

Undecidability Results on Orienting Single Rewrite Rules*

René Thiemann, Fabian Mitterwallner, and Aart Middeldorp

University of Innsbruck

May 14, 2024

Abstract

We formalize several undecidability results on termination for *one-rule* term rewrite systems by means of simple reductions from Hilbert’s 10th problem. To be more precise, for a class C of reduction orders, we consider the question for a given rewrite rule $\ell \rightarrow r$, whether there is some reduction order $\succ \in C$ such that $\ell \succ r$. We include undecidability results for each of the following classes C :

- the class of *linear* polynomial interpretations over the natural numbers,
- the class of linear polynomial interpretations over the natural numbers in the *weakly monotone* setting,
- the class of Knuth–Bendix orders with *subterm coefficients*,
- the class of *non-linear* polynomial interpretations over the natural numbers, and
- the class of non-linear polynomial interpretations over the *rational* and *real* numbers.

Contents

1	Introduction	2
2	Preliminaries: Extending the Library on Multivariate Polynomials	2
2.1	Part 1 – Extensions Without Importing Univariate Polynomials	2
2.2	Part 2 – Extensions With Importing Univariate Polynomials	8
3	Definition of Monotone Algebras and Polynomial Interpretations	14

*This research was supported by the Austrian Science Fund (FWF) project I 5943.

4	Hilbert's 10th Problem to Linear Inequality	18
5	Undecidability of Linear Polynomial Termination	20
6	Undecidability of KBO with Subterm Coefficients	29
7	Undecidability of Polynomial Termination over Integers	32
8	Undecidability of Polynomial Termination using δ -Orders	39

1 Introduction

The main part of this paper is about one of the earliest termination methods for term rewrite systems: using a polynomial interpretation over the natural numbers, which goes back to Lankford [1].

In a recent paper [3] it was shown that this and other related techniques are undecidable, even for one-rule rewrite systems. This AFP entry formally proves the results in [3]. These are all based on reduction from a variant of Hilbert's 10th problem, which was shown to be undecidable by Matiyasevich [2].

2 Preliminaries: Extending the Library on Multivariate Polynomials

2.1 Part 1 – Extensions Without Importing Univariate Polynomials

```

theory Preliminaries-on-Polynomials-1
  imports
    Polynomials.More-MPoly-Type
    Polynomials.MPoly-Type-Class-FMap
  begin

  type-synonym var = nat
  type-synonym monom = var  $\Rightarrow_0$  nat

  definition substitute :: (var  $\Rightarrow$  'a mpoly)  $\Rightarrow$  'a :: comm-semiring-1 mpoly  $\Rightarrow$  'a
    mpoly where
      substitute  $\sigma$  p = insertion  $\sigma$  (replace-coeff Const p)

  lemma Const-0: Const 0 = 0
    <proof>

  lemma Const-1: Const 1 = 1
    <proof>

```

lemma *insertion-Var*: $\text{insertion } \alpha (\text{Var } x) = \alpha x$
<proof>

lemma *insertion-Const*: $\text{insertion } \alpha (\text{Const } a) = a$
<proof>

lemma *insertion-power*: $\text{insertion } \alpha (p^{\wedge}n) = (\text{insertion } \alpha p)^{\wedge}n$
<proof>

lemma *insertion-monom-add*: $\text{insertion } \alpha (\text{monom } (f + g) a) = \text{insertion } \alpha (\text{monom } f 1) * \text{insertion } \alpha (\text{monom } g a)$
<proof>

lemma *insertion-uminus*: $\text{insertion } \alpha (- p) = - \text{insertion } \alpha p$
<proof>

lemma *insertion-sum-list*: $\text{insertion } \alpha (\text{sum-list } ps) = \text{sum-list } (\text{map } (\text{insertion } \alpha) ps)$
<proof>

lemma *coeff-uminus*: $\text{coeff } (- p) m = - \text{coeff } p m$
<proof>

lemma *insertion-substitute*: $\text{insertion } \alpha (\text{substitute } \sigma p) = \text{insertion } (\lambda x. \text{insertion } \alpha (\sigma x)) p$
<proof>

lemma *Const-add*: $\text{Const } (x + y) = \text{Const } x + \text{Const } y$
<proof>

lemma *substitute-add[simp]*: $\text{substitute } \sigma (p + q) = \text{substitute } \sigma p + \text{substitute } \sigma q$
<proof>

lemma *Const-sum*: $\text{Const } (\text{sum } f A) = \text{sum } (\text{Const } o f) A$
<proof>

lemma *Const-sum-list*: $\text{Const } (\text{sum-list } (\text{map } f xs)) = \text{sum-list } (\text{map } (\text{Const } o f) xs)$
<proof>

lemma *Const-0-eq[simp]*: $\text{Const } x = 0 \iff x = 0$
<proof>

lemma *Const-sum-any*: $\text{Const } (\text{Sum-any } f) = \text{Sum-any } (\text{Const } o f)$
<proof>

lemma *Const-mult*: $\text{Const } (x * y) = \text{Const } x * \text{Const } y$
<proof>

lemma *Const-power*: $\text{Const } (x \hat{=} e) = \text{Const } x \hat{=} e$
<proof>

lemma *lookup-replace-Const*: $\text{lookup } (\text{mapping-of } (\text{replace-coeff } \text{Const } p)) \ l = \text{Const } (\text{lookup } (\text{mapping-of } p) \ l)$
<proof>

lemma *replace-coeff-mult*: $\text{replace-coeff } \text{Const } (p * q) = \text{replace-coeff } \text{Const } p * \text{replace-coeff } \text{Const } q$
<proof>

lemma *substitute-mult[simp]*: $\text{substitute } \sigma (p * q) = \text{substitute } \sigma p * \text{substitute } \sigma q$
<proof>

lemma *replace-coeff-Var[simp]*: $\text{replace-coeff } \text{Const } (\text{Var } x) = \text{Var } x$
<proof>

lemma *replace-coeff-Const[simp]*: $\text{replace-coeff } \text{Const } (\text{Const } c) = \text{Const } (\text{Const } c)$
<proof>

lemma *substitute-Var[simp]*: $\text{substitute } \sigma (\text{Var } x) = \sigma x$
<proof>

lemma *substitute-Const[simp]*: $\text{substitute } \sigma (\text{Const } c) = \text{Const } c$
<proof>

lemma *substitute-0[simp]*: $\text{substitute } \sigma \ 0 = 0$
<proof>

lemma *substitute-1[simp]*: $\text{substitute } \sigma \ 1 = 1$
<proof>

lemma *substitute-power[simp]*: $\text{substitute } \sigma (p \hat{=} e) = (\text{substitute } \sigma p) \hat{=} e$
<proof>

lemma *substitute-monom[simp]*: $\text{substitute } \sigma (\text{monom } (\text{monomial } e \ x) \ c) = \text{Const } c * (\sigma x) \hat{=} e$
<proof>

lemma *substitute-sum-list*: $\text{substitute } \sigma (\text{sum-list } (\text{map } f \ xs)) = \text{sum-list } (\text{map } (\text{substitute } \sigma \ o \ f) \ xs)$
<proof>

lemma *substitute-sum*: $\text{substitute } \sigma (\text{sum } f \ xs) = \text{sum } (\text{substitute } \sigma \ o \ f) \ xs$
<proof>

lemma *substitute-prod*: $\text{substitute } \sigma (\text{prod } f \text{ } xs) = \text{prod } (\text{substitute } \sigma \text{ } o f) \text{ } xs$
<proof>

definition *vars-list where* $\text{vars-list} = \text{sorted-list-of-set } o \text{ } \text{vars}$

lemma *set-vars-list[simp]*: $\text{set } (\text{vars-list } p) = \text{vars } p$
<proof>

lift-definition *mpoly-coeff-filter* :: $('a :: \text{zero} \Rightarrow \text{bool}) \Rightarrow 'a \text{ } \text{mpoly} \Rightarrow 'a \text{ } \text{mpoly}$ **is**
 $\lambda f p. \text{Poly-Mapping.mapp } (\lambda m c. c \text{ when } f c) p$ *<proof>*

lemma *mpoly-coeff-filter*: $\text{coeff } (\text{mpoly-coeff-filter } f p) m = (\text{coeff } p m \text{ when } f (\text{coeff } p m))$
<proof>

lemma *total-degree-add*: **assumes** $\text{total-degree } p \leq d \text{ } \text{total-degree } q \leq d$
shows $\text{total-degree } (p + q) \leq d$
<proof>

lemma *total-degree-Var[simp]*: $\text{total-degree } (\text{Var } x :: 'a :: \text{comm-semiring-1 } \text{mpoly}) = \text{Suc } 0$
<proof>

lemma *total-degree-Const[simp]*: $\text{total-degree } (\text{Const } x) = 0$
<proof>

lemma *total-degree-Const-mult*: **assumes** $\text{total-degree } p \leq d$
shows $\text{total-degree } (\text{Const } x * p) \leq d$
<proof>

lemma *vars-0[simp]*: $\text{vars } 0 = \{\}$
<proof>

lemma *vars-1[simp]*: $\text{vars } 1 = \{\}$
<proof>

lemma *vars-Var[simp]*: $\text{vars } (\text{Var } x :: 'a :: \text{comm-semiring-1 } \text{mpoly}) = \{x\}$
<proof>

lemma *vars-Const[simp]*: $\text{vars } (\text{Const } c) = \{\}$
<proof>

lemma *coeff-sum-list*: $\text{coeff } (\text{sum-list } ps) m = (\sum p \leftarrow ps. \text{coeff } p m)$
<proof>

lemma *coeff-Const-mult*: $\text{coeff } (\text{Const } c * p) m = c * \text{coeff } p m$
<proof>

lemma *coeff-Const*: $\text{coeff } (\text{Const } c) m = (\text{if } m = 0 \text{ then } (c :: 'a :: \text{comm-semiring-1}) \text{ else } 0)$

<proof>

lemma *coeff-Var*: $\text{coeff } (\text{Var } x) m = (\text{if } m = \text{monomial } 1 \ x \text{ then } 1 :: 'a :: \text{comm-semiring-1} \text{ else } 0)$

<proof>

list-based representations, so that polynomials can be converted to first-order terms

lift-definition *monom-list* :: $'a :: \text{comm-semiring-1} \text{ mpoly} \Rightarrow (\text{monom} \times 'a) \text{ list}$
is $\lambda p. \text{map } (\lambda m. (m, \text{lookup } p \ m)) (\text{sorted-list-of-set } (\text{keys } p))$ *<proof>*

lift-definition *var-list* :: $\text{monom} \Rightarrow (\text{var} \times \text{nat}) \text{ list}$

is $\lambda m. \text{map } (\lambda x. (x, \text{lookup } m \ x)) (\text{sorted-list-of-set } (\text{keys } m))$ *<proof>*

lemma *monom-list*: $p = (\sum (m, c) \leftarrow \text{monom-list } p. \text{monom } m \ c)$
<proof>

lemma *monom-list-coeff*: $(m, c) \in \text{set } (\text{monom-list } p) \Longrightarrow \text{coeff } p \ m = c$
<proof>

lemma *monom-list-keys*: $(m, c) \in \text{set } (\text{monom-list } p) \Longrightarrow \text{keys } m \subseteq \text{vars } p$
<proof>

lemma *var-list*: $\text{monom } m \ c = \text{Const } (c :: 'a :: \text{comm-semiring-1}) * (\prod (x, e) \leftarrow \text{var-list } m. (\text{Var } x) \ \hat{=} e)$
<proof>

lemma *var-list-keys*: $(x, e) \in \text{set } (\text{var-list } m) \Longrightarrow x \in \text{keys } m$
<proof>

lemma *vars-substitute*: **assumes** $\bigwedge x. \text{vars } (\sigma \ x) \subseteq V$
shows $\text{vars } (\text{substitute } \sigma \ p) \subseteq V$
<proof>

lemma *insertion-monom-nonneg*: **assumes** $\bigwedge x. \alpha \ x \geq 0$ **and** $c: (c :: 'a :: \{\text{linordered-nonzero-semiring}, \text{ordered-semiring-0}\}) \geq 0$
shows $\text{insertion } \alpha \ (\text{monom } m \ c) \geq 0$
<proof>

lemma *insertion-nonneg*: **assumes** $\bigwedge x. \alpha \ x \geq (0 :: 'a :: \text{linordered-idom})$
and $\bigwedge m. \text{coeff } p \ m \geq 0$
shows $\text{insertion } \alpha \ p \geq 0$
<proof>

lemma *vars-sumlist*: $\text{vars } (\text{sum-list } ps) \subseteq \bigcup (\text{vars } \text{' set } ps)$
<proof>

lemma *coefficients-of-linear-poly*: **assumes** *linear*: *total-degree* ($p :: 'a :: \text{comm-semiring-1 mpoly}$) ≤ 1
shows $\exists c \ a \ vs. \ p = \text{Const } c + (\sum i \leftarrow vs. \ \text{Const } (a \ i) * \text{Var } i)$
 $\wedge \text{distinct } vs \wedge \text{set } vs = \text{vars } p \wedge \text{sorted-list-of-set } (\text{vars } p) = vs \wedge (\forall v \in \text{set } vs. \ a \ v \neq 0)$
 $\wedge (\forall i. \ a \ i = \text{coeff } p \ (\text{monomial } 1 \ i)) \wedge (c = \text{coeff } p \ 0)$
 $\langle \text{proof} \rangle$

