

Lazy Thinking: A Reasoning Strategy for Inventing Algorithms

Bruno Buchberger

RISC (Research Institute for Symbolic Computation)
Johannes Kepler University, Linz, Austria

CADGME 2007, University of Pecs, Hungary

Conference "Algebra and Dynamic Geometry
in Mathematical Education", June 19 - 23, 2007

■ Copyright Bruno Buchberger 2007

Copyright Note: Copying, printing, distributing, and storing this document is granted under the following conditions:

- the document / file is kept unchanged and complete including this copyright note
- a message is sent to bruno.buchberger@jku.at
- if you use material of this talk then this talk should be cited appropriately.

A Simple Message

■ Improving Math Education

For improving mathematical education essentially,
teach the essentials.

■ The essence of Mathematics

The essence of mathematics is reasoning.

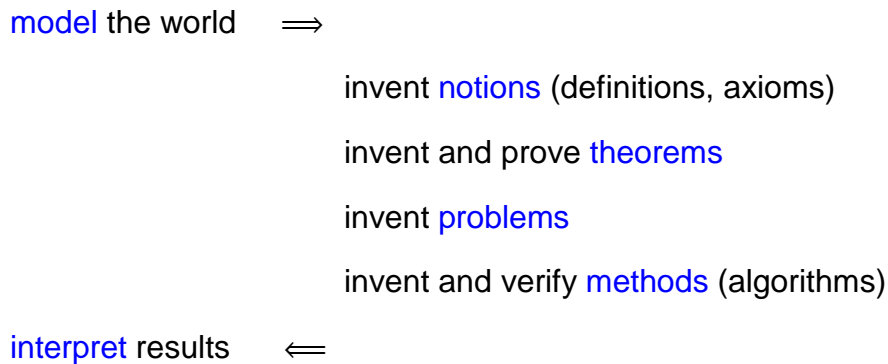
■ The "Thinking Course"

The "[TSW](#)" Course [BB 1982 ...]: Teach the practice of reasoning systematically.

Increased systematics leads to automation: the "[Theorema System](#)" [BB et al. 1995 ...]

■ Mathematical Exploration

Reasoning is instrumental in all phases of mathematical **exploration**:



■ Today's Talk

One Example:

Systematic (suitable for humans and machines)
 reasoning
 for **inventing correct algorithms**.

The approach is "lazy" [BB 2002, ...]

■ Reasoning is Formal

Reasoning: mathematical arguments must be such that "every idiot can check them".

(Not every idiot can invent them.)

Logic language: a language with "rules" for reasoning.

Predicate logic is a (practical) universal language for mathematical theory exploration.

The **Theorema** Project: Speaks predicate logic.

■ Acknowledgement

Other Members of Theorema Group:

T. Jebelean, M. Giese, T. Kutsia, F. Piroi, M. Rosenkranz, [W. Windsteiger](#)

and PhD students: [A. Craciun](#), [R. Vajda](#), ...

Inventing Algorithms by "Lazy Thinking"

■ Formal (Automated) Reasoning in Theorema: A Very Simple Example

```
Definition["addition", any[m, n],
  m + 0 = m      " +0:"
  m + n + = (m + n) + " + .:" ]
```

```
Proposition["left zero", any[m, n],
  0 + n = n  "0+"]
```

```
Prove[Proposition["left zero"],
  using → ⟨Definition["addition"]⟩,
  by → NNEqIndProver,

  ProverOptions → {TermOrder → LeftToRight},
  transformBy → ProofSimplifier, TransformerOptions → {branches → {Proved}}];
```

```
Compute[0++ + 0+++, using → ⟨Definition["addition"]⟩]
```

```
(((0+)+)+)+
```

■ The Algorithm Invention ("Synthesis") Problem

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

$$\forall_x P[x, A[x]].$$

Examples of specifications P:

```
P[x, y] ⇔ is-sorted-version[x, y]
P[x, y] ⇔ is-integral-of[x, y]
P[x, y] ⇔ is-Gröbner-basis[x, y]
....
```

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

■ Algorithm Synthesis by "Lazy Thinking" [BB 2002]

"Lazy Thinking" Method for Algorithm Synthesis =

My Advice to "Humans" (or "Computers") How to Invent Algorithms.

Given: A problem P. Find: An algorithm A for P.

- ♣ Consider known fundamental ideas of how to structure algorithms A in terms of subalgorithms B ("algorithm schemes").
Try one scheme after the other.
- ♣ For the chosen scheme, try to prove $\forall_x P[x, A[x]]$: From the **failing proof construct specifications** for the subalgorithms B occurring in A.

■ 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.

My 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.

And the work of A. Bundy and his group (U of Edinburgh) on the automated invention of induction schemes.

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

Problem: Synthesize "sorted" such that

$$\forall_x \text{is-sorted-version}[x, \text{sorted}[x]].$$

("Correctness Theorem")

Knowledge on Problem:

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

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

$$\forall_x \text{is-sorted}[\langle x \rangle]$$

$$\forall_{x,y,\bar{z}} \left(\text{is-sorted}[\langle x, y, \bar{z} \rangle] \Leftrightarrow \begin{array}{l} x \geq y \\ \text{is-sorted}[\langle y, \bar{z} \rangle] \end{array} \right)$$

etc.

■ An Algorithm Scheme: Divide and Conquer

$$\forall_x \left(\text{sorted}[x] = \begin{cases} S[x] & \Leftrightarrow \text{is-trivial-tuple}[x] \\ M[\text{sorted}[L[x]], \text{sorted}[R[x]]] & \Leftrightarrow \text{otherwise} \end{cases} \right)$$

The **unkown** algorithm **sorted** is expressed in terms of **unknown** algorithms **S, M, L, R** !

We now start an (automated) induction prover for proving the correctness theorem and analyze the failing proof: see notebooks with **failing proofs in the appendix**. At the end of each failing proof, the specification invention algorithm sketched in the next section automatically generates a specification for one of the subalgorithms **S, M, L, R**, adds this specification as additional knowledge to the knowledge base and (automatically) starts

the next round of proving. The final proof will then succeed. The specifications automatically generated are labeled by names containing the word "conjecture".

■ Automated Invention of Sufficient Specifications for the Subalgorithms

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

Collect temporary assumptions $T[x_0, \dots, A[\], \dots]$

and temporary goals $G[x_0, \dots, m[A[\]]]$

and produces specification

$$\forall_{x, \dots, y, \dots} (T[x, \dots, Y, \dots] \Rightarrow G[y, \dots, m[Y]]).$$

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):

$$\forall_x (\text{is-trivial-tuple}[x] \Rightarrow S[x] = x)$$

$$\forall_{y,z} \left(\begin{array}{l} \text{is-sorted}[y] \\ \text{is-sorted}[z] \end{array} \Rightarrow \begin{array}{l} \text{is-sorted}[M[y, z]] \\ M[y, z] \approx (y \times z) \end{array} \right)$$

$$\forall_x (L[x] \approx R[x] \approx x)$$

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

[What to we have now](#): A problem reduction !

