conjunctive parts of (42):

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

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

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

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

Proof of (41.2) merged[sorted[left-split[⟨$\overline{X_0}$⟩]], sorted[right-split[⟨$\overline{X_0}$⟩]]] ≈ ⟨$\overline{X_0}$⟩:

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

(44)

is-tuple[sorted[left-split[⟨$\overline{X_0}$⟩]]] ⋀ left-split[⟨$\overline{X_0}$⟩] ≈ sorted[left-split[⟨$\overline{X_0}$⟩]] ⋀.
is-sorted[sorted[left-split[⟨$\overline{X_0}$⟩]]] ⋀ is-tuple[sorted[right-split[⟨$\overline{X_0}$⟩]]] ⋀
right-split[⟨$\overline{X_0}$⟩] ≈ sorted[right-split[⟨$\overline{X_0}$⟩]] ⋀
is-sorted[sorted[right-split[⟨$\overline{X_0}$⟩]]] ⋀ ¬ is-trivial-tuple[⟨$\overline{X_0}$⟩]

We prove the individual conjunctive parts of (44):

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

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

Proof of (44.2) left-split[⟨$\overline{X_0}$⟩] ≈ sorted[left-split[⟨$\overline{X_0}$⟩]]:

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

Proof of (44.3) is-sorted[sorted[left-split[⟨$\overline{X_0}$⟩]]]:

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

Proof of (44.4) is-tuple[sorted[right-split[⟨$\overline{X_0}$⟩]]]:

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

Proof of (44.5) right-split[⟨$\overline{X_0}$⟩] ≈ sorted[right-split[⟨$\overline{X_0}$⟩]]:

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

Proof of (44.6) is-sorted[sorted[right-split[⟨$\overline{X_0}$⟩]]]:

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

Proof of (44.7) ¬ is-trivial-tuple[⟨$\overline{X_0}$⟩]:

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

Proof of (41.3)
is-sorted[merged[sorted[left-split[⟨$\overline{X_0}$⟩]], sorted[right-split[⟨$\overline{X_0}$⟩]]]]:

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

(52)

is-tuple[sorted[left-split[⟨$\overline{X_0}$⟩]]] ⋀ left-split[⟨$\overline{X_0}$⟩] ≈ sorted[left-split[⟨$\overline{X_0}$⟩]] ⋀.
is-sorted[sorted[left-split[⟨$\overline{X_0}$⟩]]] ⋀ is-tuple[sorted[right-split[⟨$\overline{X_0}$⟩]]] ⋀
right-split[⟨$\overline{X_0}$⟩] ≈ sorted[right-split[⟨$\overline{X_0}$⟩]] ⋀
is-sorted[sorted[right-split[⟨$\overline{X_0}$⟩]]] ⋀ ¬ is-trivial-tuple[⟨$\overline{X_0}$⟩]

We prove the individual conjunctive parts of (52):

Proof of (52.1) is-tuple[sorted[left-split[⟨$\overline{X_0}$⟩]]]:

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

Proof of (52.2) left-split[⟨$\overline{X_0}$⟩] ≈ sorted[left-split[⟨$\overline{X_0}$⟩]]:

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

Proof of (52.3) is-sorted[sorted[left-split[⟨$\overline{X_0}$⟩]]]:

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

Proof of (52.4) is-tuple[sorted[right-split[⟨$\overline{X_0}$⟩]]]:

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

Proof of (52.5) right-split[⟨$\overline{X_0}$⟩] ≈ sorted[right-split[⟨$\overline{X_0}$⟩]]:

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

Proof of (52.6) is-sorted[sorted[right-split[⟨$\overline{X_0}$⟩]]]:

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

Proof of (52.7) ¬ is-trivial-tuple[⟨$\overline{X_0}$⟩]:

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

□