"Lazy Thinking": A Method for the Automated Invention of Algorithms

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

A Non-Trivial Algorithm: Gröbner-Bases

"Lazy Thinking"

Synthesis of Gröbner-Bases Algorithm

What Are Groebner Bases?

```
F = {x^2 y - 2 x z + 5 y - 3,
     x y^2 + x^2 + z ,
     x z - y^2 + 2 x - 1}
```

$\{-3+5\,y+x^2\,y-2\,x\,z,\ x^2+x\,y^2+z,\ -1+2\,x-y^2+x\,z\}$

```
GroebnerBasis[F]
```

$\{146302+448564\,z+502763\,z^2+242180\,z^3+39771\,z^4-6231\,z^5-2448\,z^6+168\,z^7+144\,z^8+16\,z^9,\ 104376175362406+1599126115499\,x+285345650746687\,z+259094430962640\,z^2+81019429651948\,z^3-562741124769\,z^4-4290216888948\,z^5-216539112184\,z^6+199291173968\,z^7+31903397104\,z^8,\ -29252096339198961+996255569955877\,y-79297437999899296\,z-73993371970407310\,z^2-24666034475337294\,z^3-250747610968661\,z^4+1288154187383705\,z^5+85610415996090\,z^6-58609022325772\,z^7-10267480080072\,z^8\}$

4 of 50

Application Example: "Algebraic Biology" 2007, RISC, S. Petrovic et al.

Gegeben:

Gesucht:

x, y, z, ...: Probabilities at the nodes of the "similarity tree".

5 of 50

Application Example: Software Reverse Engineering, 2007, RISC, T. Jebelean, D. Kapur, ...

Given: Programm.

Find: Specification.

Given: Program, specification.

Find: Loop invariants for the formal verification.

x, y, z, ...: the values of the program variables.

6 of 50

Application Example:
Break Cryptographic Codes,
2003, Paris VI, J.C. Faugere et al.

Given: Input - Output test examples

Find: the key, e.g.   011000101011011....11011101.

x, y, z, ...: the bits in the key.

Application Example: "Algebraic Oil", Shell 2005, RISC 2009

Given: Observations about oil flow in dependence on the position of the valves.

Find: The coefficients of a polynomials systems, that describes the behavior.

x, y, z, ...: the position of the valves.

Relevance of Groebner Bases

- Dozens of (difficult) problems on non-linear systems can be reduced to the construction of Groebner bases

  (~ 1000 papers, ~ 30 books, own AMS Classification number: 13P10).
- Some of these problems were open for many decades.
- Solution of these problems is possible for Groebner bases, because Groebner bases have some nice properties (canonicality, elimination, syzygy property).
- Therefore the construction of Groebner bases is an important problem.

9 of 50

Definition of Gröbner Bases (BB 1965)

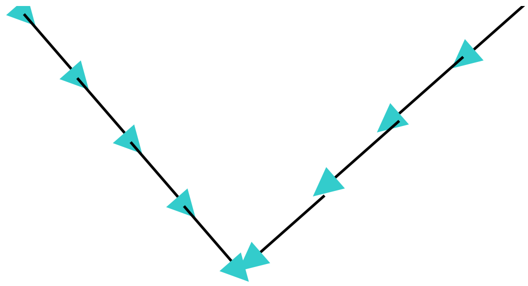
```
is-Gröbner-basis[G] ⇔ is-confluent[ →G ].
```

$\rightarrow_G$ ... a division step.

10 of XXX

Confluence of Division $\rightarrow_G$

```
is-confluent[→] :⇔ ∀_{f1,f2} (f1 ↔* f2 ⇒ f1↓* f2)
```



11 of XXX

Example of a Property of Gröbner Bases: Elimination Property

A Gröbner bases G (w.r.t. a lexicographic ordering) is "triangularized" (see the example at the beginning)!

This allows to obtain all the solutions of G by successive elimination.

$$F = \{x^2 y - 2xz + 5y - 3,\; x y^2 + x^2 + z,\; xz - y^2 + 2x - 1\}$$

```
GroebnerBasis[F]
```

$$\{146302 + 448564\, z + 502763\, z^2 + 242180\, z^3 + 39771\, z^4 - 6231\, z^5 - 2448\, z^6 + 168\, z^7 + 144\, z^8 + 16\, z^9,$$
$$104376175362406 + 1599126115499\, x + 285345650746687\, z + 259094430962640\, z^2 + 81019429651948\, z^3 - 562741124769\, z^4 - 4290216888948\, z^5 - 216539112184\, z^6 + 199291173968\, z^7 + 31903397104\, z^8,$$
$$-29252096339198961 + 996255569955877\, y - 79297437999899296\, z - 73993371970407310\, z^2 - 24666034475337294\, z^3 - 250747610968661\, z^4 + 1288154187383705\, z^5 + 85610415996090\, z^6 - 58609022325772\, z^7 - 10267480080072\, z^8\}$$