■ Example: Synthesis of Insertion-Sort

Synthesize A such that

$$\forall_{\mathbf{x}} \text{is-sorted-version}[\mathbf{x}, \mathbf{A}[\mathbf{x}]].$$

Algorithm Scheme: "simple recursion"

$$\begin{aligned} \mathbf{A}[\langle \rangle] &= \mathbf{c} \\ \forall_{\mathbf{x}} \mathbf{A}[\langle \mathbf{x} \rangle] &= \mathbf{S}[\langle \mathbf{x} \rangle] \\ \forall_{\mathbf{x}, \bar{\mathbf{y}}} (\mathbf{A}[\langle \mathbf{x}, \bar{\mathbf{y}} \rangle] &= \mathbf{i}[\mathbf{x}, \mathbf{A}[\langle \bar{\mathbf{y}} \rangle]]) \end{aligned}$$

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

$$\begin{aligned} \mathbf{c} &= \langle \rangle \\ \forall_{\mathbf{x}} (\mathbf{S}[\langle \mathbf{x} \rangle] &= \langle \mathbf{x} \rangle) \\ \forall_{\mathbf{x}, \bar{\mathbf{y}}} \left(\text{is-sorted}[\langle \bar{\mathbf{y}} \rangle] \Rightarrow \right. & \left. \begin{array}{l} \text{is-sorted}[\mathbf{i}[\mathbf{x}, \langle \bar{\mathbf{y}} \rangle]] \\ \mathbf{i}[\langle \mathbf{x}, \bar{\mathbf{y}} \rangle] \approx (\mathbf{x} - \langle \bar{\mathbf{y}} \rangle) \end{array} \right) \end{aligned}$$

Inventing a Nontrivial Algorithm

■ How Far Can We Go With the "Lazy Thinking" Method ?

Can we automatically synthesize algorithms for [non-trivial problems](#)? What is "non-trivial"?

Example of [a non-trivial problem](#) (?): construction of Gröbner bases.

The problem was open for 65 years before an algorithm was found (in 1965 by a PhD student).

The problem was conjectured to be algorithmically unsolvable.

> 1000 papers were written on the algorithm and its numerous applications.

■ What is the Nontrivial Invention in Gröbner Bases Algorithmics

"Non-trivial" part of the invention: The [invention](#) of the notion of [S-polynomial](#) and the characterization of Gröbner-bases by finitely many S-polynomial checks.

With the "Lazy Thinking" method, it is possible to invent the essential idea of the B.B.'s Gröbner bases algorithm fully automatically: See [BB 2005] and ongoing PhD thesis by A. Craciun.

■ The Problem of Constructing Gröbner Bases

Find algorithm `Gb` such that

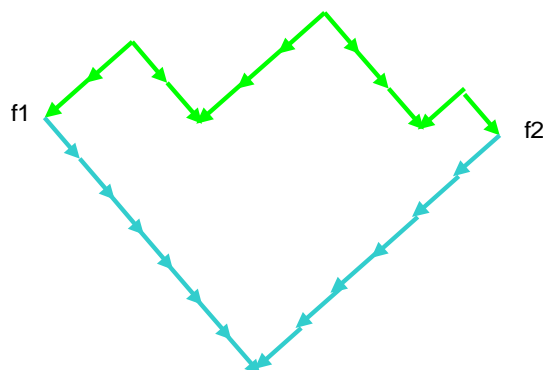
$$\forall_{\text{is-finite}[F]} \left(\begin{array}{l} \text{is-finite}[\text{Gb}[F]] \\ \text{is-Gröbner-basis}[\text{Gb}[F]] \\ \text{ideal}[F] = \text{ideal}[\text{Gb}[F]]. \end{array} \right)$$

$$\text{is-Gröbner-basis}[G] \Leftrightarrow \text{is-confluent}[\rightarrow_G].$$

\rightarrow_G ... a division step.

■ Confluence of Division \rightarrow_G

$$\text{is-confluent}[\rightarrow] : \Leftrightarrow \forall_{f_1, f_2} (f_1 \leftrightarrow^* f_2 \Rightarrow f_1 \downarrow^* f_2)$$



■ Knowledge on the Concepts Involved

$$h1 \rightarrow_G h2 \Rightarrow p . h1 \rightarrow_G p . h2$$

etc.

■ Algorithm Scheme "Critical Pair / Completion"

$$\begin{aligned}
 A[F] &= A[F, \text{pairs}[F]] \\
 A[F, \langle \rangle] &= F \\
 A[F, \langle \langle g1, g2 \rangle, \bar{p} \rangle] &= \\
 &\text{where } [f = \text{lc}[g1, g2], h1 = \text{trd}[\text{rd}[f, g1], F], h2 = \text{trd}[\text{rd}[f, g2], F], \\
 &\left\{ \begin{array}{l} A[F, \langle \bar{p} \rangle] \\ A[F - \text{df}[h1, h2], \langle \bar{p} \rangle] \times \left\langle \langle F_k, \text{df}[h1, h2] \rangle_{k=1, \dots, |F|} \right\rangle \end{array} \right\} \left[\begin{array}{l} \Leftarrow h1 = h2 \\ \Leftarrow \text{otherwise} \end{array} \right]
 \end{aligned}$$

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 $>$ which interacts with rd in the following natural way:

$$\forall_{f, g} (f > \text{rd}[f, g]).$$

■ The Essential Problem

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

$$\forall_F \left(\begin{array}{l} \text{is-finite}[A[F]] \\ \text{is-Gröbner-basis}[A[F]] \\ \text{ideal}[F] = \text{ideal}[A[F]]. \end{array} \right)$$

using the above scheme for A we can *automatically produce the idea* that

$$\text{lc}[g1, g2] = \text{lcm}[\text{lp}[g1], \text{lp}[g2]]$$

and

$$\mathit{df}[h1, h2] = h1 - h2$$

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 can be done automatically. (Ongoing PhD thesis by my PhD student A. Craciun.)

■ Details

□ After Termination

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

$$\forall_{g1, g2 \in G} \left(\text{where} [f = \mathit{lc}[g1, g2], h1 = \mathit{trd}[\mathit{rd}[f, g1], F], \right. \\ \left. h2 = \mathit{trd}[\mathit{rd}[f, g2], F], \bigvee \left\{ \begin{array}{l} h1 = h2 \\ \mathit{df}[h1, h2] \in G \end{array} \right\} \right].$$

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.

