Mathematical Invention: How Much Can Be Automated?

Bruno Buchberger
RISC

Contents

Groebner Bases

Input

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

GroebnerBasis[F]
```

$$\{-659 - 816\,z + 2692\,z^2 - 2652\,z^3 - 5022\,z^4 - 2248\,z^5 - 468\,z^6 - 44\,z^7 + z^8,$$
$$-483\,149\,651 + 1\,414\,811\,712\,y + 3\,202\,658\,389\,z - 3\,143\,902\,747\,z^2 - 3\,943\,389\,703\,z^3 - 1\,505\,657\,905\,z^4 -$$
$$286\,983\,089\,z^5 - 21\,721\,697\,z^6 + 515\,939\,z^7,\ 137\,904\,771 + 707\,405\,856\,x - 481\,941\,395\,z -$$
$$1\,722\,559\,419\,z^2 - 1\,183\,994\,965\,z^3 - 338\,037\,939\,z^4 - 52\,491\,205\,z^5 - 2\,450\,869\,z^6 + 66\,413\,z^7\}$$

Theorem: F solvable iff Groebner basis of F ≠ {1}.

Groebner Bases **Algorithm** (Buchberger 1965): Form S-polynomials

z (x y^2 - 2 x z + 3 x^2 + z - y + 2) - y^2 (z x + 2 x y + x + z + 1), ...

Proof: difficult; problem was open for over 60 years!

Two essentials: invention of S-polynomials
verification of correctness

Automated Geo Proofs by Groebner Bases (Buchberger, Kutzler, Kapur, Robu et al. 1986 - ...)

Reduction of the Problem to Gröbner bases computation:

Geo Theorem $\longrightarrow$ (by coordinatization)

$$\forall_{x,y,\ldots} \; (\text{poly1}(x,y,\ldots) = 0 \wedge \ldots \Rightarrow \text{poly}(x,y,\ldots) = 0) \longrightarrow$$

$$\neg \exists_{x,y,\ldots} (\text{poly1}(x,y,\ldots) = 0 \wedge \ldots \wedge \text{poly}(x,y,\ldots) \neq 0) \longrightarrow$$

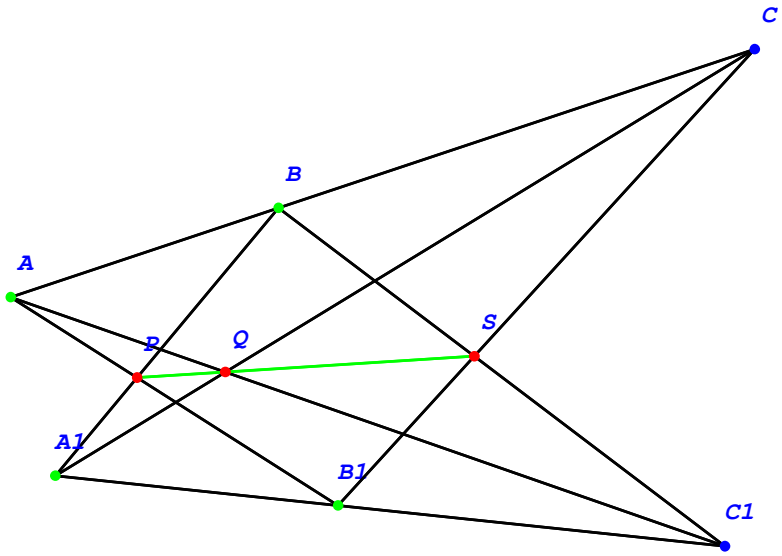
$$\neg \exists_{x,y,\ldots,a} (\text{poly1}(x,y,\ldots)=0 \wedge \ldots \wedge a \cdot \text{poly}(x,y,\ldots) - 1 = 0).$$

The latter question can be decided by the Gröbner basis method!

The method is implemented in the Theorema System (Buchberger et al, 1996 - ...).

An alternative to Wu's method for automated geo proving.

Example: Pappus Theorem



Automated Proof in the **Theorema** system:

To transform the geometric problem into algebraic form we have to choose first an orthogonal coordinate system.

Let's have the origin in point A, and the points $\{B, C\}$ on the y-axis

Using this coordinate system we have the following points:

$$\{\{A, 0, 0\}, \{B, 0, u_1\}, \{A1, u_2, u_3\}, \{B1, u_4, u_5\},\ \{C, 0, u_6\}, \{C1, u_7, x_1\}, \{P, x_2, x_3\}, \{Q, x_4, x_5\}, \{S, x_6, x_7\}\}$$

The algebraic form of the assertion is:

(1)

$$\forall_{x_1, x_2, x_3, x_4, x_5, x_6, x_7} (u_3 u_4 + -u_2 u_5 + -u_3 u_7 + u_5 u_7 + u_2 x_1 + -u_4 x_1 == 0 \bigwedge$$
$$u_5 x_2 + -u_4 x_3 == 0 \bigwedge -u_1 u_2 + u_1 x_2 + -u_3 x_2 + u_2 x_3 == 0 \bigwedge x_1 x_4 + -u_7 x_5 == 0 \bigwedge$$
$$-u_2 u_6 + -u_3 x_4 + u_6 x_4 + u_2 x_5 == 0 \bigwedge u_1 u_7 + -u_1 x_6 + x_1 x_6 + -u_7 x_7 == 0 \bigwedge$$
$$-u_4 u_6 + -u_5 x_6 + u_6 x_6 + u_4 x_7 == 0 \Rightarrow x_3 x_4 + -x_2 x_5 + -x_3 x_6 + x_5 x_6 + x_2 x_7 + -x_4 x_7 == 0)$$

This problem is equivalent to:

(2)

$$\neg\left(\exists_{x_1,x_2,x_3,x_4,x_5,x_6,x_7}\ (u_3\,u_4 + -u_2\,u_5 + -u_3\,u_7 + u_5\,u_7 + u_2\,x_1 + -u_4\,x_1 == 0 \bigwedge\right.$$
$$u_5\,x_2 + -u_4\,x_3 == 0 \bigwedge -u_1\,u_2 + u_1\,x_2 + -u_3\,x_2 + u_2\,x_3 == 0 \bigwedge$$
$$x_1\,x_4 + -u_7\,x_5 == 0 \bigwedge -u_2\,u_6 + -u_3\,x_4 + u_6\,x_4 + u_2\,x_5 == 0 \bigwedge$$
$$u_1\,u_7 + -u_1\,x_6 + x_1\,x_6 + -u_7\,x_7 == 0 \bigwedge -u_4\,u_6 + -u_5\,x_6 + u_6\,x_6 + u_4\,x_7 == 0 \bigwedge$$
$$\left.x_3\,x_4 + -x_2\,x_5 + -x_3\,x_6 + x_5\,x_6 + x_2\,x_7 + -x_4\,x_7 \neq 0\right)\Big)$$

To remove the last inequality, we use the Rabinowitsch trick: Let v_0 be a new variable. Then the problem becomes:

(3)

$$\neg\left(\exists_{x_1,x_2,x_3,x_4,x_5,x_6,x_7,v_0}\ (u_3\,u_4 + -u_2\,u_5 + -u_3\,u_7 + u_5\,u_7 + u_2\,x_1 + -u_4\,x_1 == 0 \bigwedge\right.$$
$$u_5\,x_2 + -u_4\,x_3 == 0 \bigwedge -u_1\,u_2 + u_1\,x_2 + -u_3\,x_2 + u_2\,x_3 == 0 \bigwedge$$
$$x_1\,x_4 + -u_7\,x_5 == 0 \bigwedge -u_2\,u_6 + -u_3\,x_4 + u_6\,x_4 + u_2\,x_5 == 0 \bigwedge$$
$$u_1\,u_7 + -u_1\,x_6 + x_1\,x_6 + -u_7\,x_7 == 0 \bigwedge -u_4\,u_6 + -u_5\,x_6 + u_6\,x_6 + u_4\,x_7 == 0 \bigwedge$$
$$\left.1 + -v_0\,(x_3\,x_4 + -x_2\,x_5 + -x_3\,x_6 + x_5\,x_6 + x_2\,x_7 + -x_4\,x_7) == 0\right)\Big)$$

This statement is true iff the corresponding Gröbner basis is {1}.

The Gröbner bases is `{1}`.

Hence, the statement and the original assertion is true.

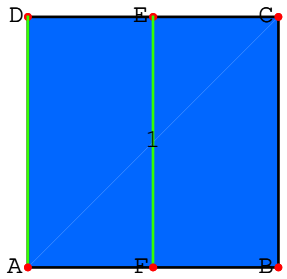
Statistics:

Time needed to compute the Gröbner bases: `0.42 Seconds`.

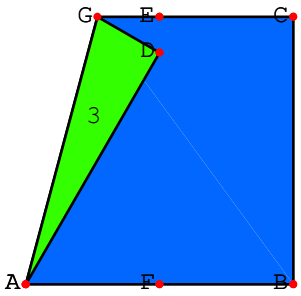
Example: The Correctness of Origami Constructions (Ida, Buchberger, Robu et al. 2003, 2004):

Starting from a square A, B, C, D, find a sequence of origami steps such that, finally, we arrive at an equilateral triangle.

Let E and F be the midpoint of $\overline{DC}$ and $\overline{AB}$, respectively.



Then we fix the point A and fold so that point D will lie on line EF. (This is a legal origami operation.)



Now we can do the analogous step with corner C, fixing B and bringing C onto the current position of D.

Then the triangle ADB is an equilateral triangle with edge length $\overline{AB}$. We could add a few easy origami operations that would result in hiding the areas that extend over the triangle ADB but we do not show these easy steps because we would like now to pose a simple proving problem:

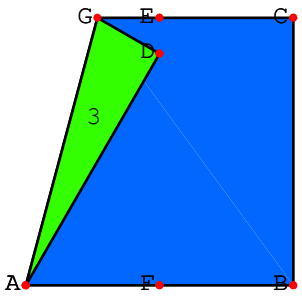
An Origami Proof Problem: Prove that, for all squares ABCD, $\overline{GD} = 2\,\overline{ED}$.

The Translation into a Prove Problem on Equalities:

First, note that $\overline{AB} = \overline{BC} = \overline{CD} = \overline{DA}$, since we start from a square. Hence, whenever the length of one of these four edges occurs, we replace it by $\overline{AB}$.

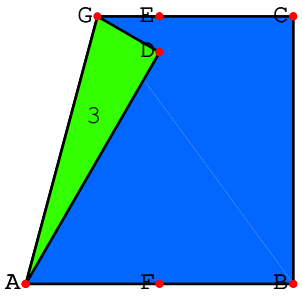
Now observe that

$$\overline{DF}^2 = \overline{AD}^2 - \overline{AF}^2 = \overline{AB}^2 - \left(\overline{AB}/2\right)^2 = 3/4\,\overline{AB}^2.$$



and

$$\overline{GD}^2 = \overline{GE}^2 + \overline{ED}^2 = \left(\overline{DC}/2 - \overline{GD}\right)^2 + \left(\overline{EF} - \overline{DF}\right)^2 = \left(\overline{AB}/2 - \overline{GD}\right)^2 + \left(\overline{AB} - \overline{DF}\right)^2.$$



We want to decide whether, under these assumptions,

$$\overline{GD} = 2\,\overline{ED} = 2\left(\overline{EF} - \overline{DF}\right) = 2\left(\overline{AB} - \overline{DF}\right).$$

For abbreviation, let's write

$$a = \overline{AB},\ b = \overline{GD},\ f = \overline{DF}.$$

Then, what we want to prove is that

$$\forall_{a\neq 0,f,b}\left(\left\{\begin{array}{l} f^2 = 3/4\,a^2 \\ b^2 = (a/2-b)^2 + (a-f)^2 \end{array}\right. \Rightarrow (b = 2\,(a-f))\right)$$

The Transformation to a Groebner Basis Construction Problem:

This is equivalent to

$$\neg\,\exists_{a,f,b}\left(\left\{\begin{array}{l} a \neq 0 \\ f^2 = 3/4\,a^2 \\ b^2 = (a/2-b)^2 + (a-f)^2 \\ b \neq 2\,(a-f) \end{array}\right.\right),$$

which is equivalent to

$$\neg \underset{a,f,b,\xi,\eta}{\exists} \left(\begin{cases} a\,\eta = 1 \\ f^2 = 3/4\,a^2 \\ b^2 = (a/2 - b)^2 + (a - f)^2 \\ (b - 2\,(a - f))\,\xi = 1 \end{cases} \right),$$

This question can be decided by computing the (reduced) Gröbner basis