Introduce notion for degree of monom

definition *degree-monom* :: $(\text{var} \Rightarrow_0 \text{nat}) \Rightarrow \text{nat}$ **where**
degree-monom $m = \text{sum } (\text{lookup } m) \ (\text{keys } m)$

lemma *total-degree-alt-def*: *total-degree* $p = \text{Max } (\text{insert } 0 \ (\text{degree-monom } ' \ \text{keys } (\text{mapping-of } p)))$
 $\langle \text{proof} \rangle$

lemma *degree-monom-le-total-degree*: **assumes** $\text{coeff } p \ m \neq 0$
shows *degree-monom* $m \leq \text{total-degree } p$
 $\langle \text{proof} \rangle$

lemma *degree-monom-eq-total-degree*: **assumes** $p \neq 0$
shows $\exists m. \ \text{coeff } p \ m \neq 0 \wedge \text{degree-monom } m = \text{total-degree } p$
 $\langle \text{proof} \rangle$

lemma *degree-add-leI*: *degree* $p \ x \leq d \implies \text{degree } q \ x \leq d \implies \text{degree } (p + q) \ x \leq d$
 $\langle \text{proof} \rangle$

lemma *degree-sum-leI*: **assumes** $\bigwedge i. \ i \in A \implies \text{degree } (p \ i) \ x \leq d$
shows *degree* $(\text{sum } p \ A) \ x \leq d$
 $\langle \text{proof} \rangle$

lemma *total-degree-sum-leI*: **assumes** $\bigwedge i. \ i \in A \implies \text{total-degree } (p \ i) \leq d$
shows *total-degree* $(\text{sum } p \ A) \leq d$
 $\langle \text{proof} \rangle$

lemma *total-degree-monom*: **assumes** $c \neq 0$
shows *total-degree* $(\text{monom } m \ c) = \text{degree-monom } m$
 $\langle \text{proof} \rangle$

lemma *degree-Var[simp]*: *degree* $(\text{Var } x :: 'a :: \text{comm-semiring-1 mpoly}) \ x = 1$
 $\langle \text{proof} \rangle$

lemma *Var-neq-0[simp]*: *Var* $x \neq (0 :: 'a :: \text{comm-semiring-1 mpoly})$
 $\langle \text{proof} \rangle$

lemma *degree-Const[simp]*: *degree* $(\text{Const } c) \ x = 0$
 $\langle \text{proof} \rangle$

lemma *vars-add-subI*: $\text{vars } p \subseteq A \implies \text{vars } q \subseteq A \implies \text{vars } (p + q) \subseteq A$
 ⟨proof⟩

lemma *vars-mult-subI*: $\text{vars } p \subseteq A \implies \text{vars } q \subseteq A \implies \text{vars } (p * q) \subseteq A$
 ⟨proof⟩

lemma *vars-eqI*: **assumes** $\text{vars } (p :: 'a :: \text{comm-ring-1 mpoly}) \subseteq V$
 $\bigwedge v. v \in V \implies \exists a b. \text{insertion } a \ p \neq \text{insertion } (a(v := b)) \ p$
shows $\text{vars } p = V$
 ⟨proof⟩

end

2.2 Part 2 – Extensions With Importing Univariate Polynomials

theory *Preliminaries-on-Polynomials-2*
imports
 Preliminaries-on-Polynomials-1
 Factor-Algebraic-Polynomial.Poly-Connection
begin

Several definitions have the same name for univariate and multivariate polynomials, so we use a prefix m for multi-variate.

hide-const (**open**) *Symmetric-Polynomials.lead-coeff*

abbreviation *mdegree* **where** $mdegree \equiv MPoly\text{-Type.degree}$

abbreviation *mcoeff* **where** $mcoeff \equiv MPoly\text{-Type.coeff}$

abbreviation *mmonom* **where** $mmonom \equiv MPoly\text{-Type.monom}$

lemma *range-coeff-poly-to-mpoly*: **assumes** $mcoeff \ (poly\text{-to-mpoly } x \ p) \ m \neq 0$
shows $\exists d. m = \text{monomial } d \ x$
 ⟨proof⟩

lemma *degree-poly-to-mpoly[simp]*: $mdegree \ (poly\text{-to-mpoly } x \ p) \ x = \text{degree } p$
 ⟨proof⟩

lemma *degree-mpoly-to-poly*: **assumes** $\text{vars } p \subseteq \{x\}$
shows $\text{degree } (mpoly\text{-to-poly } x \ p) = mdegree \ p \ x$
 ⟨proof⟩

lemma *degree-partial-insertion-bound*: $\text{degree } (partial\text{-insertion } a \ x \ p) \leq MPoly\text{-Type.degree } p \ x$
 ⟨proof⟩

lemma *insertion-partial-insertion-vars*: **assumes** $\bigwedge y. y \neq x \implies y \in \text{vars } p \implies \beta \ y = \alpha \ y$

shows $\text{poly } (\text{partial-insertion } \beta \ x \ p) \ (\alpha \ x) = \text{insertion } \alpha \ p$
<proof>

lemma *degree-mpoly-of-poly[simp]*: $\text{mdegree } (\text{mpoly-of-poly } x \ p) \ x = \text{degree } p$
<proof>

lemma *mpoly-extI*: **assumes** $\bigwedge \alpha. \text{insertion } \alpha \ p = \text{insertion } \alpha \ (q :: 'a :: \{\text{ring-char-0}, \text{idom}\})$
mpoly
shows $p = q$
<proof>

lemma *vars-empty-Const*: **assumes** $\text{vars } (p :: 'a :: \{\text{ring-char-0}, \text{idom}\}) \text{ mpoly} = \{\}$
shows $\exists c. p = \text{Const } c$
<proof>

context

assumes *ge1*: $\bigwedge c :: 'a :: \text{linordered-idom}. c > 0 \implies \exists x. c * x \geq 1$
begin

lemma *poly-ext-bounded*:
fixes $p \ q :: 'a \ \text{poly}$
assumes $\bigwedge x. x \geq b \implies \text{poly } p \ x = \text{poly } q \ x$ **shows** $p = q$
<proof>

lemma *mpoly-ext-bounded*:
assumes $\bigwedge \alpha. (\bigwedge x. \alpha \ x \geq b) \implies \text{insertion } \alpha \ p = \text{insertion } \alpha \ (q :: 'a :: \text{linordered-idom} \ \text{mpoly})$
shows $p = q$
<proof>
end

lemma *mpoly-ext-bounded-int*:
assumes $\bigwedge \alpha. (\bigwedge x. \alpha \ x \geq b) \implies \text{insertion } \alpha \ p = \text{insertion } \alpha \ (q :: \text{int} \ \text{mpoly})$
shows $p = q$
<proof>

lemma *mpoly-ext-bounded-field*:
assumes $\bigwedge \alpha. (\bigwedge x. \alpha \ x \geq b) \implies \text{insertion } \alpha \ p = \text{insertion } \alpha \ (q :: 'a :: \text{linordered-field} \ \text{mpoly})$
shows $p = q$
<proof>

lemma *mpoly-of-poly-is-poly-to-mpoly*: $\text{mpoly-of-poly} = \text{poly-to-mpoly}$
<proof>

lemma *insertion-poly-to-mpoly [simp]*: $\text{insertion } f \ (\text{poly-to-mpoly } i \ p) = \text{poly } p \ (f$

i)
⟨proof⟩

lemma *substitute-poly-to-mpoly*:

assumes $x: \alpha x = \text{poly-to-mpoly } y \ (q :: 'a :: \{\text{ring-char-0, idom}\} \text{ poly})$
shows $\text{substitute } \alpha \ (\text{poly-to-mpoly } x \ p) = \text{poly-to-mpoly } y \ (\text{pcompose } p \ q)$
⟨proof⟩

lemma *total-degree-add-Const*: $\text{total-degree } (p + \text{Const } (c :: 'a :: \text{comm-ring-1}))$
 $= \text{total-degree } p$
⟨proof⟩

lemma *mpoly-as-sum-any*: $(p :: 'a :: \text{comm-ring-1} \ \text{mpoly}) = \text{Sum-any } (\lambda m. \text{mmonom } m \ (\text{mcoeff } p \ m))$
⟨proof⟩

lemma *mpoly-as-sum*: $(p :: 'a :: \text{comm-ring-1} \ \text{mpoly}) = \text{sum } (\lambda m. \text{mmonom } m \ (\text{mcoeff } p \ m)) \ \{m . \text{mcoeff } p \ m \neq 0\}$
⟨proof⟩

lemma *monom-as-prod*: $\text{mmonom } m \ c = \text{Const } (c :: 'a :: \text{comm-semiring-1}) * \text{prod } (\lambda i. \text{Var } i \ ^{\text{lookup } m \ i}) \ (\text{keys } m)$
⟨proof⟩

lemma *poly-to-mpoly-substitute-same*: **assumes** $\text{poly-to-mpoly } x \ q = \text{substitute } (\lambda i. \text{Var } x) \ p$
shows $\text{poly } q \ a = \text{insertion } (\lambda x. a) \ p$
⟨proof⟩

lemma *substitute-monom*: **fixes** $c :: 'a :: \text{comm-semiring-1}$
shows $\text{substitute } a \ (\text{mmonom } m \ c) = \text{Const } c * \text{prod } (\lambda i. a \ i \ ^{\text{lookup } m \ i}) \ (\text{keys } m)$
⟨proof⟩

lemma *degree-prod*: **assumes** $\text{prod } p \ A \neq (0 :: 'a :: \text{idom } \text{mpoly})$
shows $\text{mdegree } (\text{prod } p \ A) \ x = \text{sum } (\lambda i. \text{mdegree } (p \ i) \ x) \ A$
⟨proof⟩

lemma *degree-prod-le*: **fixes** $p :: - \Rightarrow 'a :: \text{idom } \text{mpoly}$
shows $\text{mdegree } (\text{prod } p \ A) \ x \leq \text{sum } (\lambda i. \text{mdegree } (p \ i) \ x) \ A$
⟨proof⟩

lemma *degree-power*: **assumes** $p \neq (0 :: 'a :: \text{idom } \text{mpoly})$
shows $\text{mdegree } (p \ ^n) \ x = n * \text{mdegree } p \ x$
⟨proof⟩

lemma *mdegree-Const-mult-le*: $\text{mdegree } (\text{Const } (c :: 'a :: \text{idom}) * p) \ x \leq \text{mdegree } p \ x$
⟨proof⟩

lemma *degree-substitute-const-same-var*: $mdegree (substitute (\lambda i. Const (c i) * Var x) (p :: 'a :: idom mpoly)) x \leq total-degree p$
 ⟨proof⟩

lemma *degree-substitute-same-var*: $mdegree (substitute (\lambda i. Var x) (p :: 'a :: idom mpoly)) x \leq total-degree p$
 ⟨proof⟩

lemma *poly-pinfty-ge-int*: **assumes** $0 < lead-coeff (p :: int poly)$
and $degree p \neq 0$
shows $\exists n. \forall x \geq n. b \leq poly p x$
 ⟨proof⟩

context

assumes *poly-pinfty-ge*: $\bigwedge p b. 0 < lead-coeff (p :: 'a :: linordered-idom poly)$
 $\implies degree p \neq 0 \implies \exists n. \forall x \geq n. b \leq poly p x$

begin

lemma *degree-mono-generic*: **assumes** *pos*: $lead-coeff p \geq (0 :: 'a)$
and *le*: $\bigwedge x. x \geq c \implies poly p x \leq poly q x$
shows $degree p \leq degree q$
 ⟨proof⟩

lemma *degree-mono'-generic*: **assumes** *le*: $\bigwedge x. x \geq c \implies (bnd :: 'a) \leq poly p x$
 $\wedge poly p x \leq poly q x$
shows $degree p \leq degree q$
 ⟨proof⟩

end

definition *nneg-poly* :: $'a :: \{linordered-semidom, semiring-no-zero-divisors\} poly$
 $\Rightarrow bool$ **where**
 $nneg-poly p = ((\forall x. x \geq 0 \longrightarrow poly p x \geq 0) \wedge lead-coeff p \geq 0)$