□ Using Available Knowledge

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

$$\mathit{is-Church-Rosser}[\rightarrow_G] \Leftrightarrow \forall_p \forall_{f1, f2} \left(\left(\left\{ \begin{array}{l} p \rightarrow f1 \\ p \rightarrow f2 \end{array} \right\} \Rightarrow f1 \downarrow^* f2 \right) \right).$$

Newman's lemma (1942):

$$\text{is-Church-Rosser}[\rightarrow] \Leftrightarrow \forall_{f, f_1, f_2} \left(\left(\begin{array}{l} f \rightarrow f_1 \\ f \rightarrow f_2 \end{array} \right) \Rightarrow f_1 \downarrow^* f_2 \right).$$

Definition of "f1 and f2 have a common successor":

$$f_1 \downarrow^* f_2 \Leftrightarrow \exists_g \left(\begin{array}{l} f_1 \rightarrow^* g \\ f_2 \rightarrow^* g \end{array} \right)$$

□ The (Automated) Proof Attempt

Let now the power product p and the polynomials f_1, f_2 be arbitrary but fixed and assume

$$\begin{cases} p \rightarrow_G f_1 \\ p \rightarrow_G f_2. \end{cases}$$

We have to find a polynomial g such that

$$\begin{cases} f_1 \rightarrow_G^* g, \\ f_2 \rightarrow_G^* g. \end{cases}$$

From the assumption we know that there exist polynomials g_1 and g_2 in G such that

$$\begin{cases} lp[g_1] \mid p, \\ f_1 = rd[p, g_1], \\ lp[g_2] \mid p, \\ f_2 = rd[p, g_2]. \end{cases}$$

From the final situation in the algorithm scheme we know that for these g_1 and g_2

$$\bigvee \begin{cases} h_1 = h_2 \\ df[h_1, h_2] \in G, \end{cases}$$

where

$$\begin{cases} h_1 := \text{trd}[f_1', G], f_1' := rd[lc[g_1, g_2], g_1], \\ h_2 := \text{trd}[f_2', G], f_2' := rd[lc[g_1, g_2], g_2]. \end{cases}$$

□ Case $h_1=h_2$

$$\begin{aligned} lc[g_1, g_2] \rightarrow_{g_1} rd[lc[g_1, g_2], g_1] \rightarrow_G^* \text{trd}[rd[lc[g_1, g_2], g_1], G] = \\ \text{trd}[rd[lc[g_1, g_2], g_2], G] \leftarrow_G^* rd[lc[g_1, g_2], g_2] \leftarrow_{g_2} lc[g_1, g_2]. \end{aligned}$$

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

Hence, by elementary properties of polynomial reduction,

$$\forall_{a,q} (a \ q \ lc[g1, g2] \rightarrow_{g1} a \ q \ rd[lc[g1, g2], g1] \rightarrow_{G^*} a \ q \ trd[rd[lc[g1, g2], g1], G] = a \ q \ trd[rd[lc[g1, g2], g2], G] \leftarrow_{G^*} a \ q \ rd[lc[g1, g2], g2] \leftarrow_{g2} a \ q \ lc[g1, g2]).$$

Now we are stuck in the proof.

□ 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

$$\forall_{p,g1,g2} (((\{ \begin{matrix} lp[g1] \\ lp[g2] \end{matrix} \mid p) \Rightarrow (\exists_{a,q} (p = a \ q \ lc[g1, g2])))), \quad (lc \text{ requirement})$$

With such an lc , we then would have

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

and, hence,

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

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

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

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

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

$$\forall_{p,g1,g2} (((\{ \begin{matrix} lp[g1] \\ lp[g2] \end{matrix} \mid p) \Rightarrow (lc[g1, g2] \mid p))),$$

and the requirements:

$$\begin{aligned} l_p[g1] &| lc[g1, g2], \\ l_p[g2] &| lc[g1, g2]. \end{aligned}$$

Now this problem can be attacked independently of any Gröbner bases theory, ideal theory etc.

■ A Suitable lc

$$l_{cp}[g1, g2] = l_{cm}[l_p[g1], l_p[g2]]$$

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!

■ Case $h1 \neq h2$

In this case, $df[h1, h2] \in G$:

In this part of the proof we are basically stuck right at the beginning.

We can try to reduce this case to the first case, which would generate the following requirement

$$\forall_{h1, h2} (h1 \downarrow_{\{df[h1, h2]\}}^* h2) \quad (\text{df requirement}).$$

■ 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 satisfies (df requirement), namely

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

because, in fact,

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

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

Conclusion



With today's (automated) reasoning power, (nontrivial) PhD students of the 60ies can be trivialized.

Better start to teach (study) reasoning before you are trivialized.

Don't be lazy, study "lazy thinking".

References to Details of This Talk

□ **On my "Thinking, Speaking, Writing" Course**

B. Buchberger.

Thinking, Speaking, Writing: A Course on Using Predicate Logic as a Working Language.

Lecture Notes, RISC (Research Institute for Symbolic Computation), Johannes Kepler University, Linz, Austria, 1982 - 2007.

□ **On my "White Box / Black Box Principle" for the Didactics of Using Math Software Systems for Math Teaching**

B. Buchberger

Should Students Learn Integration Rules?

ACM SIGSAM Bulletin Vol.24/1, January 1990, pp. 10-17.

□ **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, 2005, pp. 65-75.

Appendix: Automatically Generated Proof Notebooks for the Sorting Example

■ Notebook 1: automated failing proof, automated generation of specification "conjecture 15" for subalgorithm "special" (which corresponds to "S" in the talk).

Prove:

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

under the assumptions:

(Definition (is sorted): 1) $\text{is-sorted}[\langle \rangle],$

(Definition (is sorted): 2) $\forall_{\mathbf{x}} \text{is-sorted}[\langle \mathbf{x} \rangle],$

(Definition (is sorted): 3) $\forall_{\mathbf{x}, \mathbf{y}, \bar{\mathbf{z}}} (\text{is-sorted}[\langle \mathbf{x}, \mathbf{y}, \bar{\mathbf{z}} \rangle] \Leftrightarrow \mathbf{x} \geq \mathbf{y} \wedge \text{is-sorted}[\langle \mathbf{y}, \bar{\mathbf{z}} \rangle]),$

(Definition (is permuted version): 1) $\langle \rangle \approx \langle \rangle,$

(Definition (is permuted version): 2) $\forall_{\mathbf{y}, \bar{\mathbf{y}}} (\langle \rangle \neq \langle \mathbf{y}, \bar{\mathbf{y}} \rangle),$

(Definition (is permuted version): 3) $\forall_{\mathbf{x}, \bar{\mathbf{x}}, \bar{\mathbf{y}}} (\langle \bar{\mathbf{y}} \rangle \approx \langle \mathbf{x}, \bar{\mathbf{x}} \rangle \Leftrightarrow \mathbf{x} \in \langle \bar{\mathbf{y}} \rangle \wedge \text{dfo}[\mathbf{x}, \langle \bar{\mathbf{y}} \rangle] \approx \langle \bar{\mathbf{x}} \rangle),$