12 of 50

The Problem of Constructing Gröbner Bases

Find algorithm Gb such that

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

13 of 50

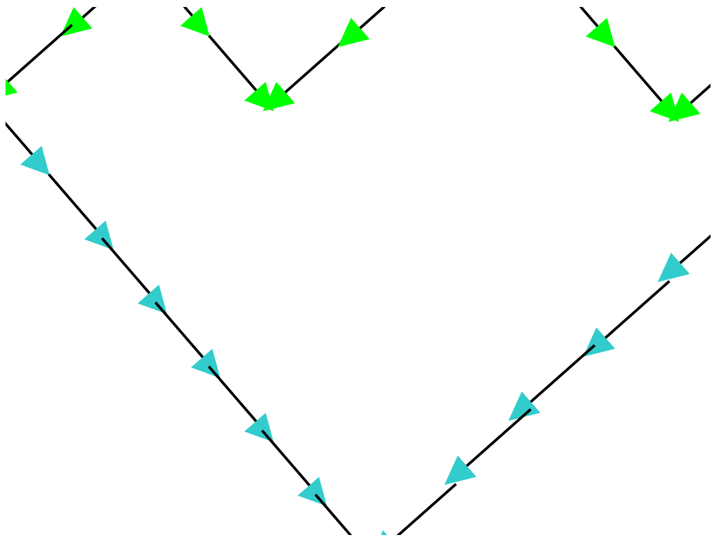
The "Main Theorem" of Algorithmic Gröbner Bases Theory (BB 1965):

F is a Gröbner basis $\Longleftrightarrow \underset{f_1, f_2 \in F}{\forall}$ remainder[F, S–polynomial[f_1, f_2]] = 0.

Proof: Nontrivial. Combinatorial.

The theorem reduces an infinite check to a finite check: Recall definition of "G is a Gröbner basis":

```
is-Gröbner-basis[G] ⇔ is-confluent[ →G ].
```



The power of the Gröbner bases method is contained in the invention of the notion of S-polynomial and the proof of the above theorem.

14 of 50

S-Polynomials

```
f1 = -2 y + x y
f2 = -x^2 + y^2
```

$-2\,y + x\,y$

$-x^2 + y^2$

```
S-polynomial[f1, f2] = y f1 - x f2
```

$y\,(-2\,y + x\,y) - x\,(-x^2 + y^2)$

```
S-polynomial[f1, f2] = y f1 - x f2    // Expand
```

$x^3 - 2\,y^2$

15 of 50

An Algorithm for *Constructing* Gröbner Bases (BB 1965)

Recall the main theorem:

F is a Gröbner basis $\Longleftrightarrow \underset{f_1, f_2 \in F}{\forall}$ remainder[F, S–polynomial[f_1, f_2]] = 0.

Based on the main theorem, the problem can be solved by the following algorithm:

Start with G:= F.

For any pair of polynomials f_1, $f_2 \in G$:

h := remainder[G, S–polynomial[f_1, f_2]]

If $h = 0$, consider the next pair.

If $h \neq 0$, add h to G and iterate.

The algorithm allows many refinements and variants which, however, are all based on the notion of S-polynomial and variants of the main theorem.

Correctness and Termination of the Algorithm

Correctness: Easy as soon as main theorem is available.

Termination: by Dickson's Lemma (Dickson 1913, BB 1970).

A sequence p_1, p_2, ... of power products with the property that, for all i < j, p_i does not divide p_j, must be finite.

Example

```
F = {x^2 y - 2 x z + 5 y - 3,
     x y^2 + x^2 + z,
     x z - y^2 + 2 x - 1}
```

$$\{-3+5y+x^2y-2xz,\ x^2+xy^2+z,\ -1+2x-y^2+xz\}$$

```
GroebnerBasis[F]
```

$$\{146\,302+448\,564\,z+502\,763\,z^2+242\,180\,z^3+39\,771\,z^4-6231\,z^5-2448\,z^6+168\,z^7+144\,z^8+16\,z^9,$$
$$104\,376\,175\,362\,406+1\,599\,126\,115\,499\,x+285\,345\,650\,746\,687\,z+259\,094\,430\,962\,640\,z^2+81\,019\,429\,651\,948\,z^3-562\,741\,124\,769\,z^4-4\,290\,216\,888\,948\,z^5-216\,539\,112\,184\,z^6+199\,291\,173\,968\,z^7+31\,903\,397\,104\,z^8,$$
$$-29\,252\,096\,339\,198\,961+996\,255\,569\,955\,877\,y-79\,297\,437\,999\,899\,296\,z-73\,993\,371\,970\,407\,310\,z^2-24\,666\,034\,475\,337\,294\,z^3-250\,747\,610\,968\,661\,z^4+1\,288\,154\,187\,383\,705\,z^5+85\,610\,415\,996\,090\,z^6-58\,609\,022\,325\,772\,z^7-10\,267\,480\,080\,072\,z^8\}$$

A Non-Trivial Algorithm: Gröbner-Bases

"Lazy Thinking"

Synthesis of Gröbner-Bases Algorithm

The "Lazy Thinking" Method

Is a method for the systematic invention of algorithms.