```
J =
 GroebnerBasis[{a η - 1, f^2 - 3 / 4 a^2, b^2 - (a / 2 - b)^2 - (a - f)^2, (b - 2 (a - f)) ξ - 1}, {ξ, η, b, f, a}]
```

```
{1}
```

and to check whether or not this Gröbner basis is equal to {1}. Since this is the case, we know that the restricted version of the theorem is true.

Construction of a Regular Heptagon (Proved with Groebner Bases Method by T. Ida, 2004)

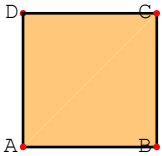


Fig. 1

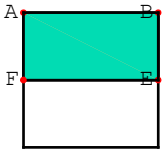


Fig. 2

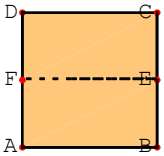


Fig. 3

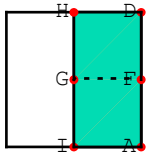


Fig. 4

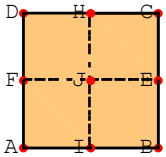


Fig. 5

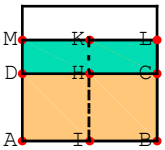


Fig. 6

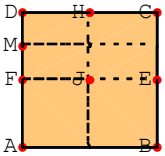


Fig. 7

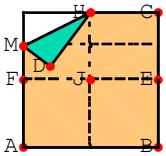


Fig. 8

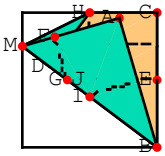


Fig. 9

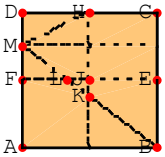


Fig. 10

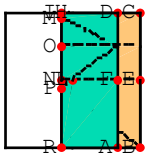


Fig. 11

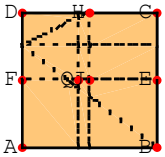


Fig. 12

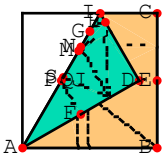


Fig. 13

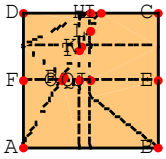


Fig. 14

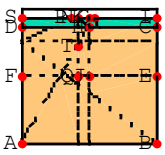


Fig. 15

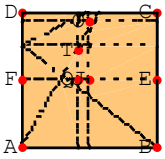


Fig. 16

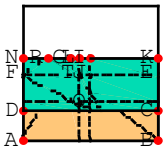


Fig. 17

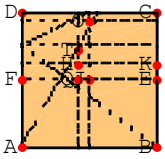


Fig. 18

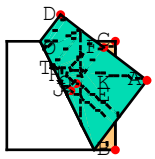


Fig. 19

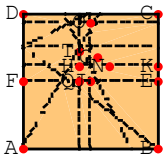


Fig. 20

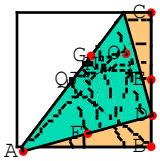


Fig. 21

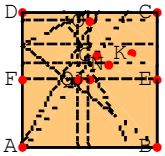


Fig. 22

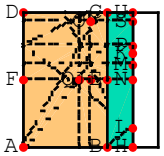


Fig. 23

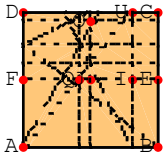


Fig. 24

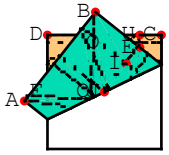


Fig. 25

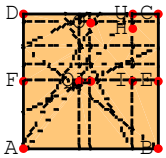


Fig. 26

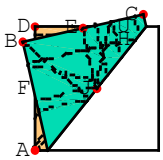


Fig. 27

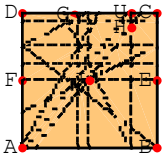


Fig. 28

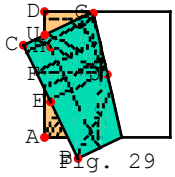


Fig. 29

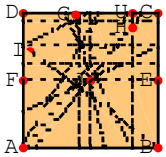


Fig. 30

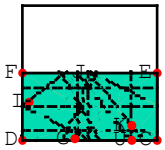


Fig. 31

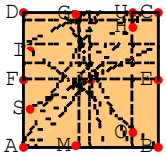


Fig. 32

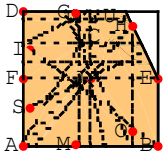


Fig. 33

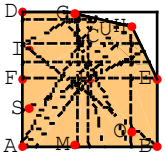


Fig. 34

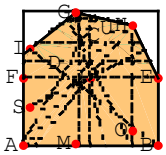


Fig. 35