(Definition (is sorted version))

$\forall_{\substack{\mathbf{x}, \mathbf{y} \\ \text{is-tuple}[\mathbf{x}]}} (\text{is-sorted-version}[\mathbf{x}, \mathbf{y}] \Leftrightarrow \text{is-tuple}[\mathbf{y}] \wedge \mathbf{x} \approx \mathbf{y} \wedge \text{is-sorted}[\mathbf{y}]),$

(Proposition (is tuple tuple)) $\forall_{\mathbf{x}} \text{is-tuple}[\langle \bar{\mathbf{x}} \rangle],$

(Definition (prepend): \neg) $\forall_{\mathbf{x}, \bar{\mathbf{y}}} (\mathbf{x} - \langle \bar{\mathbf{y}} \rangle = \langle \mathbf{x}, \bar{\mathbf{y}} \rangle),$

(Proposition (singleton tuple is singleton tuple)) $\forall_{\mathbf{x}} \text{is-singleton-tuple}[\langle \mathbf{x} \rangle],$

(Definition (is trivial tuple))

$$\forall_{\text{is-tuple}[\mathbf{X}]} (\text{is-trivial-tuple}[\mathbf{X}] \Leftrightarrow \text{is-empty-tuple}[\mathbf{X}] \vee \text{is-singleton-tuple}[\mathbf{X}]),$$

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

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

(Definition (deletion of the first occurrence): 1) $\forall_{\mathbf{a}} (\text{dfo}[\mathbf{a}, \langle \rangle] = \langle \rangle),$

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

$$\forall_{\mathbf{a}, \mathbf{x}, \bar{\mathbf{x}}} (\text{dfo}[\mathbf{a}, \langle \mathbf{x}, \bar{\mathbf{x}} \rangle] = \|\langle \bar{\mathbf{x}} \rangle \Leftarrow \mathbf{x} = \mathbf{a}, \mathbf{x} - \text{dfo}[\mathbf{a}, \langle \bar{\mathbf{x}} \rangle] \Leftarrow \text{otherwise}\|),$$

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

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

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

(Proposition (trivial tuples are sorted)) $\forall_{\bar{\mathbf{x}}} \text{is-sorted}[\langle \bar{\mathbf{x}} \rangle],$
 $\text{is-trivial-tuple}[\langle \bar{\mathbf{x}} \rangle]$

(Proposition (only trivial tuple permuted version of itself)) $\forall_{\bar{\mathbf{x}}, \mathbf{y}} ((\mathbf{Y} = \langle \bar{\mathbf{x}} \rangle) \Rightarrow \mathbf{Y} \approx \langle \bar{\mathbf{x}} \rangle),$
 $\text{is-trivial-tuple}[\langle \bar{\mathbf{x}} \rangle]$

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

(Algorithm (sorted))

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

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

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

(Lemma (splits are tuples): 2) $\forall_{\mathbf{X}} \text{is-tuple}[\text{right-split}[\mathbf{X}]],$
 $\text{is-tuple}[\mathbf{X}] \wedge \text{is-trivial-tuple}[\mathbf{X}]$

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

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

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

We try to prove (Theorem (correctness of sort)) by well-founded induction on \mathbf{X} .

Well-founded induction:

Assume:

(1) $\text{is-tuple}[\langle \bar{\mathbf{X}}_0 \rangle].$

Well-Founded Induction Hypothesis:

$$(2) \quad \forall_{\text{is-tuple}[\mathbf{x1}]} (\langle \overline{X_0} \rangle > \mathbf{x1} \Rightarrow \text{is-sorted-version}[\mathbf{x1}, \text{sorted}[\mathbf{x1}]])$$

We have to show:

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

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

Case 1:

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

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 (only trivial tuple permuted version of itself)), implies:

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

Formula (1), by ([Lemma \(Closure of Special\)](#)), implies:

$$(12) \text{is-tuple}[\text{special}[\langle \overline{X_0} \rangle]].$$

By (1), Formula (5), using (Definition (is sorted version)), is implied by:

$$(13) \text{is-tuple}[\text{special}[\langle \overline{X_0} \rangle]] \wedge \text{special}[\langle \overline{X_0} \rangle] \approx \langle \overline{X_0} \rangle \wedge \text{is-sorted}[\text{special}[\langle \overline{X_0} \rangle]].$$

Not all the conjunctive parts of (13) can be proved.

Proof of (13.1) $\text{is-tuple}[\text{special}[\langle \overline{X_0} \rangle]]$:

Formula (13.1) is true because it is identical to (12).

Proof of (13.2) $\text{special}[\langle \overline{X_0} \rangle] \approx \langle \overline{X_0} \rangle$:

Formula (13.3), using (10), is implied by:

$$(14) \text{special}[\langle \overline{X_0} \rangle] = \langle \overline{X_0} \rangle.$$

The proof of (14) fails. (The prover "QR" was unable to transform the proof situation.)

Proof of (13.4) $\text{is-sorted}[\text{special}[\langle \overline{X_0} \rangle]]$:

Pending proof of (13.4).

Case 2:

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

Hence, we have to prove

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

Pending proof of (8).

□

■ **Notebook 2: automated failing proof, automated generation of specification "conjecture 44" for subalgorithms "left-split", "right-split", and "merged" (which correspond to "L", "R", and "M" in the talk).**

Prove:

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

under the assumptions:

(Definition (is sorted): 1) $\text{is-sorted}[\langle \rangle],$

(Definition (is sorted): 2) $\forall_{\mathbf{x}} \text{is-sorted}[\langle \mathbf{x} \rangle],$

(Definition (is sorted): 3) $\forall_{\mathbf{x}, \mathbf{y}, \bar{\mathbf{z}}} (\text{is-sorted}[\langle \mathbf{x}, \mathbf{y}, \bar{\mathbf{z}} \rangle] \Leftrightarrow \mathbf{x} \geq \mathbf{y} \wedge \text{is-sorted}[\langle \mathbf{y}, \bar{\mathbf{z}} \rangle]),$

(Definition (is permuted version): 1) $\langle \rangle \approx \langle \rangle,$

(Definition (is permuted version): 2) $\forall_{\mathbf{y}, \bar{\mathbf{y}}} (\langle \rangle \neq \langle \mathbf{y}, \bar{\mathbf{y}} \rangle),$

(Definition (is permuted version): 3) $\forall_{\mathbf{x}, \bar{\mathbf{x}}, \bar{\mathbf{y}}} (\langle \bar{\mathbf{y}} \rangle \approx \langle \mathbf{x}, \bar{\mathbf{x}} \rangle \Leftrightarrow \mathbf{x} \in \langle \bar{\mathbf{y}} \rangle \wedge \text{dfo}[\mathbf{x}, \langle \bar{\mathbf{y}} \rangle] \approx \langle \bar{\mathbf{x}} \rangle),$