lemma *nneg-poly-nneg*: **assumes** *nneg-poly p*
and $x \geq 0$
shows $poly p x \geq 0$
 ⟨proof⟩

lemma *nneg-poly-lead-coeff*: **assumes** *nneg-poly p*
shows $p \neq 0 \implies lead-coeff p > 0$
 ⟨proof⟩

lemma *nneg-poly-add*: **assumes** *nneg-poly p nneg-poly q*
shows $nneg-poly (p + q) \wedge degree (p + q) = \max (degree p) (degree q)$
 ⟨proof⟩

lemma *nneg-poly-mult*: **assumes** *nneg-poly p nneg-poly q*
shows *nneg-poly (p * q)*
 ⟨*proof*⟩

lemma *nneg-poly-const[simp]*: *nneg-poly [:c:] = (c ≥ 0)*
 ⟨*proof*⟩

lemma *nneg-poly-pCons[simp]*: *a ≥ 0 ∧ nneg-poly p ⇒ nneg-poly (pCons a p)*
 ⟨*proof*⟩

lemma *nneg-poly-0[simp]*: *nneg-poly 0*
 ⟨*proof*⟩

lemma *nneg-poly-pcompose*: **assumes** *nneg-poly p nneg-poly q*
shows *nneg-poly (pcompose p q)*
 ⟨*proof*⟩

lemma *nneg-poly-degree-add-1*: **assumes** *p: nneg-poly p* **and** *a: a1 > 0 a2 > 0*
shows *degree (p * [:b, a1:] + [:c, a2:]) = 1 + degree p*
 ⟨*proof*⟩

lemma *nneg-poly-degree-add*: **assumes** *pq: nneg-poly (p :: 'a :: linordered-idom poly) nneg-poly q*
and *a: a3 > 0 a2 > 0 a1 > 0*
shows *degree ([:a3:] * q * p + ([:a2:] * q + [:a1:] * p + [:a0:])) = degree p + degree q*
 ⟨*proof*⟩

lemma *poly-pinfty-gt-lc*:
fixes *p :: 'a :: linordered-field poly*
assumes *lead-coeff p > 0*
shows $\exists n. \forall x \geq n. \text{poly } p \ x \geq \text{lead-coeff } p$
 ⟨*proof*⟩

lemma *poly-pinfty-ge*:
fixes *p :: 'a :: linordered-field poly*
assumes *lead-coeff p > 0 degree p ≠ 0*
shows $\exists n. \forall x \geq n. \text{poly } p \ x \geq b$
 ⟨*proof*⟩

lemma *nneg-polyI*: **fixes** *p :: 'a::linordered-field poly*
assumes $\bigwedge x. 0 \leq x \implies 0 \leq \text{poly } p \ x$
shows *nneg-poly p*
 ⟨*proof*⟩

lemma *poly-bounded*: **fixes** *x :: 'a:: linordered-idom*

assumes $abs\ x \leq b$
shows $abs\ (poly\ p\ x) \leq (\sum\ i \leq\ degree\ p.\ abs\ (coeff\ p\ i) * b^i)$
 $\langle proof \rangle$

lemma *poly-degree-le-large-const*:
assumes $pq: degree\ (p :: 'a :: linordered-field\ poly) \geq degree\ q$
and $p0: \bigwedge x. x \geq 0 \implies poly\ p\ x \geq 0$
shows $\exists H. \forall h \geq H. \forall x \geq 0. h * poly\ p\ x + h \geq poly\ q\ x$
 $\langle proof \rangle$

lemma *degree-monom-0[simp]*: $degree-monom\ 0 = 0$
 $\langle proof \rangle$

lemma *degree-monom-monomial[simp]*: $degree-monom\ (monomial\ n\ x) = n$
 $\langle proof \rangle$

lemma *keys-add*: $keys\ (m + n :: monom) = keys\ m \cup keys\ n$
 $\langle proof \rangle$

lemma *degree-monom-add[simp]*: $degree-monom\ (m + n) = degree-monom\ m + degree-monom\ n$
 $\langle proof \rangle$

lemma *degree-monom-of-set*: $finite\ xs \implies degree-monom\ (monom-of-set\ xs) = card\ xs$
 $\langle proof \rangle$

lemma *keys-singletonE*: **assumes** $keys\ m = \{x\}$
shows $\exists c. m = monomial\ c\ x \wedge c = degree-monom\ m \wedge c \neq 0$
 $\langle proof \rangle$

lemma *degree-monom-0-iff*: $degree-monom\ m = 0 \iff m = 0$
 $\langle proof \rangle$

lemma *degree-0-imp-Const*: **fixes** $p :: 'a :: comm-ring-1\ mpoly$
assumes $d0: total-degree\ p = 0$
shows $\exists c. p = Const\ c$
 $\langle proof \rangle$

lemma *binary-degree-2-poly*: **fixes** $p :: 'a :: \{ring-char-0, idom\}\ mpoly$
assumes $td: total-degree\ p \leq 2$
and $vars: vars\ p = \{x, y\}$
and $xy: x \neq y$
shows $\exists a\ b\ c\ d\ e\ f.$
 $p = Const\ a + Const\ b * Var\ x + Const\ c * Var\ y +$
 $Const\ d * Var\ x * Var\ x + Const\ e * Var\ y * Var\ y + Const\ f * Var\ x * Var$
 y
 $\langle proof \rangle$

lemma *bounded-negative-factor*: **assumes** $\bigwedge x. c \leq (x :: 'a :: \text{linordered-field}) \implies$
 $a * x \leq b$
shows $a \leq 0$
 $\langle \text{proof} \rangle$

end

3 Definition of Monotone Algebras and Polynomial Interpretations

theory *Polynomial-Interpretation*

imports

Preliminaries-on-Polynomials-1

First-Order-Terms.Term

First-Order-Terms.Subterm-and-Context

begin

abbreviation $PVar \equiv MPoly\text{-Type}.Var$

abbreviation $TVar \equiv Term.Var$

type-synonym $(f, v)rule = (f, v)term \times (f, v)term$

We fix the domain to the set of nonnegative numbers

lemma *subterm-size[termination-simp]*: $x < \text{length } ts \implies \text{size } (ts ! x) < Suc$
 $(\text{size-list size } ts)$
 $\langle \text{proof} \rangle$

definition *assignment* :: $(var \Rightarrow 'a :: \{ord, zero\}) \Rightarrow bool$ **where**
 $\text{assignment } \alpha = (\forall x. \alpha x \geq 0)$

lemma *assignmentD*: **assumes** $\text{assignment } \alpha$
shows $\alpha x \geq 0$
 $\langle \text{proof} \rangle$

definition *monotone-fun-wrt* :: $('a :: \{zero, ord\}) \Rightarrow 'a \Rightarrow bool) \Rightarrow nat \Rightarrow ('a \text{ list}$
 $\Rightarrow 'a) \Rightarrow bool$ **where**
 $\text{monotone-fun-wrt } gt \ n \ f = (\forall v' \ i \ vs. \text{length } vs = n \longrightarrow (\forall v \in \text{set } vs. v \geq 0)$
 $\longrightarrow i < n \longrightarrow gt \ v' \ (vs ! i) \longrightarrow$
 $gt \ (f \ (vs [i := v'])) \ (f \ vs))$

definition *valid-fun* :: $nat \Rightarrow ('a \text{ list} \Rightarrow 'a :: \{zero, ord\}) \Rightarrow bool$ **where**
 $\text{valid-fun } n \ f = (\forall vs. \text{length } vs = n \longrightarrow (\forall v \in \text{set } vs. v \geq 0) \longrightarrow f \ vs \geq 0)$

definition *monotone-poly-wrt* :: $('a :: \{comm\text{-semiring-1}, zero, ord\}) \Rightarrow 'a \Rightarrow bool)$
 $\Rightarrow var \text{ set} \Rightarrow 'a \text{ mpoly} \Rightarrow bool$ **where**
 $\text{monotone-poly-wrt } gt \ V \ p = (\forall \alpha \ x \ v. \text{assignment } \alpha \longrightarrow x \in V \longrightarrow gt \ v \ (\alpha \ x)$
 \longrightarrow
 $gt \ (\text{insertion } (\alpha(x := v)) \ p) \ (\text{insertion } \alpha \ p))$

definition *valid-poly* :: 'a :: {ord,comm-semiring-1} mpoly \Rightarrow bool **where**
valid-poly p = (\forall α . assignment $\alpha \longrightarrow$ insertion α p \geq 0)

locale *term-algebra* =
fixes F :: ('f \times nat) set
and I :: 'f \Rightarrow ('a :: {ord,zero} list) \Rightarrow 'a
and gt :: 'a \Rightarrow 'a \Rightarrow bool
begin

abbreviation *monotone-fun* **where** *monotone-fun* \equiv *monotone-fun-wrt* gt

definition *valid-monotone-fun* :: ('f \times nat) \Rightarrow bool **where**
valid-monotone-fun fn = (\forall f n p. fn = (f,n) \longrightarrow p = I f
 \longrightarrow *valid-fun* n p \wedge *monotone-fun* n p)

definition *valid-monotone-inter* **where** *valid-monotone-inter* = Ball F *valid-monotone-fun*

definition *orient-rule* :: ('f,var)rule \Rightarrow bool **where**
orient-rule rule = (case rule of (l,r) \Rightarrow (\forall α . assignment $\alpha \longrightarrow$ gt (I[[l]] α)
(I[[r]] α)))
end

locale *omega-term-algebra* = *term-algebra* F I (>) :: int \Rightarrow int \Rightarrow bool **for** F **and**
I :: 'f \Rightarrow - +

assumes *vm-inter*: *valid-monotone-inter*

begin

definition *termination-by-interpretation* :: ('f,var) rule set \Rightarrow bool **where**
termination-by-interpretation R = (\forall (l,r) \in R. *orient-rule* (l,r) \wedge *funas-term* l
 \cup *funas-term* r \subseteq F)
end

locale *poly-inter* =
fixes F :: ('f \times nat) set
and I :: 'f \Rightarrow 'a :: *linordered-idom* mpoly
and gt :: 'a \Rightarrow 'a \Rightarrow bool (**infix** \succ 50)
begin

definition I' **where** I' f vs = insertion (λ i. if i < length vs then vs ! i else 0) (I
f)
sublocale *term-algebra* F I' gt <proof>

abbreviation *monotone-poly* **where** *monotone-poly* \equiv *monotone-poly-wrt* gt

abbreviation *weakly-monotone-poly* **where** *weakly-monotone-poly* \equiv *monotone-poly-wrt*
(\geq)

definition *gt-poly* :: 'a mpoly \Rightarrow 'a mpoly \Rightarrow bool (**infix** \succ_p 50) **where**
 $(p \succ_p q) = (\forall \alpha. \text{assignment } \alpha \longrightarrow \text{insertion } \alpha p \succ \text{insertion } \alpha q)$

definition *valid-monotone-poly* :: ('f \times nat) \Rightarrow bool **where**
 $\text{valid-monotone-poly } fn = (\forall f n p. fn = (f, n) \longrightarrow p = I f$
 $\longrightarrow \text{valid-poly } p \wedge \text{monotone-poly } \{..<n\} p \wedge \text{vars } p = \{..<n\})$

definition *valid-weakly-monotone-poly* :: ('f \times nat) \Rightarrow bool **where**
 $\text{valid-weakly-monotone-poly } fn = (\forall f n p. fn = (f, n) \longrightarrow p = I f$
 $\longrightarrow \text{valid-poly } p \wedge \text{weakly-monotone-poly } \{..<n\} p \wedge \text{vars } p \subseteq \{..<n\})$

definition *valid-monotone-poly-inter* **where** $\text{valid-monotone-poly-inter} = \text{Ball } F$
 $\text{valid-monotone-poly}$

definition *valid-weakly-monotone-inter* **where** $\text{valid-weakly-monotone-inter} = \text{Ball}$
 $F \text{ valid-weakly-monotone-poly}$

fun *eval* :: ('f, var)term \Rightarrow 'a mpoly **where**
 $\text{eval } (TVar x) = PVar x$
 $| \text{eval } (Fun f ts) = \text{substitute } (\lambda i. \text{if } i < \text{length } ts \text{ then } \text{eval } (ts ! i) \text{ else } 0) (I f)$

lemma *I'-is-insertion-eval*: $I' \llbracket t \rrbracket \alpha = \text{insertion } \alpha (\text{eval } t)$
 $\langle \text{proof} \rangle$

lemma *orient-rule*: $\text{orient-rule } (l, r) = (\text{eval } l \succ_p \text{eval } r)$
 $\langle \text{proof} \rangle$

lemma *vars-eval*: $\text{vars } (\text{eval } t) \subseteq \text{vars-term } t$
 $\langle \text{proof} \rangle$

lemma *monotone-imp-weakly-monotone*: **assumes** *valid*: $\text{valid-monotone-poly } p$
and *gt*: $\bigwedge x y. (x \succ y) = (x > y)$
shows $\text{valid-weakly-monotone-poly } p$
 $\langle \text{proof} \rangle$

lemma *valid-imp-insertion-eval-pos*: **assumes** *valid*: $\text{valid-monotone-poly-inter}$
and *funas-term* $t \subseteq F$
and *assignment* α
shows $\text{insertion } \alpha (\text{eval } t) \geq 0$
 $\langle \text{proof} \rangle$

end

locale *delta-poly-inter* = $\text{poly-inter } F I (\lambda x y. x \geq y + \delta)$ **for** $F :: ('f \times \text{nat}) \text{ set}$
and I **and**