The method can be automated if suitable automated reasoners are available.

The Theorema system (BB et al. 1996 -) is a possible frame for the automation of the method.

Defining, Conjecturing, Proving, Programming, Computing in Theorema

Load Theorema

In[29]:=

```
Needs["Theorema`"];
```

Prove::shdw : Symbol Prove appears in multiple contexts {Theorema`Language`Semantics`UserLanguage`, Global`}; definitions in context Theorema`Language`Semantics`UserLanguage` may shadow or be shadowed by other definitions. ≫

Define and Conjecture

TS_In[30]:=

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

TS_In[31]:=

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

Prove

TS_In[32]:=

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

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

Automatically Generated Proof

Prove:

(Proposition (left zero): 0+) $\forall_{n} (0 + n = n)$,

under the assumptions:

(Definition (addition): +0:) $\forall_{m} (m + 0 = m)$,

(Definition (addition): + .:) $\forall_{m,n} (m + n^{+} = (m + n)^{+})$.

We prove (Proposition (left zero): 0+) by induction on n.

Induction Base:

(1) $0 + 0 = 0$.

A proof by simplification of (1) works.

Simplification of the lhs term:

$0 + 0 =$by (Definition (addition): +0:)

$0 \rfloor$

Simplification of the rhs term:

$0 \rfloor$

Induction Step:

Induction Hypothesis:

(2) $0 + n_1 = n_1$

Induction Conclusion:

(3) $0 + n_1^{+} = n_1^{+}$.

A proof by simplification of (3) works.

Simplification of the lhs term:

$0 + n_1{}^{+}$ =by (Definition (addition): + .:)

$(0 + n_1)^{+}$ =by (2)

$n_1{}^{+}$⌋

Simplification of the rhs term:

$n_1{}^{+}$⌋

□

Compute

TS_In[33]:=

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

TS_Out[33]=

$((((0^{+})^{+})^{+})^{+})^{+}$

21 of 50

Another Example of Defining, Conjecturing, Proving, ...

TS_In[34]:=

```
SetOptions[Prove, transformBy → ProofSimplifier,
  TransformerOptions → {branches → Proved}];
```

TS_In[35]:=

```
Definition["limit:", any[f, a],
    limit[f, a] ⟺ ∀_{ϵ, ϵ>0} ∃_N ∀_{n, n≥N} |f[n] - a| < ϵ]
```

General::spell1 :

New symbol name "limit" is similar to existing symbol "Limit" and may be misspelled. ≫

TS_In[36]:=

```
Proposition["limit of sum", any[f, a, g, b],
    (limit[f, a] ⋀ limit[g, b])  ⇒  limit[f + g, a + b]]
```

TS_In[37]:=

```
Definition["+:", any[f, g, x],
    (f + g)[x] = f[x] + g[x]]
```

TS_In[38]:=

```
Lemma["|+|", any[x, y, a, b, δ, ϵ],
    (|(x + y) - (a + b)| < (δ + ϵ))  ⟸  (|x - a| < δ ⋀ |y - b| < ϵ)]
```

TS_In[39]:=

```
Lemma["max", any[m, M1, M2],
    m ≥ max[M1, M2]  ⇒  (m ≥ M1 ⋀ m ≥ M2)]
```

General::spell1 :

New symbol name "max" is similar to existing symbol "Max" and may be misspelled.

≫

TS_In[40]:=

```
Theory["limit",
    Definition["limit:"]
    Definition["+:"]
    Lemma["|+|"]
    Lemma["max"]
]
```

TS_In[41]:=

```
Prove[Proposition["limit of sum"], using → Theory["limit"], by → PCS]
```

TS_Out[41]=

```
- ProofObject -
```

Proof contains interesting algorithmic and didactic information!