(Definition (is sorted version))

$\forall_{\substack{\mathbf{x}, \mathbf{y} \\ \text{is-tuple}[\mathbf{x}]}} (\text{is-sorted-version}[\mathbf{x}, \mathbf{y}] \Leftrightarrow \text{is-tuple}[\mathbf{y}] \wedge \mathbf{x} \approx \mathbf{y} \wedge \text{is-sorted}[\mathbf{y}]),$

(Proposition (is tuple tuple)) $\forall_{\bar{\mathbf{x}}} \text{is-tuple}[\langle \bar{\mathbf{x}} \rangle],$

(Definition (prepend): \neg) $\forall_{\mathbf{x}, \bar{\mathbf{y}}} (\mathbf{x} \sim \langle \bar{\mathbf{y}} \rangle = \langle \mathbf{x}, \bar{\mathbf{y}} \rangle),$

(Proposition (singleton tuple is singleton tuple)) $\forall_{\mathbf{x}} \text{is-singleton-tuple}[\langle \mathbf{x} \rangle],$

(Definition (is trivial tuple))

$\forall_{\text{is-tuple}[\mathbf{x}]} (\text{is-trivial-tuple}[\mathbf{x}] \Leftrightarrow \text{is-empty-tuple}[\mathbf{x}] \vee \text{is-singleton-tuple}[\mathbf{x}]),$

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

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

(Definition (deletion of the first occurrence): 1) $\forall_{\mathbf{a}} (\text{dfo}[\mathbf{a}, \langle \rangle] = \langle \rangle),$

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

$\forall_{\mathbf{a}, \mathbf{x}, \bar{\mathbf{x}}} (\text{dfo}[\mathbf{a}, \langle \mathbf{x}, \bar{\mathbf{x}} \rangle] = \|\langle \bar{\mathbf{x}} \rangle \Leftarrow \mathbf{x} = \mathbf{a}, \mathbf{x} \sim \text{dfo}[\mathbf{a}, \langle \bar{\mathbf{x}} \rangle] \Leftarrow \text{otherwise}\|),$

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

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

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

(Proposition (trivial tuples are sorted)) $\forall_{\bar{x}} \text{is-trivial-tuple}[\langle \bar{x} \rangle] \Rightarrow \text{is-sorted}[\langle \bar{x} \rangle],$

(Proposition (only trivial tuple permuted version of itself)) $\forall_{\bar{x}, \bar{y}} (\text{is-trivial-tuple}[\langle \bar{x} \rangle] \wedge (\bar{y} = \langle \bar{x} \rangle) \Rightarrow \bar{y} \approx \langle \bar{x} \rangle),$

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

(Algorithm (sorted))

$\forall_{\mathbf{x}} (\text{sorted}[\mathbf{x}] = \|\text{special}[\mathbf{x}] \Leftarrow \text{is-trivial-tuple}[\mathbf{x}], \text{merged}[\text{sorted}[\text{left-split}[\mathbf{x}]], \text{sorted}[\text{right-split}[\mathbf{x}]]] \Leftarrow \text{otherwise}\|)$

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

(Lemma (splits are tuples): 1) $\forall_{\mathbf{x}} \text{is-tuple}[\text{left-split}[\mathbf{x}]] \wedge \text{is-trivial-tuple}[\mathbf{x}] \Rightarrow \text{is-tuple}[\mathbf{x}],$

(Lemma (splits are tuples): 2) $\forall_{\mathbf{x}} \text{is-tuple}[\text{right-split}[\mathbf{x}]] \wedge \text{is-trivial-tuple}[\mathbf{x}] \Rightarrow \text{is-tuple}[\mathbf{x}],$

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

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

(Lemma (closure of merge)) $\forall_{\mathbf{x}, \mathbf{y}} \text{is-tuple}[\mathbf{x}] \wedge \text{is-tuple}[\mathbf{y}] \Rightarrow \text{is-tuple}[\text{merged}[\mathbf{x}, \mathbf{y}]],$

(Lemma (conjecture15): conjecture15) $\forall_{\mathbf{x1}} \text{is-tuple}[\mathbf{x1}] \Rightarrow (\text{is-trivial-tuple}[\mathbf{x1}] \Rightarrow (\text{special}[\mathbf{x1}] = \mathbf{x1})).$

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 \mathbf{X} .

Well-founded induction:

Assume:

(1) $\text{is-tuple}[\langle \overline{\mathbf{x0}} \rangle].$

Well-Founded Induction Hypothesis:

(2) $\forall_{\mathbf{x2}} (\text{is-tuple}[\mathbf{x2}] \wedge (\langle \overline{\mathbf{x0}} \rangle > \mathbf{x2} \Rightarrow \text{is-sorted-version}[\mathbf{x2}, \text{sorted}[\mathbf{x2}]])$

We have to show:

(3) $\text{is-sorted-version}[\langle \overline{\mathbf{x0}} \rangle, \text{sorted}[\langle \overline{\mathbf{x0}} \rangle]].$

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) $\text{is-trivial-tuple}[\langle \overline{\mathbf{x0}} \rangle].$

Hence, we have to prove

(5) $\text{is-sorted-version}[\langle \overline{\mathbf{x0}} \rangle, \text{special}[\langle \overline{\mathbf{x0}} \rangle]].$

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:

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

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:

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

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

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

We prove the individual conjunctive parts of (22):

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

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) $\text{is-sorted}[\langle \overline{X_0} \rangle]$:

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

Case 2:

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

Hence, we have to prove

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

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:

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

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

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

$$(25)$$

$$\text{is-tuple}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]]] \wedge \text{left-split}[\langle \overline{X_0} \rangle] \approx \text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]] \wedge \text{is-sorted}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]]].$$

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

(26) $\text{is-tuple}[\text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]] \wedge$
 $\text{right-split}[\langle \bar{X}_0 \rangle] \approx \text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]] \wedge$
 $\text{is-sorted}[\text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]]$

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

(41) $\text{is-tuple}[\text{merged}[\text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]], \text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]] \wedge$
 $\text{merged}[\text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]], \text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]] \approx \langle \bar{X}_0 \rangle \wedge$
 $\text{is-sorted}[\text{merged}[\text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]], \text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]]$

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

Proof of (41.1) $\text{is-tuple}[\text{merged}[\text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]], \text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]]$:

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

(42) $\text{is-tuple}[\text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]]] \wedge \text{is-tuple}[\text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]]$.

We prove the individual conjunctive parts of (42):

Proof of (42.1) $\text{is-tuple}[\text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]]]$:

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

Proof of (42.2) $\text{is-tuple}[\text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]]$:

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

Proof of (41.3) $\text{merged}[\text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]], \text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]] \approx \langle \bar{X}_0 \rangle$:

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

Proof of (41.4)

$\text{is-sorted}[\text{merged}[\text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]], \text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]]$:

Pending proof of (41.4).