$\delta :: 'a :: \{\text{floor-ceiling, linordered-field}\} +$
assumes *valid*: $\text{valid-monotone-poly-inter}$
and $\delta 0$: $\delta > 0$

begin

definition *termination-by-delta-interpretation* :: ('f, var) rule set \Rightarrow bool **where**

termination-by-delta-interpretation $R = (\forall (l,r) \in R. \text{orient-rule } (l,r) \wedge \text{funas-term } l \cup \text{funas-term } r \subseteq F)$

end

locale *int-poly-inter* = *poly-inter* $F I (>) :: \text{int} \Rightarrow \text{int} \Rightarrow \text{bool}$ **for** $F :: ('f \times \text{nat}) \text{ set}$ **and** $I +$

assumes *valid*: *valid-monotone-poly-inter*

begin

sublocale *omega-term-algebra* $F I'$

<proof>

definition *termination-by-poly-interpretation* $:: ('f, \text{var}) \text{ rule set} \Rightarrow \text{bool}$ **where**

termination-by-poly-interpretation = *termination-by-interpretation*

end

locale *wm-int-poly-inter* = *poly-inter* $F I (>) :: \text{int} \Rightarrow \text{int} \Rightarrow \text{bool}$ **for** $F :: ('f \times \text{nat}) \text{ set}$ **and** $I +$

assumes *valid*: *valid-weakly-monotone-inter*

begin

definition *oriented-by-interpretation* $:: ('f, \text{var}) \text{ rule set} \Rightarrow \text{bool}$ **where**

oriented-by-interpretation $R = (\forall (l,r) \in R. \text{orient-rule } (l,r) \wedge \text{funas-term } l \cup \text{funas-term } r \subseteq F)$

end

locale *linear-poly-inter* = *poly-inter* $F I \text{ gt}$ **for** $F I \text{ gt} +$

assumes *linear*: $\bigwedge f n. (f,n) \in F \Longrightarrow \text{total-degree } (I f) \leq 1$

locale *linear-int-poly-inter* = *int-poly-inter* $F I +$ *linear-poly-inter* $F I (>)$

for $F :: ('f \times \text{nat}) \text{ set}$ **and** I

locale *linear-wm-int-poly-inter* = *wm-int-poly-inter* $F I +$ *linear-poly-inter* $F I (>)$

for $F :: ('f \times \text{nat}) \text{ set}$ **and** I

definition *termination-by-linear-int-poly-interpretation* $:: ('f \times \text{nat}) \text{ set} \Rightarrow ('f, \text{var}) \text{ rule set} \Rightarrow \text{bool}$ **where**

termination-by-linear-int-poly-interpretation $F R = (\exists I. \text{linear-int-poly-inter } F I \wedge \text{int-poly-inter.termination-by-poly-interpretation } F I R)$

definition *omega-termination* $:: ('f \times \text{nat}) \text{ set} \Rightarrow ('f, \text{var}) \text{ rule set} \Rightarrow \text{bool}$ **where**

omega-termination $F R = (\exists I. \text{omega-term-algebra } F I \wedge \text{omega-term-algebra.termination-by-interpretation } F I R)$

definition *termination-by-int-poly-interpretation* $:: ('f \times \text{nat}) \text{ set} \Rightarrow ('f, \text{var}) \text{ rule set} \Rightarrow \text{bool}$ **where**

termination-by-int-poly-interpretation $F R = (\exists I. \text{int-poly-inter } F I \wedge$

int-poly-inter.termination-by-poly-interpretation F I R)

definition *termination-by-delta-poly-interpretation* :: 'a :: {floor-ceiling,linordered-field}
itself \Rightarrow ('f \times nat)set \Rightarrow ('f,var)rule set \Rightarrow bool **where**
termination-by-delta-poly-interpretation TYPE('a) F R = (\exists I δ . *delta-poly-inter*
F I (δ :: 'a) \wedge
delta-poly-inter.termination-by-delta-interpretation F I δ R)

definition *orientation-by-linear-wm-int-poly-interpretation* :: ('f \times nat)set \Rightarrow ('f,var)rule
set \Rightarrow bool **where**
orientation-by-linear-wm-int-poly-interpretation F R = (\exists I. *linear-wm-int-poly-inter*
F I \wedge
wm-int-poly-inter.oriented-by-interpretation F I R)

end

4 Hilbert's 10th Problem to Linear Inequality

theory *Hilbert10-to-Inequality*

imports

Preliminaries-on-Polynomials-1

begin

definition *hilbert10-problem* :: int mpoly \Rightarrow bool **where**
hilbert10-problem p = (\exists α . *insertion* α p = 0)

A polynomial is positive, if every coefficient is positive. Since the @{const coeff}-function of 'a mpoly maps a coefficient to every monomial, this means that positiveness is expressed as $\text{coeff } p \ m \neq (0::'a) \longrightarrow (0::'a) < \text{coeff } p \ m$ for monomials m . However, this condition is equivalent to just demand $(0::'a) \leq \text{coeff } p \ m$ for all m .

This is the reason why *positive polynomials* are defined in the same way as one would define *non-negative polynomials*.

definition *positive-poly* :: 'a :: linordered-idom mpoly \Rightarrow bool **where**
positive-poly p = (\forall m. *coeff* p m \geq 0)

definition *positive-interpr* :: (var \Rightarrow 'a :: linordered-idom) \Rightarrow bool **where**
positive-interpr α = (\forall x. α x $>$ 0)

definition *positive-poly-problem* :: 'a :: linordered-idom mpoly \Rightarrow 'a mpoly \Rightarrow bool
where
positive-poly p \Longrightarrow *positive-poly* q \Longrightarrow *positive-poly-problem* p q =
(\exists α . *positive-interpr* α \wedge *insertion* α p \geq *insertion* α q)

datatype flag = Positive | Negative | Zero

fun *flag-of* :: 'a :: {ord,zero} \Rightarrow flag **where**
flag-of x = (if x < 0 then Negative else if x > 0 then Positive else Zero)

definition *subst-flag* :: *var set* \Rightarrow (*var* \Rightarrow *flag*) \Rightarrow *var* \Rightarrow 'a :: *comm-ring-1 mpoly*
where

subst-flag *V flag* *x* = (if *x* \in *V* then (case *flag* *x* of
 Positive \Rightarrow *Var* *x*
 | Negative \Rightarrow - *Var* *x*
 | Zero \Rightarrow 0)
 else 0)

definition *assignment-flag* :: *var set* \Rightarrow (*var* \Rightarrow *flag*) \Rightarrow (*var* \Rightarrow 'a :: *comm-ring-1*)
 \Rightarrow (*var* \Rightarrow 'a) **where**

assignment-flag *V flag* α *x* = (if *x* \in *V* then (case *flag* *x* of
 Positive \Rightarrow α *x*
 | Negative \Rightarrow - α *x*
 | Zero \Rightarrow 1)
 else 1)

definition *correct-flags* :: *var set* \Rightarrow (*var* \Rightarrow *flag*) \Rightarrow (*var* \Rightarrow 'a :: *ordered-comm-ring*)
 \Rightarrow *bool* **where**

correct-flags *V flag* α = (\forall *x* \in *V*. *flag* *x* = *flag-of* (α *x*))

lemma *correct-flag-substitutions*: **fixes** *p* :: 'a :: *linordered-idom mpoly*

assumes *vars* *p* \subseteq *V*

and *beta*: β = *assignment-flag* *V flag* α

and *sigma*: σ = *subst-flag* *V flag*

and *q*: *q* = *substitute* σ *p*

and *corr*: *correct-flags* *V flag* α

shows *insertion* β *q* = *insertion* α *p* *positive-interpr* β
 <proof>

definition *hilbert-encode1* :: *int mpoly* \Rightarrow *int mpoly list* **where**

hilbert-encode1 *r* = (let *r2* = *r*²;

V = *vars-list* *r2*;

flag-lists = *product-lists* (*map* (λ *x*. *map* (λ *f*. (*x*,*f*)) [*Positive*,*Negative*,*Zero*])
V);

subst = (λ *fl*. *subst-flag* (*set* *V*) (λ *x*. case *map-of fl* *x* of *Some* *f* \Rightarrow *f* | *None*
 \Rightarrow *Zero*))

in *map* (λ *fl*. *substitute* (*subst fl*) *r2*) *flag-lists*)

lemma *hilbert-encode1*:

hilbert10-problem *r* \longleftrightarrow (\exists *p* \in *set* (*hilbert-encode1* *r*). \exists α . *positive-interpr* α \wedge
insertion α *p* \leq 0)
 <proof>

lemma *pos-neg-split*: *mpoly-coeff-filter* (λ *x*. (*x* :: 'a :: *linordered-idom*) $>$ 0) *p* +
mpoly-coeff-filter (λ *x*. *x* $<$ 0) *p* = *p* (**is** ?*l* + ?*r* = *p*)
 <proof>

definition *hilbert-encode2* :: *int mpoly* \Rightarrow *int mpoly* \times *int mpoly* **where**

hilbert-encode2 *p* =
 (− *mpoly-coeff-filter* (λ *x*. *x* < 0) *p*, *mpoly-coeff-filter* (λ *x*. *x* > 0) *p*)

lemma *hilbert-encode2*: **assumes** *hilbert-encode2* *p* = (*r*,*s*)
shows *positive-poly r positive-poly s insertion α p ≤ 0 ↔ insertion α r ≥ insertion α s*
 ⟨*proof*⟩

definition *hilbert-encode* :: *int mpoly* ⇒ (*int mpoly* × *int mpoly*)*list* **where**
hilbert-encode = *map hilbert-encode2 o hilbert-encode1*

Lemma 2.2 in paper

lemma *hilbert-encode-positive: hilbert10-problem p*
 ↔ (∃ (*r*,*s*) ∈ *set (hilbert-encode p)*. *positive-poly-problem r s*)
 ⟨*proof*⟩

end

5 Undecidability of Linear Polynomial Termination

theory *Linear-Poly-Termination-Undecidable*

imports

Hilbert10-to-Inequality

Polynomial-Interpretation

begin

Definition 3.1

locale *poly-input* =
fixes *p q* :: *int mpoly*
assumes *pq: positive-poly p positive-poly q*
begin

datatype *symbol* = *a-sym* | *z-sym* | *o-sym* | *f-sym* | *v-sym var* | *q-sym* | *h-sym* | *g-sym*

abbreviation *a-t* **where** *a-t t1 t2* ≡ *Fun a-sym [t1, t2]*

abbreviation *z-t* **where** *z-t* ≡ *Fun z-sym []*

abbreviation *o-t* **where** *o-t* ≡ *Fun o-sym []*

abbreviation *f-t* **where** *f-t t1 t2 t3 t4* ≡ *Fun f-sym [t1,t2,t3,t4]*

abbreviation *v-t* **where** *v-t i t* ≡ *Fun (v-sym i) [t]*

definition *encode-num* :: *var* ⇒ *int* ⇒ (*symbol*,*var*)*term* **where**

encode-num x n = ((λ *t*. *a-t (Var x) t*) $\widetilde{\widetilde{}}$ (*nat n*)) *z-t*

definition *encode-monom* :: *var* ⇒ *monom* ⇒ *int* ⇒ (*symbol*,*var*)*term* **where**

encode-monom x m c = *rec-list (encode-num x c) (λ (*i*,*e*) -. (λ *t*. *v-t i t*) $\widetilde{\widetilde{e}}$)*
 (*var-list m*)

definition *encode-poly* :: *var* \Rightarrow *int mpoly* \Rightarrow (*symbol, var*)*term* **where**
encode-poly *x r* = *rec-list z-t* (λ (*m, c*) - *t. a-t* (*encode-monom x m c*) *t*) (*monom-list r*)

lemma *vars-encode-num*: *vars-term* (*encode-num x n*) \subseteq {*x*}
<proof>

lemma *vars-encode-monom*: *vars-term* (*encode-monom x m c*) \subseteq {*x*}
<proof>

lemma *vars-encode-poly*: *vars-term* (*encode-poly x r*) \subseteq {*x*}
<proof>

definition *V* **where** *V* = *vars p* \cup *vars q*

definition *y1* :: *var* **where** *y1* = 0

definition *y2* :: *var* **where** *y2* = 1

definition *y3* :: *var* **where** *y3* = 2

lemma *y-vars*: *y1* \neq *y2* *y2* \neq *y3* *y1* \neq *y3*
<proof>

Definition 3.3

definition *lhs-R* = *f-t* (*Var y1*) (*Var y2*) (*a-t* (*encode-poly y3 p*) (*Var y3*)) *o-t*

definition *rhs-R* = *f-t* (*a-t* (*Var y1*) *z-t*) (*a-t z-t* (*Var y2*)) (*a-t* (*encode-poly y3 q*) (*Var y3*)) *z-t*

definition *F* **where** *F* = {(*a-sym*, 2), (*z-sym*, 0)} \cup (λ *i. (v-sym i, 1 :: nat)) \cup *V**

definition *F-R* **where** *F-R* = {(*f-sym*, 4), (*o-sym*, 0)} \cup *F*

definition *R* **where** *R* = {(*lhs-R*, *rhs-R*)}

definition *V-list* **where** *V-list* = *sorted-list-of-set V*

definition *contexts* :: (*symbol* \times *nat* \times *nat*) *list*

where *contexts* = [

(*a-sym*, 2, 0),

(*a-sym*, 2, 1),

(*f-sym*, 4, 0),

(*f-sym*, 4, 1),

(*f-sym*, 4, 2),

(*f-sym*, 4, 3)] @

map (λ *i. (v-sym i, 1, 0)) *V-list**

replace *t* by *f*(*z*, ..., *z*, *t*, *z*, ..., *z*)

definition *z-context* :: *symbol* \times *nat* \times *nat* \Rightarrow (*symbol, var*)*term* \Rightarrow (*symbol, var*)*term* **where**