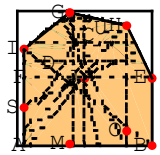


Fig. 36

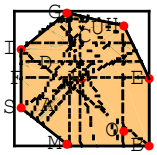


Fig. 37

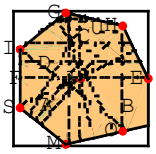


Fig. 38

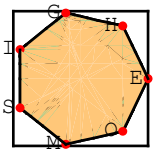
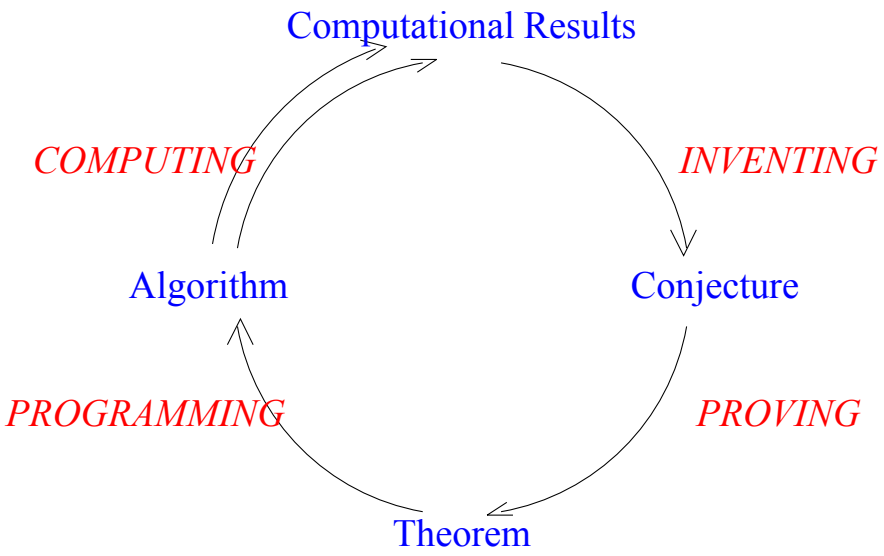


Fig. 39

Contents

The Creativity Spiral



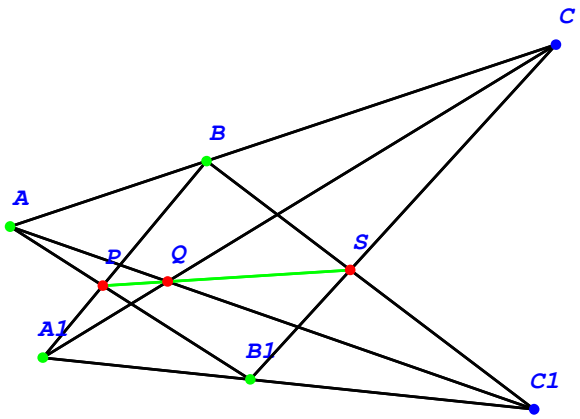
Invent by observing many examples.

Near-at hand automation of invention: automate the generation of examples.

Dynamic Geometry: Automate Geo Observation.

Automated Invention of Geo Theorems (Chou, DM Wang, Robu, ...)

1. Systematically (automatically) **generate geo "configuations.** Example of a configuration:



2. For each configuration: **observe for "many" parameter values** what happens with the conclusion in the configuration (dramatically facilitated by **dynamic geometry** software). This may lead to a "conjecture".

3. Automatically **prove / disprove the conjecture**. (By Wu's method, Groebner bases method, area method, coordinate free methods, ...)

Automated Invention of Side Conditions for Geo Theorems (Wu, Kapur, Winkler, ...)

1. Often, geo theorems are not true "for all" instances of a configuration but only for the "non-degenerate" cases.

2. In Wu's method and Groebner bases method, it is possible to find "non-degenerate conditions" (under which the theorems become true) automatically. (In Groebner bases method: by analyzing the polynomial reduction process.)

Contents

Example: The Theorema PCS ("Prove Compute Simplify") Prover (Buchberger et al. 2000 ...)

An automated proving method for "alternating quantifiers" (e.g. elementary analysis)

Initialization

Example:

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

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

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

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

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

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

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

- ProofObject -
```

Note that this automatically generated proof contains an automatically generated algorithm! (See last line of the proof the expression for the index bound N***.)