□

■ Notebook 3: automated failing proof, automated generation of specification

"conjecture 46" for subalgorithms "left-split", "right-split", and "merged" (which correspond to "L", "R", and "M" in the talk).

Prove:

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

under the assumptions:

(Definition (is sorted): 1) $\text{is-sorted}[\langle \rangle]$,

(Definition (is sorted): 2) $\forall_{\mathbf{x}} \text{is-sorted}[\langle \mathbf{x} \rangle]$,

(Definition (is sorted): 3) $\forall_{\mathbf{x}, \mathbf{y}, \bar{\mathbf{z}}} (\text{is-sorted}[\langle \mathbf{x}, \mathbf{y}, \bar{\mathbf{z}} \rangle] \Leftrightarrow \mathbf{x} \geq \mathbf{y} \wedge \text{is-sorted}[\langle \mathbf{y}, \bar{\mathbf{z}} \rangle])$,

(Definition (is permuted version): 1) $\langle \rangle \approx \langle \rangle$,

(Definition (is permuted version): 2) $\forall_{\mathbf{y}, \bar{\mathbf{y}}} (\langle \rangle \neq \langle \mathbf{y}, \bar{\mathbf{y}} \rangle),$

(Definition (is permuted version): 3) $\forall_{\mathbf{x}, \bar{\mathbf{x}}, \bar{\mathbf{y}}} (\langle \bar{\mathbf{y}} \rangle \approx \langle \mathbf{x}, \bar{\mathbf{x}} \rangle \Leftrightarrow \mathbf{x} \in \langle \bar{\mathbf{y}} \rangle \wedge \text{dfo}[\mathbf{x}, \langle \bar{\mathbf{y}} \rangle] \approx \langle \bar{\mathbf{x}} \rangle),$

(Definition (is sorted version))

$\forall_{\mathbf{x}, \mathbf{y}} (\text{is-sorted-version}[\mathbf{x}, \mathbf{y}] \Leftrightarrow \text{is-tuple}[\mathbf{y}] \wedge \mathbf{x} \approx \mathbf{y} \wedge \text{is-sorted}[\mathbf{y}]),$
 $\text{is-tuple}[\mathbf{x}]$

(Proposition (is tuple tuple)) $\forall_{\bar{\mathbf{x}}} \text{is-tuple}[\langle \bar{\mathbf{x}} \rangle],$

(Definition (prepend): \neg) $\forall_{\mathbf{x}, \bar{\mathbf{y}}} (\mathbf{x} - \langle \bar{\mathbf{y}} \rangle = \langle \mathbf{x}, \bar{\mathbf{y}} \rangle),$

(Proposition (singleton tuple is singleton tuple)) $\forall_{\mathbf{x}} \text{is-singleton-tuple}[\langle \mathbf{x} \rangle],$

(Definition (is trivial tuple))

$\forall_{\mathbf{x}} (\text{is-trivial-tuple}[\mathbf{x}] \Leftrightarrow \text{is-empty-tuple}[\mathbf{x}] \vee \text{is-singleton-tuple}[\mathbf{x}]),$
 $\text{is-tuple}[\mathbf{x}]$

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

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

(Definition (deletion of the first occurrence): 1) $\forall_{\mathbf{a}} (\text{dfo}[\mathbf{a}, \langle \rangle] = \langle \rangle),$

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

$\forall_{\mathbf{a}, \mathbf{x}, \bar{\mathbf{x}}} (\text{dfo}[\mathbf{a}, \langle \mathbf{x}, \bar{\mathbf{x}} \rangle] = \|\langle \bar{\mathbf{x}} \rangle \Leftarrow \mathbf{x} = \mathbf{a}, \mathbf{x} - \text{dfo}[\mathbf{a}, \langle \bar{\mathbf{x}} \rangle] \Leftarrow \text{otherwise}\|),$

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

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

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

(Proposition (trivial tuples are sorted)) $\forall_{\bar{\mathbf{x}}} \text{is-trivial-tuple}[\langle \bar{\mathbf{x}} \rangle] \Rightarrow \text{is-sorted}[\langle \bar{\mathbf{x}} \rangle],$

(Proposition (only trivial tuple permuted version of itself)) $\forall_{\bar{\mathbf{x}}, \mathbf{y}} (\text{is-trivial-tuple}[\langle \bar{\mathbf{x}} \rangle] \wedge \langle \mathbf{y} \rangle = \langle \bar{\mathbf{x}} \rangle \Rightarrow \mathbf{y} \approx \langle \bar{\mathbf{x}} \rangle),$

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

(Algorithm (sorted))

$\forall_{\mathbf{x}} (\text{sorted}[\mathbf{x}] = \|\text{special}[\mathbf{x}] \Leftarrow \text{is-trivial-tuple}[\mathbf{x}],$
 $\text{merged}[\text{sorted}[\text{left-split}[\mathbf{x}]], \text{sorted}[\text{right-split}[\mathbf{x}]]] \Leftarrow \text{otherwise}\|)$

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

(Lemma (splits are tuples): 1) $\forall_{\mathbf{x}} \text{is-tuple}[\text{left-split}[\mathbf{x}]],$
 $\text{is-tuple}[\mathbf{x}] \wedge \text{is-trivial-tuple}[\mathbf{x}]$

(Lemma (splits are tuples): 2) $\forall_{\mathbf{x}} \text{is-tuple}[\text{right-split}[\mathbf{x}]],$
 $\text{is-tuple}[\mathbf{x}] \wedge \text{is-trivial-tuple}[\mathbf{x}]$

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

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

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

(Lemma (conjecture15): conjecture15)

$\forall_{\substack{\mathbf{x1} \\ \text{is-tuple}[\mathbf{x1}]}} (\text{is-trivial-tuple}[\mathbf{x1}] \wedge \text{is-sorted}[\mathbf{x1}] \Rightarrow (\text{special}[\mathbf{x1}] = \mathbf{x1})),$

(Lemma (conjecture44): conjecture44)

$\forall_{\substack{\mathbf{x2}, \mathbf{x3}, \mathbf{x4} \\ \text{is-tuple}[\mathbf{x4}]}} (\text{is-tuple}[\mathbf{x2}] \wedge \text{left-split}[\mathbf{x4}] \approx \mathbf{x2} \wedge$
 $\text{is-sorted}[\mathbf{x2}] \wedge \text{is-tuple}[\mathbf{x3}] \wedge \text{right-split}[\mathbf{x4}] \approx \mathbf{x3} \wedge$
 $\text{is-sorted}[\mathbf{x3}] \wedge \neg \text{is-trivial-tuple}[\mathbf{x4}] \Rightarrow \text{merged}[\mathbf{x2}, \mathbf{x3}] \approx \mathbf{x4})$

We try to prove (Theorem (correctness of sort)) by well-founded induction on \mathbf{X} .