$z\text{-context } c \ t = (\text{case } c \text{ of } (f, n, i) \Rightarrow \text{Fun } f \ (\text{replicate } i \ z\text{-}t \ @ \ [t] \ @ \ \text{replicate } (n - i - 1) \ z\text{-}t))$

definition $z\text{-contexts}$ **where**

$z\text{-contexts } cs = \text{foldr } z\text{-context } cs$

definition $\text{all-symbol-pos-ctxt} :: (\text{symbol}, \text{var})\text{term} \Rightarrow (\text{symbol}, \text{var})\text{term}$ **where**

$\text{all-symbol-pos-ctxt} = z\text{-contexts contexts}$

definition $\text{lhs-}R' = \text{all-symbol-pos-ctxt lhs-}R$

definition $\text{rhs-}R' = \text{all-symbol-pos-ctxt rhs-}R$

definition R' **where** $R' = \{(\text{lhs-}R', \text{rhs-}R')\}$

lemma funas-encode-num : $\text{funas-term } (\text{encode-num } x \ n) \subseteq F$

$\langle \text{proof} \rangle$

lemma $\text{funas-encode-monom}$: **assumes** $\text{keys } m \subseteq V$

shows $\text{funas-term } (\text{encode-monom } x \ m \ c) \subseteq F$

$\langle \text{proof} \rangle$

lemma funas-encode-poly : **assumes** $\text{vars } r \subseteq V$ **shows** $\text{funas-term } (\text{encode-poly } x \ r) \subseteq F$

$\langle \text{proof} \rangle$

lemma $\text{funas-encode-poly-p}$: $\text{funas-term } (\text{encode-poly } x \ p) \subseteq F$

$\langle \text{proof} \rangle$

lemma $\text{funas-encode-poly-q}$: $\text{funas-term } (\text{encode-poly } x \ q) \subseteq F$

$\langle \text{proof} \rangle$

lemma $\text{lhs-}R\text{-}F$: $\text{funas-term lhs-}R \subseteq F\text{-}R$

$\langle \text{proof} \rangle$

lemma $\text{rhs-}R\text{-}F$: $\text{funas-term rhs-}R \subseteq F\text{-}R$

$\langle \text{proof} \rangle$

lemma $\text{finite-}V$: $\text{finite } V \langle \text{proof} \rangle$

lemma $V\text{-list}$: $\text{set } V\text{-list} = V \langle \text{proof} \rangle$

lemma contexts : **assumes** $(f, n, i) \in \text{set contexts}$

shows $(f, n) \in F\text{-}R \ i < n$

$\langle \text{proof} \rangle$

lemma $z\text{-contexts-append}$: $z\text{-contexts } (cs \ @ \ ds) \ t = z\text{-contexts } cs \ (z\text{-contexts } ds \ t)$

$\langle \text{proof} \rangle$

lemma $z\text{-context}$: **assumes** $(f, n) \in F\text{-}R \ i < n$ **and** $\text{funas-term } t \subseteq F\text{-}R$

shows *funas-term* (z-context (f,n,i) t) \subseteq F-R
{proof}

lemma *funas-all-symbol-pos-ctxt*: **assumes** *funas-term* t \subseteq F-R
shows *funas-term* (all-symbol-pos-ctxt t) \subseteq F-R
{proof}

lemma *lhs-R'-F*: *funas-term* lhs-R' \subseteq F-R
{proof}

lemma *rhs-R'-F*: *funas-term* rhs-R' \subseteq F-R
{proof}
end

lemma *insertion-positive-poly*: **assumes** $\bigwedge x. \alpha x \geq (0 :: 'a :: \text{linordered-idom})$
and *positive-poly* p
shows *insertion* α p ≥ 0
{proof}

locale *solvable-poly-problem* = *poly-input* p q **for** p q +
assumes *sol*: *positive-poly-problem* p q
begin

definition α **where** $\alpha = (\text{SOME } \alpha. \text{positive-interpr } \alpha \wedge \text{insertion } \alpha q \leq \text{insertion } \alpha p)$

lemma α : *positive-interpr* α *insertion* α q \leq *insertion* α p
{proof}

lemma $\alpha 1$: $\alpha x > 0$ {proof}

context
fixes I :: *symbol* \Rightarrow *int* *mpoly*
assumes *inter*: I a-sym = PVar 0 + PVar 1
I z-sym = 0
I o-sym = 1
I (v-sym i) = Const (α i) * PVar 0
begin

lemma *inter-encode-num*: **assumes** c ≥ 0
shows *poly-inter.eval* I (encode-num x c) = Const c * PVar x
{proof}

lemma *inter-v-pow-e*: *poly-inter.eval* I ((v-t x $\hat{\sim}$ e) t) = Const ((α x) \hat{e}) *
poly-inter.eval I t
{proof}

lemma *inter-encode-monom*: **assumes** c: c ≥ 0

shows $\text{poly-inter.eval } I (\text{encode-monom } y \ m \ c) = \text{Const } (\text{insertion } \alpha (\text{monom } m \ c)) * \text{PVar } y$
 ⟨proof⟩

lemma *inter-foldr-v-t*:

$\text{poly-inter.eval } I (\text{foldr } v\text{-t } xs \ t) = \text{Const } (\text{prod-list } (\text{map } \alpha \ xs)) * \text{poly-inter.eval } I \ t$
 ⟨proof⟩

lemma *inter-encode-poly-generic*: **assumes** *positive-poly* r

shows $\text{poly-inter.eval } I (\text{encode-poly } x \ r) = \text{Const } (\text{insertion } \alpha \ r) * \text{PVar } x$
 ⟨proof⟩

lemma *valid-monotone-inter-F*: **assumes** *positive-interpr* α

and $\text{inF}: \text{fn } \in F$

shows $\text{poly-inter.valid-monotone-poly } I (>) \text{fn}$
 ⟨proof⟩

end

fun *I-R* :: *symbol* \Rightarrow *int mpoly* **where**

$I\text{-R } f\text{-sym} = \text{PVar } 0 + \text{PVar } 1 + \text{PVar } 2 + \text{PVar } 3$
 | $I\text{-R } a\text{-sym} = \text{PVar } 0 + \text{PVar } 1$
 | $I\text{-R } z\text{-sym} = 0$
 | $I\text{-R } o\text{-sym} = 1$
 | $I\text{-R } (v\text{-sym } i) = \text{Const } (\alpha \ i) * \text{PVar } 0$

interpretation *inter-R*: $\text{poly-inter } F\text{-R } I\text{-R } (>) \langle \text{proof} \rangle$

lemma *inter-R-encode-poly*: **assumes** *positive-poly* r

shows $\text{inter-R.eval } (\text{encode-poly } x \ r) = \text{Const } (\text{insertion } \alpha \ r) * \text{PVar } x$
 ⟨proof⟩

lemma *valid-monotone-inter-R*: $\text{inter-R.valid-monotone-poly-inter} \langle \text{proof} \rangle$

sublocale *inter-R*: $\text{linear-int-poly-inter } F\text{-R } I\text{-R}$

⟨proof⟩

lemma *orient-R-main*: **assumes** *assignment* β

shows $\text{insertion } \beta (\text{inter-R.eval } \text{lhs-R}) > \text{insertion } \beta (\text{inter-R.eval } \text{rhs-R})$
 ⟨proof⟩

The easy direction of Theorem 3.4

lemma *orient-R*: $\text{inter-R.termination-by-poly-interpretation } R$

⟨proof⟩

lemma *solution-imp-linear-termination-R*: $\text{termination-by-linear-int-poly-interpretation } F\text{-R } R$

<proof>
end

context *poly-input*
begin

lemma *inter-z-context:*

assumes $i: i < n$ **and** $I: I f = \text{Const } c0 + (\text{sum-list } (\text{map } (\lambda j. \text{Const } (c j)) * \text{PVar } j) [0..<n]))$
and $Ize: I z\text{-sym} = \text{Const } d0$
shows $\exists d. \forall t. \text{poly-inter.eval } I (z\text{-context } (f,n,i) t) = \text{Const } d + \text{Const } (c i) * \text{poly-inter.eval } I t$
<proof>

lemma *inter-z-contexts:*

assumes $cs: \bigwedge f n i. (f,n,i) \in \text{set } cs \implies i < n \wedge I f = \text{Const } (c0 f) + (\text{sum-list } (\text{map } (\lambda j. \text{Const } (c f j)) * \text{PVar } j) [0..<n]))$
and $Ize: I z\text{-sym} = \text{Const } d0$
shows $\exists d. \forall t. \text{poly-inter.eval } I (z\text{-contexts } cs t) = \text{Const } d + \text{Const } (\text{prod-list } (\text{map } (\lambda (f,n,i). c f i) cs)) * \text{poly-inter.eval } I t$
<proof>

lemma *inter-all-symbol-pos-ctxt-generic:*

assumes $f: I f\text{-sym} = \text{Const } fc + \text{Const } f0 * \text{PVar } 0 + \text{Const } f1 * \text{PVar } 1 + \text{Const } f2 * \text{PVar } 2 + \text{Const } f3 * \text{PVar } 3$
and $a: I a\text{-sym} = \text{Const } ac + \text{Const } a0 * \text{PVar } 0 + \text{Const } a1 * \text{PVar } 1$
and $v: \bigwedge i. i \in V \implies I (v\text{-sym } i) = \text{Const } (vc i) + \text{Const } (v0 i) * \text{PVar } 0$
and $I z\text{-sym} = \text{Const } zc$
shows $\exists d. \forall t. \text{poly-inter.eval } I (\text{all-symbol-pos-ctxt } t) = \text{Const } d + \text{Const } (\text{prod-list } ([a0, a1, f0, f1, f2, f3] @ \text{map } v0 V\text{-list})) * \text{poly-inter.eval } I t$
<proof>
end

context *solvable-poly-problem*
begin

lemma *inter-all-symbol-pos-ctxt:*

$\exists d e. e \geq 1 \wedge (\forall t. \text{inter-R.eval } (\text{all-symbol-pos-ctxt } t) = \text{Const } d + \text{Const } e * \text{inter-R.eval } t)$
<proof>

The easy direction of Theorem 3.4 for R'

lemma *orient-R': inter-R.termination-by-poly-interpretation R'*

<proof>

lemma *solution-imp-linear-termination-R': termination-by-linear-int-poly-interpretation F-R R'*

<proof>

end

Now for the other direction of Theorem 3.4

lemma *monotone-linear-poly-to-coeffs*: **fixes** $p :: \text{int}$ *mpoly*

assumes *linear*: *total-degree* $p \leq 1$

and *poly*: *valid-poly* p

and *mono*: *poly-inter.monotone-poly* $(>)$ $\{..<n\}$ p

and *vars*: *vars* $p = \{..<n\}$

shows $\exists c a. p = \text{Const } c + (\sum i \leftarrow [0..<n]. \text{Const } (a \ i) * \text{PVar } i)$
 $\wedge c \geq 0 \wedge (\forall i < n. a \ i > 0)$