■ Automatically Generated Proof

Prove:

(Proposition (limit of sum)) $\underset{f,a,g,b}{\forall}\left(\mathrm{limit}[f, a] \wedge \mathrm{limit}[g, b] \Rightarrow \mathrm{limit}[f+g, a+b]\right)$,

under the assumptions:

(Definition (limit:)) $\underset{f,a}{\forall}\left(\mathrm{limit}[f, a] \Leftrightarrow \underset{\substack{\epsilon \\ \epsilon>0}}{\forall}\ \underset{N}{\exists}\ \underset{\substack{n \\ n\geq N}}{\forall}\left(|f[n]-a| < \epsilon\right)\right)$,

(Definition (+:)) $\underset{f,g,x}{\forall}\left((f+g)[x] = f[x] + g[x]\right)$,

(Lemma (|+|)) $\underset{x,y,a,b,\delta,\epsilon}{\forall}\left(|x+y-(a+b)| < \delta+\epsilon \Leftarrow (|x-a| < \delta \wedge |y-b| < \epsilon)\right)$,

(Lemma (max)) $\underset{m,M1,M2}{\forall}\left(m \geq \max[M1, M2] \Rightarrow m \geq M1 \wedge m \geq M2\right)$.

We assume

(1) $\mathrm{limit}[f_0, a_0] \wedge \mathrm{limit}[g_0, b_0]$,

and show

(2) $\text{limit}[f_0 + g_0, a_0 + b_0]$.

Formula (1.1), by (Definition (limit:)), implies:

(3) $\underset{\substack{\epsilon \\ \epsilon>0}}{\forall} \underset{N}{\exists} \underset{\substack{n \\ n\geq N}}{\forall} (|f_0[n] - a_0| < \epsilon)$.

By (3), we can take an appropriate Skolem function such that

(4) $\underset{\substack{\epsilon \\ \epsilon>0}}{\forall} \underset{\substack{n \\ n\geq N_0[\epsilon]}}{\forall} (|f_0[n] - a_0| < \epsilon)$,

Formula (1.2), by (Definition (limit:)), implies:

(5) $\underset{\substack{\epsilon \\ \epsilon>0}}{\forall} \underset{N}{\exists} \underset{\substack{n \\ n\geq N}}{\forall} (|g_0[n] - b_0| < \epsilon)$.

By (5), we can take an appropriate Skolem function such that

(6) $\underset{\substack{\epsilon \\ \epsilon>0}}{\forall} \underset{\substack{n \\ n\geq N_1[\epsilon]}}{\forall} (|g_0[n] - b_0| < \epsilon)$,

Formula (2), using (Definition (limit:)), is implied by:

(7) $\underset{\substack{\epsilon \\ \epsilon>0}}{\forall} \underset{N}{\exists} \underset{\substack{n \\ n\geq N}}{\forall} (|(f_0 + g_0)[n] - (a_0 + b_0)| < \epsilon)$.

We assume

(8) $\epsilon_0 > 0$,

and show

(9) $\underset{N}{\exists} \underset{\substack{n \\ n\geq N}}{\forall} (|(f_0 + g_0)[n] - (a_0 + b_0)| < \epsilon_0)$.

We have to find N^{***} such that

(10) $\underset{n}{\forall} (n \geq N^{***} \Rightarrow |(f_0 + g_0)[n] - (a_0 + b_0)| < \epsilon_0)$.

Formula (10), using (Definition (+:)), is implied by:

(11) $\underset{n}{\forall} (n \geq N^{***} \Rightarrow |f_0[n] + g_0[n] - (a_0 + b_0)| < \epsilon_0)$.

Formula (11), using (Lemma (|+|)), is implied by:

(12) $\underset{\substack{\delta,\epsilon \\ \delta+\epsilon=\epsilon_0}}{\exists} \underset{n}{\forall} (n \geq N^{***} \Rightarrow |f_0[n] - a_0| < \delta \bigwedge |g_0[n] - b_0| < \epsilon)$.

We have to find δ^*, ϵ^{**}, and N^{***} such that

(13) $(\delta^* + \epsilon^{**} = \epsilon_0) \bigwedge \underset{n}{\forall} (n \geq N^{***} \Rightarrow |f_0[n] - a_0| < \delta^* \bigwedge |g_0[n] - b_0| < \epsilon^{**})$.

Formula (13), using (6), is implied by:

$(\delta^* + \epsilon^{**} = \epsilon_0) \bigwedge \underset{n}{\forall} (n \geq N^{***} \Rightarrow \epsilon^{**} > 0 \bigwedge n \geq N_1[\epsilon^{**}] \bigwedge |f_0[n] - a_0| < \delta^*)$,

which, using (4), is implied by:

$(\delta^* + \epsilon^{**} = \epsilon_0) \bigwedge \underset{n}{\forall} (n \geq N^{***} \Rightarrow \delta^* > 0 \wedge \epsilon^{**} > 0 \wedge n \geq N_0[\delta^*] \wedge n \geq N_1[\epsilon^{**}])$,

which, using (Lemma (max)), is implied by:

(14) $(\delta^* + \epsilon^{**} = \epsilon_0) \bigwedge \underset{n}{\forall} (n \geq N^{***} \Rightarrow \delta^* > 0 \wedge \epsilon^{**} > 0 \wedge n \geq \max[N_0[\delta^*], N_1[\epsilon^{**}]])$.