Automatically Generated Proof with Algorithm Included

Prove:

(Proposition (limit of sum)) $\underset{f,a,g,b}{\forall}\ (\mathtt{limit}[f,\ a] \bigwedge \mathtt{limit}[g,\ b] \Rightarrow \mathtt{limit}[f+g,\ a+b])$,

under the assumptions:

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

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

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

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

We assume

(1) $\mathtt{limit}[f_0,\ a_0] \bigwedge \mathtt{limit}[g_0,\ b_0]$,

and show

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

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

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

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

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

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

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

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

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

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

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

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 $\mathbb{N}^{"***"}$ such that

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

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

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

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

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

We have to find $\delta^{"*"}$, $\epsilon^{"**"}$, and $\mathbb{N}^{"***"}$ such that

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

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

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

which, using (4), is implied by:

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

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

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

Formula (14) is implied by

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

Partially solving it, formula (15) is implied by

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

Now,

$\left(\delta^{"*"} + \epsilon^{"**"} = \epsilon_0\right) \bigwedge \delta^{"*"} > 0 \bigwedge \epsilon^{"**"} > 0$

can be solved for $\delta^{"*"}$ and $\epsilon^{"**"}$ 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

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

for $\mathbb{N}^{"***"}$ by taking

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

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

□

Contents

How Can a Theorem Like lim[f⊕g] = lim[f] + lim[g] be invented?

By applying "formulae schemes". (Buchberger 2002 ...).

A formula:

```
L[P[f, g]] = p[L[f], L[g]]
```

A quantified formula:

$$\forall_{f,g} \left(L[P[f, g]] = p[L[f], L[g]]\right).$$

A scheme:

$$\forall_{L,P,p} \left(\text{is-homomorphic}[L, P, p] \Leftrightarrow \forall_{f,g} \left(L[P[f, g]] = p[L[f], L[g]]\right)\right).$$

For new operations like lim, invent axioms, definitions, propositions, problems, algorithms by applying schemes: For example,

$$\text{is-homomorphic}\left[\text{lim}, \oplus, +\right].$$

Application of a scheme = the (higher order) predicate logic operation of substituting concrete operations for the operation variables in the scheme.

Observation: there are not so many schemes!

Examples of Schemes:

Can be used for inventing axioms, propositions, problems, algorithms:

$$\forall_{p,r} \Big(\text{right-unit}[p, r] \Leftrightarrow \forall_{x,y} \; (p[x, r] = x) \Big)$$

$$\forall_{p,s} \Big(\text{left-successor-recursion}[p, s] \Leftrightarrow \forall_{x,y} \; (p[s[x],] = s[p[x, y]]) \Big)$$

$$\forall_{+,s} \Big(\text{right-successor-recursion}[p, s] \Leftrightarrow \forall_{x,y} \; (p[x, \; s[y]] = s[p[x, y]]) \Big)$$

Examples of Schemes:

Can be used for inventing axioms, propositions, problems, algorithms:

$$\forall_{I,P,A} \left(\text{explicit-problem}[I, P, A] \Leftrightarrow \forall_{x \in I[x]} \left(P[x, A[x]] \right) \right)$$

Examples of Schemes:

Can be used for inventing axioms, propositions, problems, algorithms:

$$\text{Divide-and-Conquer}[A, S, M, L, R] \Leftrightarrow$$

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

Examples of Schemes:

Can be used for inventing axioms, propositions, problems, algorithms:

$$\forall_{A,lc,df}\ \text{critical-pair-completion}[A, lc, df] \iff$$

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

$$A[F, \langle\langle g1, g2\rangle, \overline{p}\rangle] =$$
$$\text{where}\left[f = lc[g1, g2],\right.$$
$$\forall_{F,g1,g2,\overline{p}}\quad h1 = \text{trd}[\text{rd}[f, g1], F],$$
$$h2 = \text{trd}[\text{rd}[f, g2], F],$$
$$\left\{\begin{array}{ll} A[F, \langle\overline{p}\rangle] & \Leftarrow\ h1 = h2 \\ A\left[F \frown df[h1, h2], \langle\overline{p}\rangle \asymp \left\langle \langle F_k, df[h1, h2]\rangle \mid_{k=1,\ldots,|F|} \right\rangle\right] & \Leftarrow\ \text{otherwise} \end{array}\right.\Bigg]$$

Example: Invent Bottom-up by "Formulae Schemes"

Initialization

Inductive Definition of Addition:

Application of scheme right–unit to '+' and '0' and application of scheme right–successor–recursion to '+' and '$^+$' yields:

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

Invent a First Theorem:

Application of scheme right–unit to '+' and ' 0' yields:

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

Now we can use our automated Theorema induction prover:

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

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

Invent a Second Theorem:

Application of scheme left–successor–recursion to '+' and '$^{+}$' yields:

```
Proposition["left induction", any[m, n],
    m⁺ + n = (m + n)⁺ ". +"]

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

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

Invent a Third Theorem:

```
Proposition["commutativity of addition", any[m, n],
    m + n = n + m " + = "]

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

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

An Experiment (Craciun 2008):

Using a library of formulae schemes, he (automatically) built up the inductive theory of natural numbers until the prime factorization theorem.

Contents

What Happens if we Go from the Axioms for + Straight to Commutativity?