Well-founded induction:

Assume:

(1) $\text{is-tuple}[\langle \overline{\mathbf{x0}} \rangle].$

Well-Founded Induction Hypothesis:

(2) $\forall_{\text{is-tuple}[\mathbf{x3}]} (\langle \overline{\mathbf{x0}} \rangle > \mathbf{x3} \Rightarrow \text{is-sorted-version}[\mathbf{x3}, \text{sorted}[\mathbf{x3}]])$

We have to show:

(3) $\text{is-sorted-version}[\langle \overline{\mathbf{x0}} \rangle, \text{sorted}[\langle \overline{\mathbf{x0}} \rangle]].$

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) $\text{is-trivial-tuple}[\langle \overline{\mathbf{x0}} \rangle].$

Hence, we have to prove

(5) $\text{is-sorted-version}[\langle \overline{\mathbf{x0}} \rangle, \text{special}[\langle \overline{\mathbf{x0}} \rangle]].$

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

(9) $\text{is-sorted}[\langle \overline{\mathbf{x0}} \rangle].$

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

(10) $\forall_{\mathbf{y}} ((\mathbf{y} = \langle \overline{\mathbf{x0}} \rangle) \Rightarrow \mathbf{y} \approx \langle \overline{\mathbf{x0}} \rangle).$

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

(11) $\text{is-tuple}[\text{special}[\langle \overline{\mathbf{x0}} \rangle]].$

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

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

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

We prove the individual conjunctive parts of (22):

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

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) $\text{is-sorted}[\langle \overline{X_0} \rangle]$:

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

Case 2:

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

Hence, we have to prove

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

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

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

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

(25)

$\text{is-tuple}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]]] \wedge \text{left-split}[\langle \overline{X_0} \rangle] \approx \text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]] \wedge \text{is-sorted}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]]]$.

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

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

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

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

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

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

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

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

We prove the individual conjunctive parts of (42):

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

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

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

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

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

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

(44)

$$\begin{aligned} & \text{is-tuple}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]]] \wedge \text{left-split}[\langle \overline{X_0} \rangle] \approx \text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]] \wedge \\ & \text{is-sorted}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]]] \wedge \text{is-tuple}[\text{sorted}[\text{right-split}[\langle \overline{X_0} \rangle]]] \wedge \\ & \text{right-split}[\langle \overline{X_0} \rangle] \approx \text{sorted}[\text{right-split}[\langle \overline{X_0} \rangle]] \wedge \\ & \text{is-sorted}[\text{sorted}[\text{right-split}[\langle \overline{X_0} \rangle]]] \wedge \neg \text{is-trivial-tuple}[\langle \overline{X_0} \rangle] \end{aligned}$$

We prove the individual conjunctive parts of (44):

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Proof of (41.3)

$\text{is-sorted}[\text{merged}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]], \text{sorted}[\text{right-split}[\langle \overline{X_0} \rangle]]]]$:

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

□

■ **Notebook 4: automated succeeding proof, invention of correct (!) algorithm "sorted" completed (under the assumption that one provides subalgorithms "special", "left-split", "right-split", "merged" that meet the automatically generated specifications)**

Prove:

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

under the assumptions:

(Definition (is sorted): 1) $\text{is-sorted}[\langle \rangle],$

(Definition (is sorted): 2) $\forall_{\mathbf{x}} \text{is-sorted}[\langle \mathbf{x} \rangle],$

(Definition (is sorted): 3) $\forall_{\mathbf{x}, \mathbf{y}, \bar{\mathbf{z}}} (\text{is-sorted}[\langle \mathbf{x}, \mathbf{y}, \bar{\mathbf{z}} \rangle] \Leftrightarrow \mathbf{x} \geq \mathbf{y} \wedge \text{is-sorted}[\langle \mathbf{y}, \bar{\mathbf{z}} \rangle]),$

(Definition (is permuted version): 1) $\langle \rangle \approx \langle \rangle,$

(Definition (is permuted version): 2) $\forall_{\mathbf{y}, \bar{\mathbf{y}}} (\langle \rangle \neq \langle \mathbf{y}, \bar{\mathbf{y}} \rangle),$

(Definition (is permuted version): 3) $\forall_{\mathbf{x}, \bar{\mathbf{x}}, \bar{\mathbf{y}}} (\langle \bar{\mathbf{y}} \rangle \approx \langle \mathbf{x}, \bar{\mathbf{x}} \rangle \Leftrightarrow \mathbf{x} \in \langle \bar{\mathbf{y}} \rangle \wedge \text{dfo}[\mathbf{x}, \langle \bar{\mathbf{y}} \rangle] \approx \langle \bar{\mathbf{x}} \rangle),$

(Definition (is sorted version))

$\forall_{\substack{\mathbf{x}, \mathbf{y} \\ \text{is-tuple}[\mathbf{x}]}} (\text{is-sorted-version}[\mathbf{x}, \mathbf{y}] \Leftrightarrow \text{is-tuple}[\mathbf{y}] \wedge \mathbf{x} \approx \mathbf{y} \wedge \text{is-sorted}[\mathbf{y}]),$

(Proposition (is tuple tuple)) $\forall_{\bar{\mathbf{x}}} \text{is-tuple}[\langle \bar{\mathbf{x}} \rangle],$

(Definition (prepend): \neg) $\forall_{\mathbf{x}, \bar{\mathbf{y}}} (\mathbf{x} - \langle \bar{\mathbf{y}} \rangle = \langle \mathbf{x}, \bar{\mathbf{y}} \rangle),$

(Proposition (singleton tuple is singleton tuple)) $\forall_{\mathbf{x}} \text{is-singleton-tuple}[\langle \mathbf{x} \rangle],$

(Definition (is trivial tuple))

$\forall_{\text{is-tuple}[\mathbf{x}]} (\text{is-trivial-tuple}[\mathbf{x}] \Leftrightarrow \text{is-empty-tuple}[\mathbf{x}] \vee \text{is-singleton-tuple}[\mathbf{x}]),$

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

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

(Definition (deletion of the first occurrence): 1) $\forall_{\mathbf{a}} (\text{dfo}[\mathbf{a}, \langle \rangle] = \langle \rangle),$

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

$\forall_{\mathbf{a}, \mathbf{x}, \bar{\mathbf{x}}} (\text{dfo}[\mathbf{a}, \langle \mathbf{x}, \bar{\mathbf{x}} \rangle] = \|\langle \bar{\mathbf{x}} \rangle \Leftarrow \mathbf{x} = \mathbf{a}, \mathbf{x} - \text{dfo}[\mathbf{a}, \langle \bar{\mathbf{x}} \rangle] \Leftarrow \text{otherwise}\|),$