Formula (14) is implied by

(15) $(\delta^* + \epsilon^{**} = \epsilon_0) \bigwedge \delta^* > 0 \bigwedge \epsilon^{**} > 0 \bigwedge \underset{n}{\forall} (n \geq N^{***} \Rightarrow n \geq \max[N_0[\delta^*], N_1[\epsilon^{**}]])$.

Partially solving it, formula (15) is implied by

(16) $(\delta^* + \epsilon^{**} = \epsilon_0) \wedge \delta^* > 0 \wedge \epsilon^{**} > 0 \wedge (N^{***} = \max[N_0[\delta^*], N_1[\epsilon^{**}]])$.

Now,

$(\delta^* + \epsilon^{**} = \epsilon_0) \wedge \delta^* > 0 \wedge \epsilon^{**} > 0$

can be solved for δ^* and ϵ^{**} by a call to Collins cad–method yielding a sample solution

$\delta^* \leftarrow \frac{\epsilon_0}{2}$,

$\epsilon^{**} \leftarrow \frac{\epsilon_0}{2}$.

Furthermore, we can immediately solve

$N^{***} = \max[N_0[\delta^*], N_1[\epsilon^{**}]]$

for N^{***} by taking

$N^{***} \leftarrow \max\left[N_0\left[\frac{\epsilon_0}{2}\right], N_1\left[\frac{\epsilon_0}{2}\right]\right]$.

Hence formula (16) is solved, and we are done.

□

The Algorithm Invention ("Synthesis") Problem

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

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

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]
 ....
```


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 (specification) P. Find: An algorithm A for P.

Overall Strategy of Lazy Thinking: (Automatically) reduce problem P to a couple of (hopefully simpler) problems Q, R, ...

until ...

Two Key Ideas of Lazy Thinking

Given: A problem (specification) P. Find: An algorithm A for P.

- ♣ (Understand the problem "completely": Specification P must be spelled out and "complete" knowledge must be available on the notions that occur in the specification P.)
- ♣ Consider known fundamental ideas of how to structure algorithms in terms of subalgorithms ("algorithm schemes A").

Try one scheme A after the other.

- ♣ For the chosen scheme A, try to prove $\underset{x}{\forall}$ P[x, A[x]]: From the failing proof construct specifications for the subalgorithms B occurring in A.

Example of an Algorithm Scheme ("Divide and Conquer"):

$$\underset{x}{\forall}\left(A[x] = \begin{cases} S[x] & \Leftarrow \text{is-trivial-tuple}[x] \\ M[A[L[x]], A[R[x]]] & \Leftarrow \text{otherwise} \end{cases}\right)$$

A is unknown algorithm.

S, M, L, R are unknown subalgorithms.

25 of 50

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.

26 of 50

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

Problem: Synthesize algorithm "sorted" such that

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

("Correctness Theorem")

Knowledge on the Problem:

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

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

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

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

etc. (approx. 20 formulae, see notebook of proofs in the Appendix.)

27 of 50

An Algorithm Scheme: Divide and Conquer

$$\underset{x}{\forall}\left(A[x] = \begin{cases} S[x] & \Leftarrow \textbf{is-trivial-tuple}[x] \\ M[A[L[x]], A[R[x]]] & \Leftarrow \textbf{otherwise} \end{cases}\right)$$

sorted is unknown algorithm.

S, M, L, R are unknown subalgorithms.

The only thing known is how the unknown algorithm sorted is composed from the 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.

28 of 50

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

$$\underset{x,\ldots,Y,\ldots}{\forall} (T[x, \ldots Y, \ldots] \Longrightarrow G[x, \ldots m[Y]]).$$

Details: see papers [Buchberger 2003] and example (in appendix).

29 of 50

The Result of Applying Lazy Thinking in the Sorting Example

Lazy Thinking, automatically (in approx. 1 minute 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 !

What do we have now: A problem reduction !

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

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

A Non-Trivial Algorithm: Gröbner-Bases

"Lazy Thinking"

Synthesis of Gröbner-Bases 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.

"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 Buchberger's Gröbner bases algorithm (1965) fully automatically: See [Buchberger 2005, Craciun 2008].

The Problem of Constructing Gröbner Bases

Find algorithm Gb such that

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

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

$\rightarrow_G$ ... a division step.

34 of XXX

Confluence of Division $\rightarrow_G$

$$\text{is-confluent}[\,\rightarrow\,] \;:\Leftrightarrow \underset{f1,f2}{\forall} (f1 \leftrightarrow^* f2 \Rightarrow f1 \downarrow^* f2)$$

35 of 50

Knowledge on the Concepts Involved

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

etc.

Algorithm Scheme "Critical Pair / Completion"

$$A[F] = A[F, \text{pairs}[F]]$$
$$A[F, \langle\rangle] = F$$

$$A[F, \langle\langle g1, g2\rangle, \bar{p}\rangle] =$$
$$\text{where}\Big[f = \text{lc}[g1, g2],\ h1 = \text{trd}[\text{rd}[f, g1], F],\ h2 = \text{trd}[\text{rd}[f, g2], F],$$
$$\begin{cases} A[F, \langle\bar{p}\rangle] & \Leftarrow\ h1 = h2 \\ A\Big[F \frown \text{df}[h1, h2],\ \langle\bar{p}\rangle \asymp \Big\langle \langle F_k, \text{df}[h1, h2]\rangle \underset{k=1,\dots,|F|}{|} \Big\rangle\Big] & \Leftarrow\ \text{otherwise} \end{cases}\Big]$$

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:

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

The Essential Problem

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

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

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