<proof>

locale *poly-input-to-solution-common* = *poly-input* $p \ q +$

poly-inter $F' \ I \ (>) :: \text{int} \Rightarrow \text{int} \Rightarrow \text{bool}$ **for** $p \ q \ I$ **and** $F' :: (\text{poly-input.symbol} \times \text{nat}) \text{ set}$ **and** *argsL* *argsR* $+$

assumes *orient*:

orient-rule $(\text{Fun } f\text{-sym } ([\text{Var } y1, \text{Var } y2, a\text{-t } (\text{encode-poly } y3 \ p) (\text{Var } y3)]) \ @ \text{argsL}),$

$\text{Fun } f\text{-sym } ([a\text{-t } (\text{Var } y1) \ z\text{-t}, a\text{-t } z\text{-t } (\text{Var } y2), a\text{-t } (\text{encode-poly } y3 \ q) (\text{Var } y3)]) \ @ \text{argsR})$

and *len-args*: *length* *argsL* = *length* *argsR*

and *y123*: $\{y1, y2, y3\} \cap (\bigcup (\text{vars-term } ' \text{set } (\text{argsL} \ @ \ \text{argsR}))) = \{\}$

and *FF'*: *insert* $(f\text{-sym}, 3 + \text{length } \text{argsR}) \ F \subseteq F'$

and *linear-mono-interpretation*: $(g, n) \in \text{insert } (f\text{-sym}, 3 + \text{length } \text{argsR}) \ F \Longrightarrow$

$\exists c a. I \ g = \text{Const } c + (\sum i \leftarrow [0..<n]. \text{Const } (a \ i) * \text{PVar } i)$
 $\wedge c \geq 0 \wedge (\forall i < n. a \ i > 0)$

begin

abbreviation *ff* **where** $ff \equiv (f\text{-sym}, 3 + \text{length } \text{argsR})$

abbreviation *args* **where** $\text{args} \equiv [3..<\text{length } \text{argsR} + 3]$

lemma *extract-a-poly*: $\exists a0 \ a1 \ a2. I \ a\text{-sym} = \text{Const } a0 + \text{Const } a1 * \text{PVar } 0 + \text{Const } a2 * \text{PVar } 1$

$\wedge a0 \geq 0 \wedge a1 > 0 \wedge a2 > 0$

<proof>

lemma *extract-f-poly*: $\exists f0 \ f1 \ f2 \ f3 \ f4. I \ f\text{-sym} = \text{Const } f0 + \text{Const } f1 * \text{PVar } 0 + \text{Const } f2 * \text{PVar } 1$

$+ \text{Const } f3 * \text{PVar } 2 + (\sum i \leftarrow \text{args}. \text{Const } (f4 \ i) * \text{PVar } i)$

$\wedge f0 \geq 0 \wedge f1 > 0 \wedge f2 > 0 \wedge f3 > 0$

<proof>

lemma *extract-z-poly*: $\exists ze0. I \ z\text{-sym} = \text{Const } ze0 \wedge ze0 \geq 0$

<proof>

lemma *solution*: *positive-poly-problem* $p \ q$

<proof>

end

locale *solution-poly-input-R* = *poly-input* *p q* + *poly-inter* *F-R I* ($>$) :: *int* \Rightarrow -
for *p q I* +
assumes *orient*: *orient-rule* (*lhs-R,rhs-R*)
and *linear-mono-interpretation*: $(g,n) \in F-R \implies$
 $\exists c a. I g = \text{Const } c + (\sum i \leftarrow [0..<n]. \text{Const } (a i) * PVar i)$
 $\wedge c \geq 0 \wedge (\forall i < n. a i > 0)$

begin

lemma *solution: positive-poly-problem* *p q*
 $\langle proof \rangle$
end

locale *lin-term-poly-input* = *poly-input* *p q* **for** *p q* +
assumes *lin-term*: *termination-by-linear-int-poly-interpretation* *F-R R*
begin

definition *I* **where** $I = (\text{SOME } I. \text{linear-int-poly-inter } F-R I \wedge \text{int-poly-inter.termination-by-poly-interpretation } F-R I R)$

lemma *I*: *linear-int-poly-inter* *F-R I* *int-poly-inter.termination-by-poly-interpretation* *F-R I R*
 $\langle proof \rangle$

sublocale *linear-int-poly-inter* *F-R I* $\langle proof \rangle$

lemma *orient*: *orient-rule* (*lhs-R,rhs-R*)
 $\langle proof \rangle$

lemma *extract-linear-poly*: **assumes** $(g,n) \in F-R$
shows $\exists c a. I g = \text{Const } c + (\sum i \leftarrow [0..<n]. \text{Const } (a i) * PVar i)$
 $\wedge c \geq 0 \wedge (\forall i < n. a i > 0)$
 $\langle proof \rangle$

lemma *solution: positive-poly-problem* *p q*
 $\langle proof \rangle$
end

locale *wm-lin-orient-poly-input* = *poly-input* *p q* **for** *p q* +
assumes *wm-orient*: *orientation-by-linear-wm-int-poly-interpretation* *F-R R'*
begin

definition *I* **where** $I = (\text{SOME } I. \text{linear-wm-int-poly-inter } F-R I \wedge \text{wm-int-poly-inter.oriented-by-interpretation } F-R I R')$

lemma *I*: *linear-wm-int-poly-inter* *F-R I* *wm-int-poly-inter.oriented-by-interpretation* *F-R I R'*
 $\langle proof \rangle$

sublocale *linear-wm-int-poly-inter* $F-R$ I \langle *proof* \rangle

lemma *orient-R'*: *orient-rule* (*lhs-R'*,*rhs-R'*)
 \langle *proof* \rangle

lemma *extract-linear-poly*: **assumes** $g: (g,n) \in F-R$
shows $\exists c a. I g = \text{Const } c + (\sum_{i \leftarrow [0..<n]}. \text{Const } (a i) * \text{PVar } i)$
 $\wedge c \geq 0 \wedge (\forall i < n. a i \geq 0)$
 \langle *proof* \rangle

lemma *extract-a-poly*: $\exists a0 a1 a2. I a\text{-sym} = \text{Const } a0 + \text{Const } a1 * \text{PVar } 0 +$
 $\text{Const } a2 * \text{PVar } 1$
 $\wedge a0 \geq 0 \wedge a1 \geq 0 \wedge a2 \geq 0$
 \langle *proof* \rangle

lemma *extract-f-poly*: $\exists f0 f1 f2 f3 f4. I f\text{-sym} = \text{Const } f0 + \text{Const } f1 * \text{PVar } 0$
 $+ \text{Const } f2 * \text{PVar } 1$
 $+ \text{Const } f3 * \text{PVar } 2 + \text{Const } f4 * \text{PVar } 3$
 $\wedge f0 \geq 0 \wedge f1 \geq 0 \wedge f2 \geq 0 \wedge f3 \geq 0 \wedge f4 \geq 0$
 \langle *proof* \rangle

lemma *solution: positive-poly-problem* p q
 \langle *proof* \rangle
end

context *poly-input*
begin

Theorem 3.4 in paper

theorem *linear-polynomial-termination-with-natural-numbers-undecidable*:
positive-poly-problem p $q \iff$ *termination-by-linear-int-poly-interpretation* $F-R$
 R
 \langle *proof* \rangle

Theorem 3.9

theorem *orientation-by-linear-wm-int-poly-interpretation-undecidable*:
positive-poly-problem p $q \iff$ *orientation-by-linear-wm-int-poly-interpretation* $F-R$
 R'
 \langle *proof* \rangle

end

Separate locale to define another interpretation, i.e., the one of Lemma 3.6

locale *poly-input-non-lin-solution* = *poly-input*
begin

Non-linear interpretation of Lemma 3.6

fun $I :: \text{symbol} \Rightarrow \text{int mpoly}$ **where**

```

  | f-sym = PVar 2 * PVar 3 + PVar 0 + PVar 1 + PVar 2 + PVar 3
  | a-sym = PVar 0 + PVar 1
  | z-sym = 0
  | o-sym = Const (1 + insertion (λ -. 1) q)
  | (v-sym i) = PVar 0

```

sublocale *inter-R*: *poly-inter F-R I (>)* *<proof>*

lemma *inter-encode-num*: **assumes** $c \geq 0$
shows $\text{inter-R.eval (encode-num } x \ c) = \text{Const } c * \text{PVar } x$
<proof>

lemma *inter-v-pow-e*: $\text{inter-R.eval } ((v\text{-}t \ x \ \rightsquigarrow \ e) \ t) = \text{inter-R.eval } t$
<proof>

lemma *inter-encode-monom*: **assumes** $c: c \geq 0$
shows $\text{inter-R.eval (encode-monom } y \ m \ c) = \text{Const (insertion } (\lambda \ .1) \ (\text{monom } m \ c)) * \text{PVar } y$
<proof>

lemma *inter-encode-poly*: **assumes** *positive-poly r*
shows $\text{inter-R.eval (encode-poly } x \ r) = \text{Const (insertion } (\lambda \ .1) \ r) * \text{PVar } x$
<proof>

lemma *valid-monotone-inter*: $\text{inter-R.valid-monotone-poly-inter}$
<proof>

Lemma 3.6 in the paper

lemma *orient-R-main*: **assumes** *assignment β*
shows $\text{insertion } \beta \ (\text{inter-R.eval lhs-R}) > \text{insertion } \beta \ (\text{inter-R.eval rhs-R})$
<proof>

lemma *polynomial-termination-R*: *termination-by-int-poly-interpretation F-R R*
<proof>

lemma *polynomial-termination-R'*: *termination-by-int-poly-interpretation F-R R'*
<proof>

end
end

6 Undecidability of KBO with Subterm Coefficients

```

theory KBO-Subterm-Coefficients-Undecidable
  imports
    Hilbert10-to-Inequality
    Knuth-Bendix-Order.KBO
    Linear-Poly-Termination-Undecidable
begin

```

lemma *count-sum-list*: $\text{count } (\text{sum-list } ms) x = \text{sum-list } (\text{map } (\lambda m. \text{count } m x) ms)$
 ⟨proof⟩

lemma *sum-list-scf-list-prod*: $\text{sum-list } (\text{map } f (\text{scf-list } scf as)) = \text{sum-list } (\text{map } (\lambda i. \text{scf } i * f (as ! i)) [0..<\text{length } as])$
 ⟨proof⟩

lemma *count-vars-term-different-var*: **assumes** $x \notin \text{vars-term } t$
shows $\text{count } (\text{vars-term-ms } (\text{scf-term } scf t)) x = 0$
 ⟨proof⟩

context *kbo*

begin

definition *kbo-orientation* :: $(f, v)\text{rule set} \Rightarrow \text{bool}$ **where**

$kbo\text{-orientation } R = (\forall (l, r) \in R. \text{fst } (kbo l r))$

end

definition *kbo-with-sc-termination* :: $(f, v)\text{rule set} \Rightarrow \text{bool}$ **where**

$kbo\text{-with-sc-termination } R = (\exists w w0 sc \text{ least } pr\text{-strict } pr\text{-weak. } \text{admissible-kbo } w w0 \text{ } pr\text{-strict } pr\text{-weak } \text{least } sc$

$\wedge kbo.kbo\text{-orientation } w w0 sc \text{ least } pr\text{-strict } pr\text{-weak } R)$

context *poly-input*

begin

context

fixes *sc*

assumes $sc (a\text{-sym}, \text{Suc } (\text{Suc } 0)) 0 = (1 :: \text{nat})$

$sc (a\text{-sym}, \text{Suc } (\text{Suc } 0)) (\text{Suc } 0) = 1$

begin

lemma *count-vars-term-encode-num-nat*:

$\text{count } (\text{vars-term-ms } (\text{scf-term } sc (\text{encode-num } x (\text{int } n)))) x = n$

⟨proof⟩

lemma *count-vars-term-encode-num*:

$c \geq 0 \implies \text{int } (\text{count } (\text{vars-term-ms } (\text{scf-term } sc (\text{encode-num } x c)))) x = c$

⟨proof⟩

lemma *count-vars-term-v-pow-e*:

$\text{count } (\text{vars-term-ms } (\text{scf-term } sc ((v\text{-t } x \hat{\sim} e) t))) y$

$= (sc (v\text{-sym } x, 1) 0) \hat{\sim} e * \text{count } (\text{vars-term-ms } (\text{scf-term } sc t)) y$

⟨proof⟩

lemma *count-vars-term-encode-monom*: **assumes** $c \geq 0$

shows $\text{int } (\text{count } (\text{vars-term-ms } (\text{scf-term } sc (\text{encode-monom } x m c)))) x$

$= \text{insertion } (\lambda v. \text{int } (sc (v\text{-sym } v, 1) 0)) (\text{monom } m c)$