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

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

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

(Proposition (trivial tuples are sorted)) $\forall_{\bar{\mathbf{x}}} \text{is-trivial-tuple}[\langle \bar{\mathbf{x}} \rangle] \Rightarrow \text{is-sorted}[\langle \bar{\mathbf{x}} \rangle],$

(Proposition (only trivial tuple permuted version of itself)) $\forall_{\bar{\mathbf{x}}, \mathbf{y}} (\text{is-trivial-tuple}[\langle \bar{\mathbf{x}} \rangle] \Rightarrow (\mathbf{Y} = \langle \bar{\mathbf{x}} \rangle \Rightarrow \mathbf{Y} \approx \langle \bar{\mathbf{x}} \rangle),$

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

(Algorithm (sorted))

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

(Lemma (closure of special)) $\forall_{\mathbf{x}} \text{is-tuple}[\text{special}[\mathbf{X}]],$

(Lemma (splits are tuples): 1) $\forall_{\mathbf{x}} \text{is-tuple}[\text{left-split}[\mathbf{X}]],$

(Lemma (splits are tuples): 2) $\forall_{\mathbf{x}} \text{is-tuple}[\text{right-split}[\mathbf{X}]],$

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

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

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

(Lemma (conjecture15): conjecture15)

$\forall_{\text{is-tuple}[\mathbf{x1}]} (\text{is-trivial-tuple}[\mathbf{X1}] \wedge \text{is-sorted}[\mathbf{X1}] \Rightarrow (\text{special}[\mathbf{X1}] = \mathbf{X1})),$

(Lemma (conjecture44): conjecture44)

$\forall_{\text{is-tuple}[\mathbf{x4}]} (\text{is-tuple}[\mathbf{X2}] \wedge \text{left-split}[\mathbf{X4}] \approx \mathbf{X2} \wedge \text{is-sorted}[\mathbf{X2}] \wedge \text{is-tuple}[\mathbf{X3}] \wedge \text{right-split}[\mathbf{X4}] \approx \mathbf{X3} \wedge \text{is-sorted}[\mathbf{X3}] \wedge \neg \text{is-trivial-tuple}[\mathbf{X4}] \Rightarrow \text{merged}[\mathbf{X2}, \mathbf{X3}] \approx \mathbf{X4})$

(Lemma (conjecture46): conjecture46)

$\forall_{\text{is-tuple}[\mathbf{x7}]} (\text{is-tuple}[\mathbf{X5}] \wedge \text{left-split}[\mathbf{X7}] \approx \mathbf{X5} \wedge \text{is-sorted}[\mathbf{X5}] \wedge \text{is-tuple}[\mathbf{X6}] \wedge \text{right-split}[\mathbf{X7}] \approx \mathbf{X6} \wedge \text{is-sorted}[\mathbf{X6}] \wedge \neg \text{is-trivial-tuple}[\mathbf{X7}] \Rightarrow \text{is-sorted}[\text{merged}[\mathbf{X5}, \mathbf{X6}]])$

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

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

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

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

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

We prove the individual conjunctive parts of (22):

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

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) $\text{is-sorted}[\langle \overline{X_0} \rangle]$:

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

Case 2:

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

Hence, we have to prove

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

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:

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

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

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

$$(25)$$

$$\text{is-tuple}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]]] \wedge \text{left-split}[\langle \overline{X_0} \rangle] \approx \text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]] \wedge \text{is-sorted}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]]]$$

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

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

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

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

We prove the individual conjunctive parts of (41):

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

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

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

We prove the individual conjunctive parts of (42):

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

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

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

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

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

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

$$(44)$$

$$\text{is-tuple}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]]] \wedge \text{left-split}[\langle \overline{X_0} \rangle] \approx \text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]] \wedge \text{is-sorted}[\text{sorted}[\text{left-split}[\langle \overline{X_0} \rangle]]] \wedge \text{is-tuple}[\text{sorted}[\text{right-split}[\langle \overline{X_0} \rangle]]] \wedge \text{right-split}[\langle \overline{X_0} \rangle] \approx \text{sorted}[\text{right-split}[\langle \overline{X_0} \rangle]] \wedge \text{is-sorted}[\text{sorted}[\text{right-split}[\langle \overline{X_0} \rangle]]] \wedge \neg \text{is-trivial-tuple}[\langle \overline{X_0} \rangle]$$

We prove the individual conjunctive parts of (44):

Proof of (44.1) `is-tuple[sorted[left-split[⟨X̄₀⟩]]]`:

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

Proof of (44.2) `left-split[⟨X̄₀⟩] ≈ sorted[left-split[⟨X̄₀⟩]]`:

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

Proof of (44.3) `is-sorted[sorted[left-split[⟨X̄₀⟩]]]`:

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

Proof of (44.4) `is-tuple[sorted[right-split[⟨X̄₀⟩]]]`:

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

Proof of (44.5) `right-split[⟨X̄₀⟩] ≈ sorted[right-split[⟨X̄₀⟩]]`:

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

Proof of (44.6) `is-sorted[sorted[right-split[⟨X̄₀⟩]]]`:

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

Proof of (44.7) `¬ is-trivial-tuple[⟨X̄₀⟩]`:

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

Proof of (41.3)

`is-sorted[merged[sorted[left-split[⟨X̄₀⟩]], sorted[right-split[⟨X̄₀⟩]]]`:

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

(52)

$$\begin{aligned} & \text{is-tuple}[\text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]]] \wedge \text{left-split}[\langle \bar{X}_0 \rangle] \approx \text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]] \wedge \\ & \text{is-sorted}[\text{sorted}[\text{left-split}[\langle \bar{X}_0 \rangle]]] \wedge \text{is-tuple}[\text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]] \wedge \\ & \text{right-split}[\langle \bar{X}_0 \rangle] \approx \text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]] \wedge \\ & \text{is-sorted}[\text{sorted}[\text{right-split}[\langle \bar{X}_0 \rangle]]] \wedge \neg \text{is-trivial-tuple}[\langle \bar{X}_0 \rangle] \end{aligned}$$

We prove the individual conjunctive parts of (52):

Proof of (52.1) `is-tuple[sorted[left-split[⟨X̄₀⟩]]]`:

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

Proof of (52.2) `left-split[⟨X̄₀⟩] ≈ sorted[left-split[⟨X̄₀⟩]]`:

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

Proof of (52.3) `is-sorted[sorted[left-split[⟨X̄₀⟩]]]`:

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

Proof of (52.4) `is-tuple[sorted[right-split[⟨X̄₀⟩]]]`:

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

Proof of (52.5) `right-split[⟨X̄₀⟩] ≈ sorted[right-split[⟨X̄₀⟩]]`:

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).

□