```
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.

(PhD thesis 2008 by my student A. Craciun.)

Details

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

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

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

$$\begin{array}{l} \text{is-finite}[G], \\ \text{ideal}[F] = \text{ideal}[G], \\ \text{is-Gröbner-basis}[G], \\ \quad \text{i.e. is-Church-Rosser}[\rightarrow_G]. \end{array}$$

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

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

Newman's lemma (1942):

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

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

$$f1 \downarrow^* f2 \Leftrightarrow \underset{g}{\exists} \left(\begin{cases} f1 \to^* g \\ f2 \to^* g \end{cases}\right)$$

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

$$lp[g1] \mid p,$$
$$f1 = rd[p, g1],$$
$$lp[g2] \mid p,$$
$$f2 = rd[p, g2].$$

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

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

where

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

42 of 50

Case h1=h2

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

(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,

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

Now we are stuck in the proof.

43 of 50

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

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

With such an lc, we then would have

```
p →g1 rd[p, g1] =
  a q rd[lc[g1, g2], g1] →G* a q trd[rd[lc[g1, g2], g1], G] =
   a q trd[rd[lc[g1, g2], g2], G] ←G* a q rd[lc[g1, g2], g2] =
    rd[p, g2] ←g2 p
```

and, hence,

```
f1 →G* a q trd[rd[lc[g1, g2], g1], G],
```

```
f2 →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

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

and the requirements:

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

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

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!

Case h1≠h2

In this case, df[h1,h2]∈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

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

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 satifies (df requirement), namely

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

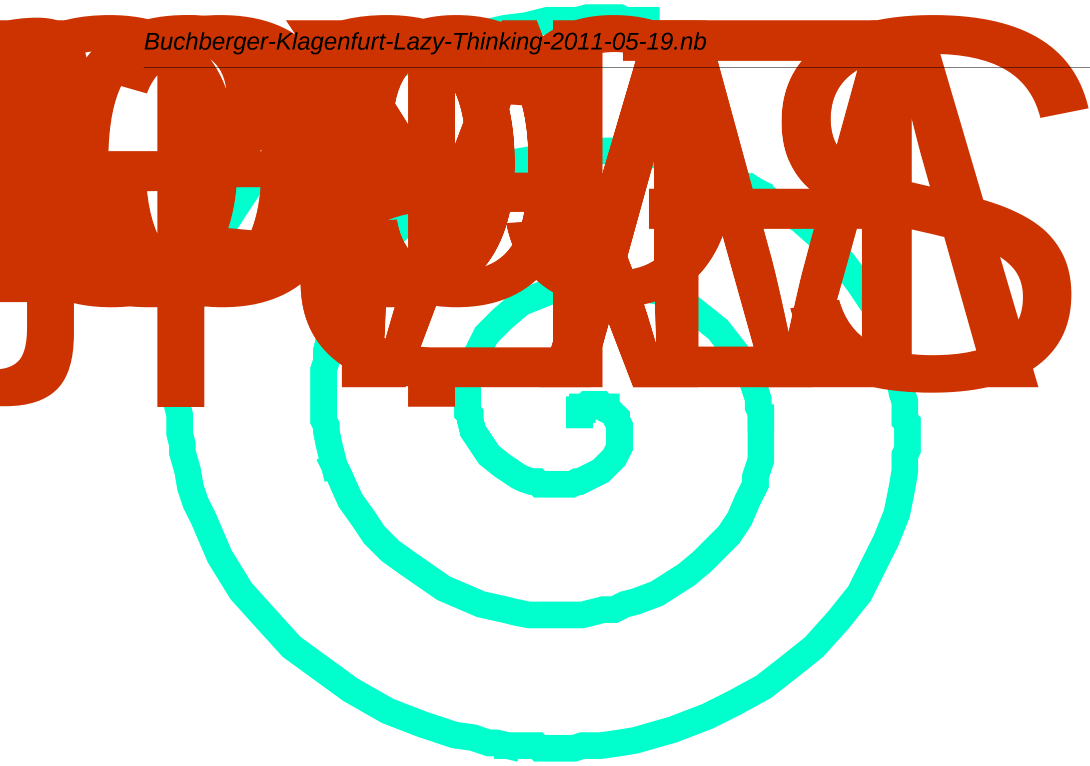
because, in fact,

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

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

Conclusion

Automation of mathematical reasoning ("formal methods") is in the center of the technology spiral:



References

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.

[Craciun 2008]

A. Craciun.

The Implementation of Buchberger's Lazy Thinking Method for Automated Algorithm Synthesis in Theorema.

PhD Thesis, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, April 2008.

Appendix: The Proofs Generated During the Automated Synthesis of the Merge-Sort Algorithm

First Proof Attempt

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, $\bar{z}$⟩]),

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

(Definition (is permuted version): 2) $\underset{y,\bar{y}}{\forall}$ (⟨⟩ ≉ ⟨y, $\bar{y}$⟩),

(Definition (is permuted version): 3) $\underset{x,\bar{x},\bar{y}}{\forall}$ (⟨$\bar{y}$⟩ ≈ ⟨x, $\bar{x}$⟩ ⇔ x ∈ ⟨$\bar{y}$⟩ ⋀ dfo[x, ⟨$\bar{y}$⟩] ≈ ⟨$\bar{x}$⟩),

(Definition (is sorted version))

$\underset{\substack{X,Y \\ \text{is-tuple}[X]}}{\forall}$ (is-sorted-version[X, Y] ⇔ is-tuple[Y] ⋀ $X \approx Y$ ⋀ is-sorted[Y]),

(Proposition (is tuple tuple)) $\underset{\bar{x}}{\forall}$ is-tuple[⟨$\bar{x}$⟩],

(Definition (prepend): ⌣) $\underset{x,\bar{y}}{\forall}$ (x ⌣ ⟨$\bar{y}$⟩ = ⟨x, $\bar{y}$⟩),

(Proposition (singleton tuple is singleton tuple)) $\underset{x}{\forall}$ is-singleton-tuple[⟨x⟩],

(Definition (is trivial tuple))

$\underset{\text{is-tuple}[X]}{\forall}$ (is-trivial-tuple[X] ⇔ is-empty-tuple[X] ⋁ is-singleton-tuple[X]),

(Definition (is element): 1) $\underset{x}{\forall}$ (x ∉ ⟨⟩),

(Definition (is element): 2) $\underset{x,y,\bar{y}}{\forall}$ (x ∈ ⟨y, $\bar{y}$⟩ ⇔ ($x = y$) ⋁ x ∈ ⟨$\bar{y}$⟩), (Definition (deletion of the first occurrence): 1) $\underset{a}{\forall}$ (dfo[a, ⟨⟩] = ⟨⟩),

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

$\underset{a,x,\bar{x}}{\forall}$ (dfo[a, ⟨x, $\bar{x}$⟩] = ‖⟨$\bar{x}$⟩ ⇐ x = a, x ⌣ dfo[a, ⟨$\bar{x}$⟩] ⇐ otherwise‖),

(Definition (is longer than): 1) $\underset{\bar{y}}{\forall}$ (⟨⟩ ≯ ⟨$\bar{y}$⟩),

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

(Definition (is longer than): 3) $\underset{x,\bar{x},y,\bar{y}}{\forall}$ (⟨x, $\bar{x}$⟩ > ⟨y, $\bar{y}$⟩ ⇔ ⟨$\bar{x}$⟩ > ⟨$\bar{y}$⟩),

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

(Proposition (only trivial tuple permuted version of itself)) $\underset{\substack{\bar{x},Y \\ \text{is-trivial-tuple}[\langle \bar{x} \rangle]}}{\forall}$ ((Y = ⟨$\bar{x}$⟩) ⇒ Y ≈ ⟨$\bar{x}$⟩),

(Proposition (reflexivity of permuted version)) $\underset{\bar{x}}{\forall}$ (⟨$\bar{x}$⟩ ≈ ⟨$\bar{x}$⟩),

(Algorithm (sorted))

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

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

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

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

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

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

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