Commutativity

```
Proposition["commutativity of addition", any[m, n],
    m + n = n + m  " + = "]

Prove[Proposition["commutativity of addition"],
    using → ⟨Definition["addition"]⟩,
    by → NNEqIndProver,

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

Analyze the Reason for the Failing Proof

Guess Conjectures that Would Make the Proof Possible

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

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

Automation of Failure Analysis and Guessing Conjectures (Buchberger 1998, 2000, 2004, ...)

```
Prove[Proposition["commutativity of addition"],
    using → Definition["addition"], by → Cascade[NNEqIndProver, ConjectureGenerator],
  ProverOptions → {TermOrder → LeftToRight}];
```

The Algorithm Invention ("Synthesis") Problem

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

```
∀ P[x, A[x]].
 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" (the "Cascade") (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, ...

Two Key Ideas of Lazy Thinking for Algorithm Synthesis

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

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

A is unknown algorithm.

S, M, L, R are unknown subalgorithms.

The only thing known is how the unknown algorithm sorted is composed from the unknown algorithms S, M, L, R.

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 algorithm "sorted" such that

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

("Correctness Theorem")

Knowledge on the Problem:

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

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

$$\forall_{x} \; \mathbf{is\text{-}sorted[\langle x\rangle]}$$

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

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

An Algorithm Scheme: Divide and Conquer

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

A is unknown algorithm. In our case: 'sorted'.

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.

This proof will fail! Why?

We now analyze the failing proof: see notebooks with failing proofs.

Automated Invention of Sufficient Specifications for the Subalgorithms

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

Collect temporary assumptions T[x0, ... A [...], ...]
and temporary goals G[x0, ...m [A [...]]]

and produces specification

$$\forall_{x,\ldots,Y,\ldots} \left(T[x,\ldots Y,\ldots] \Rightarrow G[x,\ldots m\,[Y]]\right).$$

Details: see papers [Buchberger 2003] and example below.

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):

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

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

$$\forall_{x}\ (L[x] \asymp R[x] \approx x)$$

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

What do we have now: A problem reduction !

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

First Proof Attempt

Prove:

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

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

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

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

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

(Definition (is sorted version))

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

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

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

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

(Definition (is trivial tuple))

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

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

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

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

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

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

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

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

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

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

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

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

(Algorithm (sorted)) $\forall_{\text{is-tuple}[X]} \left(\text{sorted}[X] = \| \text{special}[X] \Leftarrow \text{is-trivial-tuple}[X], \right.$

$\left. \text{merged}[\text{sorted}[\text{left-split}[X]], \text{sorted}[\text{right-split}[X]]] \Leftarrow \text{otherwise} \|\right)$

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

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

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

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

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

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

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

Well–founded induction:

Assume:

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

Well–Founded Induction Hypothesis:

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

We have to show:

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

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

Hence, we have to prove

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

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

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

Formula (1), by (Lemma (Closure of Special)), implies:

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

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

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

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

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

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

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

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

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

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

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

Pending proof of (13.4).

Case 2:

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

Hence, we have to prove

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

Pending proof of (8).

□

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

Prove:

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

under the assumptions:

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

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

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

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

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

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

(Definition (is sorted version))

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

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

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

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

(Definition (is trivial tuple))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(Lemma (conjecture15): conjecture15) $\forall_{\substack{X1 \\ \text{is-tuple}[X1]}} (\text{is-trivial-tuple}[X1] \Rightarrow (\text{special}[X1] = X1))$.

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

Well–Founded Induction Hypothesis:

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

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

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

We prove the individual conjunctive parts of (22):

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

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

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

Formula (22.2) is true by (10).

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

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

Case 2:

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

Hence, we have to prove

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

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

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

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

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

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

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

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

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

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

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

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

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

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

We prove the individual conjunctive parts of (42):

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

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

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

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

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

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

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

Pending proof of (41.4).

□

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

Prove:

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

under the assumptions:

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

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

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

(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) $\underset{x,\overline{x},\overline{y}}{\forall} (\langle \overline{y} \rangle \approx \langle x, \overline{x} \rangle \Leftrightarrow x \in \langle \overline{y} \rangle \bigwedge \texttt{dfo}[x, \langle \overline{y} \rangle] \approx \langle \overline{x} \rangle)$,

(Definition (is sorted version))

$\underset{\texttt{is-tuple}[x]}{\underset{x,y}{\forall}} (\texttt{is-sorted-version}[x, y] \Leftrightarrow \texttt{is-tuple}[y] \bigwedge x \approx y \bigwedge \texttt{is-sorted}[y])$,

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

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

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

(Definition (is trivial tuple))

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

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

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

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

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

$$\forall_{a,x,\overline{x}} \left(\mathrm{dfo}[a, \langle x, \overline{x}\rangle] = \| \langle \overline{x}\rangle \Leftarrow x = a,\ x - \mathrm{dfo}[a, \langle \overline{x}\rangle] \Leftarrow \mathrm{otherwise}\|\right),$$

(Definition (is longer than): 1) $\forall_{\overline{y}} \left(\langle\rangle \not> \langle \overline{y}\rangle\right),$

(Definition (is longer than): 2) $\forall_{x,\overline{x}} \left(\langle x, \overline{x}\rangle > \langle\rangle\right),$

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

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

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

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

(Algorithm (sorted)) $\forall_{\mathrm{is\text{-}tuple}[X]} \left(\mathrm{sorted}[X] = \|\mathrm{special}[X] \Leftarrow \mathrm{is\text{-}trivial\text{-}tuple}[X],\right.$

$$\left.\mathrm{merged}[\mathrm{sorted}[\mathrm{left\text{-}split}[X]], \mathrm{sorted}[\mathrm{right\text{-}split}[X]]] \Leftarrow \mathrm{otherwise}\|\right)$$

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

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

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

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

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

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

(Lemma (conjecture15): conjecture15)

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

(Lemma (conjecture44): conjecture44)

$$\forall_{X2,X3,X4, \mathrm{is\text{-}tuple}[X4]} \left(\mathrm{is\text{-}tuple}[X2] \wedge \mathrm{left\text{-}split}[X4] \approx X2 \wedge \mathrm{is\text{-}sorted}[X2] \wedge \mathrm{is\text{-}tuple}[X3] \wedge \right.$$

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

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

Well–founded induction:

Assume:

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

Well–Founded Induction Hypothesis:

(2) $\forall_{\texttt{is-tuple}[\boldsymbol{x3}]} \left(\langle \overline{X_0} \rangle \succ \boldsymbol{x3} \Rightarrow \texttt{is-sorted-version}[\boldsymbol{x3}, \texttt{sorted}[\boldsymbol{x3}]]\right)$

We have to show:

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

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

Hence, we have to prove

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

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

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

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

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

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

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

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

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

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

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

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

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

We prove the individual conjunctive parts of (22):

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

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

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

Formula (22.2) is true by (10).

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

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

Case 2:

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

Hence, we have to prove

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

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

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

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

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

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

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

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

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

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

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

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

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

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

We prove the individual conjunctive parts of (42):

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

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

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

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

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

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

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

We prove the individual conjunctive parts of (44):

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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)) $\underset{\texttt{is-tuple}[\boldsymbol{X}]}{\forall} \texttt{is-sorted-version}[\boldsymbol{X}, \texttt{sorted}[\boldsymbol{X}]]$,

under the assumptions:

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

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

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

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

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

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

(Definition (is sorted version))

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

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

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

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

(Definition (is trivial tuple))

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

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

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

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

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

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

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

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

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

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

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

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

(Algorithm (sorted)) $\forall_{\text{is-tuple}[X]} (\text{sorted}[X] = \| \text{special}[X] \Leftarrow \text{is-trivial-tuple}[X],$,

$$\text{merged}[\text{sorted}[\text{left-split}[X]], \text{sorted}[\text{right-split}[X]]] \Leftarrow \text{otherwise}\|)$$

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

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

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

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

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

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

(Lemma (conjecture15): conjecture15)

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

(Lemma (conjecture44): conjecture44)

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

(Lemma (conjecture46): conjecture46)

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

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

Well–founded induction:

Assume:

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

Well–Founded Induction Hypothesis:

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

We have to show:

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

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

Case 1:

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

Hence, we have to prove

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

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

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

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

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

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

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

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

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

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

We prove the individual conjunctive parts of (41):

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

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

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

We prove the individual conjunctive parts of (42):

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

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

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

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

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

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

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

We prove the individual conjunctive parts of (44):

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

We prove the individual conjunctive parts of (52):

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

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

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

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

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

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

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

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

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

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

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

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

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

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

□

Example: Synthesis of Insertion-Sort

Synthesize A such that

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

Algorithm Scheme: "simple recursion"

$$A[\langle\rangle] = c$$

$$\forall_{x} \; A[\langle x \rangle] = s[\langle x \rangle]$$

$$\forall_{x,\overline{y}} \; (A[\langle x, \overline{y} \rangle] = i[x, A[\langle \overline{y} \rangle]])$$

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

$$c = \langle\rangle$$

$$\forall_{x} \; (s[\langle x \rangle] = \langle x \rangle)$$

$$\forall_{x,\overline{y}} \left(\text{is-sorted}[\langle \overline{y} \rangle] \Rightarrow \begin{array}{l} \text{is-sorted}[i[x, \langle \overline{y} \rangle]] \\ i[\langle x, \overline{y} \rangle] \approx (x \frown \langle \overline{y} \rangle) \end{array} \right)$$

Contents

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

the correctness proof is delivered as a side-product.

The Problem of Constructing Gröbner Bases

Find algorithm Gb such that

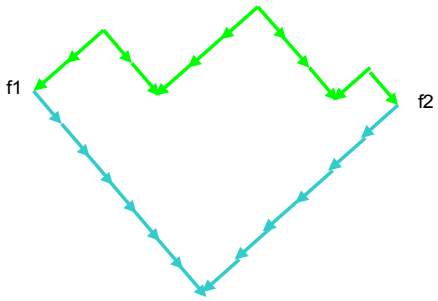
$$\forall_{\texttt{is-finite[F]}} \begin{pmatrix} \texttt{is-finite[Gb[F]]} \\ \texttt{is-Gröbner-basis[Gb[F]]} \\ \texttt{ideal[F] = ideal[Gb[F]].} \end{pmatrix}$$

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

$\rightarrow_G$ ... a division step.

Confluence of Division $\rightarrow_G$

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



Knowledge on the Concepts Involved

$$h1 \to_G h2 \Rightarrow p \cdot h1 \to_G p \cdot h2$$

etc.