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

Well–founded induction:

Assume:

(1) is-tuple[⟨$\bar{X}_0$⟩].

Well–Founded Induction Hypothesis:

(2) $\underset{\text{is-tuple}[x1]}{\forall} \left(\langle \overline{X_0} \rangle > x1 \Rightarrow \text{is-sorted-version}[x1, \text{sorted}[x1]] \right)$

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

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).

□

Second Proof Attempt (with Specifications of Subalgorithms Extractd from First Proof Attempt)

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[$\langle\rangle$],

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

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

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

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

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

(Definition (is sorted version))

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

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

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

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

(Definition (is trivial tuple))

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

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

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

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

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

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

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

(Definition (is longer than): 2) $\underset{x,\bar{x}}{\forall}$ ($\langle x, \bar{x}\rangle > \langle\rangle$), (Definition (is longer than): 3) $\underset{x,\overline{x},y,\overline{y}}{\forall}\left(\langle x, \overline{x}\rangle > \langle y, \overline{y}\rangle \Leftrightarrow \langle\overline{x}\rangle > \langle\overline{y}\rangle\right)$,

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

(Proposition (only trivial tuple permuted version of itself)) $\underset{\substack{\overline{x},Y \\ \text{is-trivial-tuple}[\langle\overline{x}\rangle]}}{\forall}\left((Y = \langle\overline{x}\rangle) \Rightarrow Y \approx \langle\overline{x}\rangle\right)$,

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

(Algorithm (sorted))

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

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

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

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

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

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

(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[$X1$] ⇒ (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[$\langle\overline{X_0}\rangle$].

Well–Founded Induction Hypothesis:

(2) $\underset{\text{is-tuple}[x2]}{\forall}\left(\langle\overline{X_0}\rangle > x2 \Rightarrow \text{is-sorted-version}[x2, \text{sorted}[x2]]\right)$

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

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

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

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

Proof of (41.4)
$\text{is-sorted}[\text{merged}[\text{sorted}[\text{left-split}[\langle\overline{X_0}\rangle]], \text{sorted}[\text{right-split}[\langle\overline{X_0}\rangle]]]]$:

Pending proof of (41.4).

□

■ Third Proof Attempt (with Specifications of Subalgorithms Extractd from Second Proof Attempt)

Prove:

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

under the assumptions:

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

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

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

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

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

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

(Definition (is sorted version))

$\underset{\text{is-tuple}[X]}{\forall_{X,Y}}\ (\text{is-sorted-version}[X, Y] \Leftrightarrow \text{is-tuple}[Y] \bigwedge X \approx Y \bigwedge \text{is-sorted}[Y])$,

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

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

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

(Definition (is trivial tuple))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(Lemma (conjecture15): conjecture15)

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

(Lemma (conjecture44): conjecture44)

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

We try to 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) $\underset{\text{is-tuple}[x3]}{\forall} \left(\langle \overline{X_0} \rangle \succ x3 \Rightarrow \text{is-sorted-version}[x3, \text{sorted}[x3]]\right)$

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

Hence, we have to prove

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

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

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

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

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

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

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

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

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

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

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