Algorithm Scheme "Critical Pair / Completion"

$$A[F] = A[F, pairs[F]]$$
$$A[F, \langle\rangle] = F$$

$$A[F, \langle\langle g1, g2\rangle, \overline{p}\rangle] =$$
$$\text{where}\left[f = lc[g1, g2],\right.$$
$$h1 = trd[rd[f, g1], F],$$
$$h2 = trd[rd[f, g2], F],$$
$$\left\{\begin{array}{ll} A[F, \langle\overline{p}\rangle] & \Leftarrow\ h1 = h2 \\ A\left[F \frown df[h1, h2], \langle\overline{p}\rangle \asymp \left\langle \langle F_k, df[h1, h2]\rangle \;\middle|\; {\scriptstyle k=1,\ldots,|F|} \right\rangle\right] & \Leftarrow\ \text{otherwise} \end{array}\right.\Bigg]$$

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

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

The Essential Problem

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

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

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

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

and

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

and prove that the idea is correct.

Now Start the (Automated) Correctness Proof

With current theorem proving technology, in the *Theorema* system (and other provers), the proof attempt 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

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

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

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

Newman's lemma (1942):

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

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

$$f1 \downarrow^* f2 \ \Leftrightarrow\ \exists_{g}\ \left(\left\{\begin{array}{l} f1 \rightarrow^* g \\ f2 \rightarrow^* g \end{array}\right.\right)$$

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

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

where

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

Case h1=h2

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

(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,

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

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

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

With such an lc, we then would have

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

and, hence,

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

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

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

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

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

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

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,

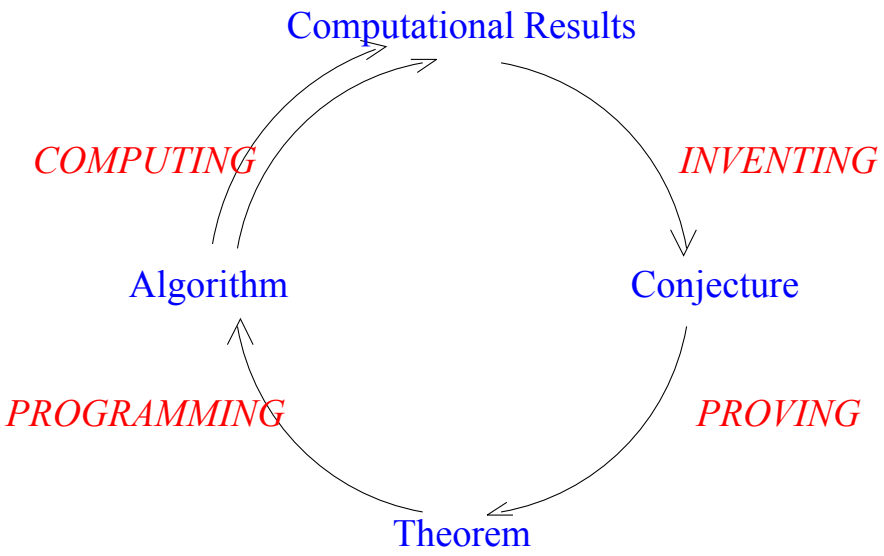
$$\forall_{f,g} \left(f \downarrow_{\{f-g\}} {}^{*} g \right).$$

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

Conclusions

Invention and verification are the two important aspects of mathematical exploration.

Both, the automation of invention and verification hinge on the automation of reasoning.



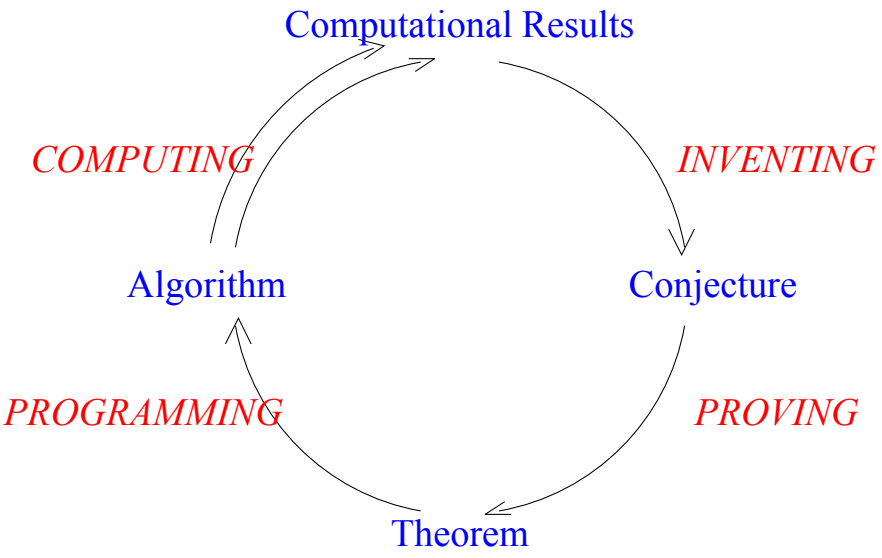
Algorithmic algebra is a basis for the automation of reasoning.

Automated reasoning can produce algebra.

Progress happens in rounds of "self-application" going to higher and higher levels.

Each round is a big challenge.

There is no upper bound to automation and sophistication (Gödel !)


Computational Results
COMPUTING
INVENTING
Algorithm
Conjecture
PROGRAMMING
PROVING
Theorem

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.