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

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

We prove the individual conjunctive parts of (22):

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

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

Case 2:

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

Hence, we have to prove

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

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

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[⟨$\overline{X_0}$⟩]]]]

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

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

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.)

□

■ Successful Proof (with Specifications of Subalgorithms Extractd from Third Proof Attempt)

Prove:

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

under the assumptions:

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

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

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

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

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

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

(Definition (is sorted version))

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

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

(Definition (prepend): ⌣) $\underset{x,\overline{y}}{\forall}\ (x \smile \langle \overline{y} \rangle = \langle x, \overline{y} \rangle)$,

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

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

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

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

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

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

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

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

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

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

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

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

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

(Algorithm (sorted))

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

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

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

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

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

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

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

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

(Lemma (conjecture44): conjecture44)

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

(Lemma (conjecture46): conjecture46)

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

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

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

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

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

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

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

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

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

We prove the individual conjunctive parts of (22):

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

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

Case 2:

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

Hence, we have to prove

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

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

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

(25) is-tuple[sorted[left-split[$\langle \overline{X_0} \rangle$]]] $\wedge$
left-split[$\langle \overline{X_0} \rangle$] $\approx$ sorted[left-split[$\langle \overline{X_0} \rangle$]] $\wedge$
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[$\langle \overline{X_0} \rangle$]]] $\wedge$
right-split[$\langle \overline{X_0} \rangle$] $\approx$ sorted[right-split[$\langle \overline{X_0} \rangle$]] $\wedge$
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[⟨$\overline{X_0}$⟩]], 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[$\langle \overline{X_0} \rangle$]], sorted[right-split[$\langle \overline{X_0} \rangle$]]]]:

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

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

We prove the individual conjunctive parts of (52):

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

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

Proof of (52.2) left-split[$\langle \overline{X_0} \rangle$] $\approx$ sorted[left-split[$\langle \overline{X_0} \rangle$]]:

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

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

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

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

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

Proof of (52.5) right-split[$\langle \overline{X_0} \rangle$] $\approx$ sorted[right-split[$\langle \overline{X_0} \rangle$]]:

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

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

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

Proof of (52.7) $\neg$ is-trivial-tuple[$\langle \overline{X_0} \rangle$]:

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

□