<proof>

Lemma 4.5

lemma *count-vars-term-encode-poly-generic*: **assumes** *positive-poly r*
shows $\text{int } (\text{count } (\text{vars-term-ms } (\text{scf-term } \text{sc } (\text{encode-poly } x \ r))) \ x) =$
 $\text{insertion } (\lambda \ v. \ \text{int } (\text{sc } (\text{v-sym } v, 1) \ 0)) \ r$
<proof>
end

Theorem 4.6

theorem *kbo-sc-termination-R-imp-solution*:
assumes *kbo-with-sc-termination R*
shows *positive-poly-problem p q*
<proof>
end

context *solvable-poly-problem*
begin

definition $w0 :: \text{nat}$ **where** $w0 = 1$

fun $\text{sc} :: \text{symbol} \times \text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat}$ **where**
 $\text{sc } (\text{v-sym } i, \ \text{Suc } 0) \ - = \text{nat } (\alpha \ i)$
 $|\ \text{sc } _ \ - = 1$

context **fixes** $wr :: \text{nat}$
begin

fun $w\text{-}R :: \text{symbol} \times \text{nat} \Rightarrow \text{nat}$ **where**
 $w\text{-}R \ (\text{f-sym}, n) = (\text{if } n = 4 \ \text{then } 0 \ \text{else } 1)$
 $|\ w\text{-}R \ (\text{a-sym}, n) = (\text{if } n = 2 \ \text{then } 0 \ \text{else } 1)$
 $|\ w\text{-}R \ (\text{o-sym}, 0) = wr$
 $|\ w\text{-}R \ _ = 1$
end

definition $w\text{-}rhs$ **where** $w\text{-}rhs = \text{weight-fun.weight } (w\text{-}R \ 1) \ w0 \ \text{sc } rhs\text{-}R$

abbreviation w **where** $w \equiv w\text{-}R \ w\text{-}rhs$

definition $least$ **where** $least \ f = (w \ (f, \ 0) = w0 \ \wedge \ (\forall \ g. \ w \ (g, \ 0) = w0 \ \longrightarrow \ (g,$
 $0 :: \text{nat}) = (f, \ 0)))$

lemma $\alpha 0: \alpha \ x > 0$ *<proof>*

sublocale *admissible-kbo* $w \ w0 \ (\lambda \ _ \ . \ \text{False}) \ (=) \ least \ \text{sc}$
<proof>

lemma *insertion-pos*: $\text{positive-poly } r \Longrightarrow \text{insertion } \alpha \ r \geq 0$
<proof>

lemma *count-vars-term-encode-poly*: **assumes** *positive-poly r*
shows *count (vars-term-ms (SCF (encode-poly x r))) y = (nat (insertion α r))*
when $x = y$
<proof>

Theorem 4.7 in context

theorem *kbo-with-sc-termination*: *kbo-with-sc-termination R*
<proof>

end

Theorem 4.7 outside solvable-context

context *poly-input*

begin

theorem *solvable-imp-kbo-with-sc-termination*:

assumes *positive-poly-problem p q*

shows *kbo-with-sc-termination R*

<proof>

Combining 4.6 and 4.7

corollary *solvable-iff-kbo-with-sc-termination*:

positive-poly-problem p q \longleftrightarrow kbo-with-sc-termination R

<proof>

end

end

7 Undecidability of Polynomial Termination over Integers

theory *Poly-Termination-Undecidable*

imports

Linear-Poly-Termination-Undecidable

Preliminaries-on-Polynomials-2

begin

context *poly-input*

begin

definition *y4* :: *var* **where** *y4 = 3*

definition *y5* :: *var* **where** *y5 = 4*

definition *y6* :: *var* **where** *y6 = 5*

definition *y7* :: *var* **where** *y7 = 6*

abbreviation *q-t* **where** *q-t t \equiv Fun q-sym [t]*

abbreviation *h-t* **where** *h-t t \equiv Fun h-sym [t]*

abbreviation *g-t* **where** *g-t t1 t2 \equiv Fun g-sym [t1, t2]*

Definition 5.1

definition $lhs-S = Fun\ f\text{-sym}$ [
Var $y1$,
Var $y2$,
a-t (*encode-poly* $y3\ p$) (*Var* $y3$),
q-t (*h-t* (*Var* $y4$)),
h-t (*Var* $y5$),
h-t (*Var* $y6$),
g-t (*Var* $y7$) *o-t*]

definition $rhs-S = Fun\ f\text{-sym}$ [
a-t (*Var* $y1$) *z-t*,
a-t *z-t* (*Var* $y2$),
a-t (*encode-poly* $y3\ q$) (*Var* $y3$),
h-t (*h-t* (*q-t* (*Var* $y4$))),
foldr *v-t* *V-list* (*a-t* (*Var* $y5$) (*Var* $y5$)),
Fun *f-sym* (*replicate* γ (*Var* $y6$)),
g-t (*Var* $y7$) *z-t*]

definition S **where** $S = \{(lhs-S, rhs-S)\}$

definition $F-S$ **where** $F-S = \{(f\text{-sym}, \gamma), (h\text{-sym}, 1), (g\text{-sym}, 2), (o\text{-sym}, 0), (q\text{-sym}, 1)\} \cup F$

lemma $lhs-S-F$: *funas-term* $lhs-S \subseteq F-S$
<proof>

lemma $funas-fold-vs[simp]$: *funas-term* (*foldr* *v-t* *V-list* t) = $(\lambda\ i.\ (v\text{-sym}\ i, 1))\ 'V \cup funas-term\ t$
<proof>

lemma $vars-fold-vs[simp]$: *vars-term* (*foldr* *v-t* *vs* t) = *vars-term* t
<proof>

lemma $funas-term-r5$: *funas-term* (*foldr* *v-t* *V-list* (*a-t* (*Var* $y5$) (*Var* $y5$))) $\subseteq F-S$
<proof>

lemma $rhs-S-F$: *funas-term* $rhs-S \subseteq F-S$
<proof>
end

lemma $poly-inter-eval-cong$: **assumes** $\bigwedge f\ a.\ (f, a) \in funas-term\ t \implies I\ f = I'\ f$
shows $poly-inter.eval\ I\ t = poly-inter.eval\ I'\ t$
<proof>

The easy direction of Theorem 5.4

context *solvable-poly-problem*
begin

definition $c\text{-}S$ where $c\text{-}S = \max 7 (2 * \text{prod-list} (\text{map } \alpha V\text{-list}))$

lemma $c\text{-}S$: $c\text{-}S > 0$ $\langle \text{proof} \rangle$

fun $I\text{-}S :: \text{symbol} \Rightarrow \text{int mpoly}$ **where**

$I\text{-}S f\text{-sym} = \text{PVar } 0 + \text{PVar } 1 + \text{PVar } 2 + \text{PVar } 3 + \text{PVar } 4 + \text{PVar } 5 + \text{PVar } 6$

| $I\text{-}S a\text{-sym} = \text{PVar } 0 + \text{PVar } 1$

| $I\text{-}S z\text{-sym} = 0$

| $I\text{-}S o\text{-sym} = 1$

| $I\text{-}S (v\text{-sym } i) = \text{Const } (\alpha i) * \text{PVar } 0$

| $I\text{-}S q\text{-sym} = \text{mmonom} (\text{monomial } 2 0) c\text{-}S - c * (\text{PVar } 0)^2$

| $I\text{-}S g\text{-sym} = \text{PVar } 0 + \text{PVar } 1$

| $I\text{-}S h\text{-sym} = \text{mmonom} (\text{monomial } 1 0) c\text{-}S - c * \text{PVar } 0$

declare $\text{single-numeral}[\text{simp del}]$

declare $\text{insertion-monom}[\text{simp del}]$

interpretation $\text{inter-}S$: $\text{poly-inter } F\text{-}S I\text{-}S (>)$ $\langle \text{proof} \rangle$

lemma $\text{inter-}S\text{-encode-poly}$: **assumes** $\text{positive-poly } r$

shows $\text{inter-}S.\text{eval} (\text{encode-poly } x r) = \text{Const} (\text{insertion } \alpha r) * \text{PVar } x$
 $\langle \text{proof} \rangle$

lemma $\text{valid-monotone-inter-}S$: $\text{inter-}S.\text{valid-monotone-poly-inter}$
 $\langle \text{proof} \rangle$

interpretation $\text{inter-}S$: $\text{int-poly-inter } F\text{-}S I\text{-}S$
 $\langle \text{proof} \rangle$

lemma orient-trs : $\text{inter-}S.\text{termination-by-poly-interpretation } S$
 $\langle \text{proof} \rangle$

lemma $\text{solution-imp-poly-termination}$: $\text{termination-by-int-poly-interpretation } F\text{-}S$
 S
 $\langle \text{proof} \rangle$

end

Towards Lemma 5.2

lemma (**in** int-poly-inter) $\text{monotone-imp-weakly-monotone}$: **assumes** monotone-poly
 $xs p$

shows $\text{weakly-monotone-poly } xs p$
 $\langle \text{proof} \rangle$

context

fixes $gt :: 'a :: \text{linordered-idom} \Rightarrow 'a \Rightarrow \text{bool}$

assumes trans-gt : $\text{transp } gt$

and gt-imp-ge : $\bigwedge x y. gt x y \implies x \geq y$

begin

lemma *monotone-poly-wrt-insertion-main*: **assumes** *monotone-poly-wrt gt xs p*
 and *a*: *assignment* ($a :: \text{var} \Rightarrow 'a :: \text{linordered-idom}$)
 and *b*: $\bigwedge x. x \in xs \implies \text{gt}^{==} (b\ x) (a\ x)$
 $\bigwedge x. x \notin xs \implies a\ x = b\ x$
 shows $\text{gt}^{==} (\text{insertion } b\ p) (\text{insertion } a\ p)$
<proof>

lemma *monotone-poly-wrt-insertion*: **assumes** *monotone-poly-wrt gt (vars p) p*
 and *a*: *assignment* ($a :: \text{var} \Rightarrow 'a :: \text{linordered-idom}$)
 and *b*: $\bigwedge x. x \in \text{vars } p \implies \text{gt}^{==} (b\ x) (a\ x)$
shows $\text{gt}^{==} (\text{insertion } b\ p) (\text{insertion } a\ p)$
<proof>

lemma *partial-insertion-mono-wrt*: **assumes** *mono: monotone-poly-wrt gt (vars p) p*
 and *a*: *assignment* *a*
 and *b*: $\bigwedge y. y \neq x \implies \text{gt}^{==} (b\ y) (a\ y)$
 and *d*: $\bigwedge y. y \geq d \implies \text{gt}^{==} y\ 0$
 shows $\exists c. \forall y. y \geq d \longrightarrow c \leq \text{poly } (\text{partial-insertion } a\ x\ p)\ y$
 $\wedge \text{poly } (\text{partial-insertion } a\ x\ p)\ y \leq \text{poly } (\text{partial-insertion } b\ x\ p)\ y$
<proof>

context

assumes *poly-pinfty-ge*: $\bigwedge p\ b. 0 < \text{lead-coeff } (p :: 'a\ \text{poly}) \implies \text{degree } p \neq 0$
 $\implies \exists n. \forall x \geq n. b \leq \text{poly } p\ x$

begin

context

fixes *p d*
 assumes *mono: monotone-poly-wrt gt (vars p) p*
 and *d*: $\bigwedge y. y \geq d \implies \text{gt}^{==} y\ 0$

begin

lemma *degree-partial-insertion-mono-generic*: **assumes**

a: *assignment* *a*
 and *b*: $\bigwedge y. y \neq x \implies \text{gt}^{==} (b\ y) (a\ y)$
 shows $\text{degree } (\text{partial-insertion } a\ x\ p) \leq \text{degree } (\text{partial-insertion } b\ x\ p)$
<proof>

lemma *degree-partial-insertion-stays-constant-generic*:

$\exists a. \text{assignment } a \wedge$
 $(\forall b. (\forall y. \text{gt}^{==} (b\ y) (a\ y)) \longrightarrow \text{degree } (\text{partial-insertion } a\ x\ p) = \text{degree } (\text{partial-insertion } b\ x\ p))$
<proof>
end

lemma *monotone-poly-partial-insertion-generic*:

assumes *delta-order*: $\bigwedge x y. \text{gt } y x \longleftrightarrow y \geq x + \delta$
and *delta*: $\delta > 0$
and *eps-delta*: $\varepsilon * \delta \geq 1$
and *ceil-nat*: $\bigwedge x :: 'a. \text{of-nat } (\text{ceil-nat } x) \geq x$
assumes *x*: $x \in xs$
and *mono*: *monotone-poly-wrt gt xs p*
and *ass*: *assignment a*
shows $0 < \text{degree } (\text{partial-insertion } a \ x \ p)$
lead-coeff $(\text{partial-insertion } a \ x \ p) > 0$
valid-poly $p \implies \text{poly } (\text{partial-insertion } a \ x \ p) (\delta * \text{of-nat } y) \geq \delta * \text{of-nat } y$
<proof>
end
end

context *poly-inter*
begin

lemma *monotone-poly-eval-generic*:
assumes *valid*: *valid-monotone-poly-inter*
and *trans-gt*: *transp* (\succ)
and *gt-imp-ge*: $\bigwedge x y. x \succ y \implies y \leq x$
and *gt-exists*: $\bigwedge x. x \geq 0 \implies \exists y. y \succ x$
and *gt-irrefl*: $\bigwedge x. \neg (x \succ x)$
and *tF*: *funas-term* $t \subseteq F$
shows *monotone-poly* $(\text{vars-term } t) (\text{eval } t) \text{ vars } (\text{eval } t) = \text{vars-term } t$
<proof>
end

context *int-poly-inter*
begin

lemma *degree-mono*: **assumes** *pos*: *lead-coeff* $p \geq (0 :: \text{int})$
and *le*: $\bigwedge x. x \geq c \implies \text{poly } p \ x \leq \text{poly } q \ x$
shows *degree* $p \leq \text{degree } q$
<proof>

lemma *degree-mono'*: **assumes** $\bigwedge x. x \geq c \implies (\text{bnd} :: \text{int}) \leq \text{poly } p \ x \wedge \text{poly } q \ x$
 $\leq \text{poly } q \ x$
shows *degree* $p \leq \text{degree } q$
<proof>

lemma *weakly-monotone-insertion*: **assumes** *weakly-monotone-poly* $(\text{vars } p) \ p$
and *assignment* $(a :: - \Rightarrow \text{int})$
and $\bigwedge x. x \in \text{vars } p \implies a \ x \leq b \ x$
shows *insertion* $a \ p \leq \text{insertion } b \ p$
<proof>

Lemma 5.2

lemma *degree-partial-insertion-stays-constant*: **assumes** *mono*: *monotone-poly* (*vars p*) *p*
shows $\exists a. \text{assignment } (a :: - \Rightarrow \text{int}) \wedge$
 $(\forall b. (\forall y. a\ y \leq b\ y) \longrightarrow \text{degree } (\text{partial-insertion } a\ x\ p) = \text{degree } (\text{partial-insertion } b\ x\ p))$
 $\langle \text{proof} \rangle$

lemma *degree-partial-insertion-stays-constant-wm*: **assumes** *wm*: *weakly-monotone-poly* (*vars p*) *p*
shows $\exists a. \text{assignment } (a :: - \Rightarrow \text{int}) \wedge$
 $(\forall b. (\forall y. a\ y \leq b\ y) \longrightarrow \text{degree } (\text{partial-insertion } a\ x\ p) = \text{degree } (\text{partial-insertion } b\ x\ p))$
 $\langle \text{proof} \rangle$

Lemma 5.3

lemma *subst-same-var-weakly-monotone-imp-same-degree*:
assumes *wm*: *weakly-monotone-poly* (*vars p*) (*p* :: *int mpoly*)
and *qp*: *poly-to-mpoly* *x q = substitute* ($\lambda i. \text{PVar } x$) *p*
shows *total-degree p = degree q*
 $\langle \text{proof} \rangle$

lemma *monotone-poly-partial-insertion*:
assumes *x*: $x \in xs$
and *mono*: *monotone-poly* *xs p*
and *ass*: *assignment a*
shows $0 < \text{degree } (\text{partial-insertion } a\ x\ p)$
 $\text{lead-coeff } (\text{partial-insertion } a\ x\ p) > 0$
 $\text{valid-poly } p \Longrightarrow y \geq 0 \Longrightarrow \text{poly } (\text{partial-insertion } a\ x\ p)\ y \geq y$
 $\text{valid-poly } p \Longrightarrow \text{insertion } a\ p \geq a\ x$
 $\langle \text{proof} \rangle$

end

context *int-poly-inter*
begin

lemma *insertion-eval-pos*: **assumes** *funas-term t* $\subseteq F$
and *assignment* α
shows $\text{insertion } \alpha\ (\text{eval } t) \geq 0$
 $\langle \text{proof} \rangle$

lemma *monotone-poly-eval*: **assumes** *funas-term t* $\subseteq F$
shows $\text{monotone-poly } (\text{vars-term } t)\ (\text{eval } t)\ \text{vars } (\text{eval } t) = \text{vars-term } t$
 $\langle \text{proof} \rangle$
end

locale *term-poly-input = poly-input p q* **for** *p q* +
assumes *terminating-poly*: *termination-by-int-poly-interpretation F-S S*

begin

definition *I* **where** $I = (\text{SOME } I. \text{int-poly-inter } F\text{-S } I \wedge \text{int-poly-inter.termination-by-poly-interpretation } F\text{-S } I\ S)$

lemma *I*: $\text{int-poly-inter } F\text{-S } I \text{ int-poly-inter.termination-by-poly-interpretation } F\text{-S } I\ S$
<proof>

sublocale *int-poly-inter F-S I <proof>*

lemma *orient: orient-rule (lhs-S,rhs-S)*
<proof>

lemma *solution: positive-poly-problem p q*
<proof>
end

context *poly-input*
begin

Theorem 5.4 in paper

theorem *polynomial-termination-with-natural-numbers-undecidable:*
positive-poly-problem p q \longleftrightarrow termination-by-int-poly-interpretation F-S S
<proof>

end

Now head for Lemma 5.6

locale *poly-input-omega-solution = poly-input*
begin

fun *I :: symbol \Rightarrow int list \Rightarrow int* **where**
 I o-sym xs = insertion (λ -. 1) q
 | I z-sym xs = 0
 | I a-sym xs = xs ! 0 + xs ! 1
 *| I g-sym xs = (xs ! 1 + 1) * xs ! 0 + xs ! 1*
 *| I h-sym xs = (xs ! 0)² + 7 * (xs ! 0) + 4*
 *| I f-sym xs = xs ! 2 * xs ! 6 + sum-list xs*
 | I q-sym xs = 5^{(nat (xs ! 0))}
 | I (v-sym i) xs = xs ! 0

lemma *I-encode-num: assumes c \geq 0*
 shows $I[\text{encode-num } x\ c]\alpha = c * \alpha\ x$
<proof>

lemma *I-v-pow-e: I [(v-t x \sim e) t]\alpha = I [t]\alpha*
<proof>

lemma *I-encode-monom*: **assumes** $c: c \geq 0$
shows $I[\text{encode-monom } x \ m \ c]\alpha = c * \alpha \ x$
 $\langle \text{proof} \rangle$

lemma *I-encode-poly*: **assumes** *positive-poly* r
shows $I[\text{encode-poly } x \ r]\alpha = \text{insertion } (\lambda \cdot. 1) \ r * \alpha \ x$
 $\langle \text{proof} \rangle$
end

lemma *length2-cases*: $\text{length } xs = 2 \implies \exists \ x \ y. \ xs = [x, y]$
 $\langle \text{proof} \rangle$

lemma *length7-cases*: $\text{length } xs = 7 \implies \exists \ x1 \ x2 \ x3 \ x4 \ x5 \ x6 \ x7. \ xs = [x1, x2, x3, x4, x5, x6, x7]$
 $\langle \text{proof} \rangle$

lemma *length1-cases*: $\text{length } xs = \text{Suc } 0 \implies \exists \ x. \ xs = [x]$
 $\langle \text{proof} \rangle$

lemma *less2-cases*: $i < 2 \implies i = 0 \vee (i :: \text{nat}) = 1$
 $\langle \text{proof} \rangle$

lemma *less7-cases*: $i < 7 \implies i = 0 \vee (i :: \text{nat}) = 1 \vee i = 2 \vee i = 3 \vee i = 4$
 $\vee i = 5 \vee i = 6$
 $\langle \text{proof} \rangle$

context *poly-input-omega-solution*
begin

sublocale *inter-S*: *term-algebra* $F\text{-}S \ I \ (>) \ \langle \text{proof} \rangle$
sublocale *inter-S*: *omega-term-algebra* $F\text{-}S \ I$
 $\langle \text{proof} \rangle$

Lemma 5.6

lemma *S-is-omega-terminating*: *omega-termination* $F\text{-}S \ S$
 $\langle \text{proof} \rangle$
end

end

8 Undecidability of Polynomial Termination using δ -Orders

theory *Delta-Poly-Termination-Undecidable*
imports
Poly-Termination-Undecidable
begin

context *poly-input*

begin

definition $y8 :: var$ **where** $y8 = 7$

definition $y9 :: var$ **where** $y9 = 8$

Definition 6.3

definition $lhs-Q = Fun\ f\text{-}sym$ [
 $q\text{-}t\ (h\text{-}t\ (Var\ y1))$,
 $h\text{-}t\ (Var\ y2)$,
 $h\text{-}t\ (Var\ y3)$,
 $g\text{-}t\ (q\text{-}t\ (Var\ y4))\ (h\text{-}t\ (h\text{-}t\ (h\text{-}t\ (Var\ y4))))$,
 $q\text{-}t\ (Var\ y5)$,
 $a\text{-}t\ (Var\ y6)\ (Var\ y6)$,
 $Var\ y7$,
 $Var\ y8$,
 $h\text{-}t\ (a\text{-}t\ (encode\text{-}poly\ y9\ p)\ (Var\ y9))$]

fun $g\text{-}list :: - \Rightarrow (symbol, var)term$ **where**
 $g\text{-}list\ [] = z\text{-}t$
 $| g\text{-}list\ ((f, n) \# fs) = g\text{-}t\ (Fun\ f\ (replicate\ n\ z\text{-}t))\ (g\text{-}list\ fs)$

definition $symbol\text{-}list$ **where** $symbol\text{-}list = [(f\text{-}sym, 9), (q\text{-}sym, 1), (h\text{-}sym, 1), (a\text{-}sym, 2)]$
 $@\ map\ (\lambda\ i.\ (v\text{-}sym\ i,\ 1))\ V\text{-}list$

definition $t\text{-}t :: (symbol, var)term$ **where** $t\text{-}t = (g\text{-}list\ ((z\text{-}sym, 0) \# symbol\text{-}list))$

definition $rhs-Q = Fun\ f\text{-}sym$ [
 $h\text{-}t\ (h\text{-}t\ (q\text{-}t\ (Var\ y1)))$,
 $g\text{-}t\ (Var\ y2)\ (Var\ y2)$,
 $Fun\ f\text{-}sym\ (replicate\ 9\ (Var\ y3))$,
 $q\text{-}t\ (g\text{-}t\ (Var\ y4)\ t\text{-}t)$,
 $a\text{-}t\ (Var\ y5)\ (Var\ y5)$,
 $q\text{-}t\ (Var\ y6)$,
 $a\text{-}t\ z\text{-}t\ (Var\ y7)$,
 $a\text{-}t\ (Var\ y8)\ z\text{-}t$,
 $a\text{-}t\ (encode\text{-}poly\ y9\ q)\ (Var\ y9)$]

definition Q **where** $Q = \{(lhs-Q, rhs-Q)\}$

definition $F-Q$ **where** $F-Q = \{(f\text{-}sym, 9), (h\text{-}sym, 1), (g\text{-}sym, 2), (q\text{-}sym, 1)\} \cup F$

lemma $lhs-Q\text{-}F$: $funas\text{-}term\ lhs-Q \subseteq F-Q$
<proof>

lemma $g\text{-}list\text{-}F$: $set\ zs \subseteq F-Q \implies funas\text{-}term\ (g\text{-}list\ zs) \subseteq F-Q$
<proof>

lemma $symbol\text{-}list$: $set\ symbol\text{-}list \subseteq F-Q$ *<proof>*

lemma *t-F*: *funas-term t-t* \subseteq *F-Q*
 ⟨*proof*⟩

lemma *vars-g-list[simp]*: *vars-term (g-list zs)* = {}
 ⟨*proof*⟩

lemma *vars-t*: *vars-term t-t* = {}
 ⟨*proof*⟩

lemma *rhs-Q-F*: *funas-term rhs-Q* \subseteq *F-Q*
 ⟨*proof*⟩

context

fixes *I* :: *symbol* \Rightarrow '*a* :: *linordered-field mpoly* **and** δ :: '*a* **and** *a3 a2 a1 a0 z0 v*
assumes *I*: *I a-sym* = *Const a3 * PVar 0 * PVar 1 + Const a2 * PVar 0 +*
*Const a1 * PVar 1 + Const a0*
I z-sym = *Const z0*
I (v-sym i) = *mpoly-of-poly 0 (v i)*
and *a*: *a3 > 0 a2 > 0 a1 > 0 a0 \geq 0*
and *z*: *z0 \geq 0*
and *v*: *nneg-poly (v i) degree (v i) > 0*

begin

lemma *nneg-combination*: **assumes** *nneg-poly r*
shows *nneg-poly ([:a1, a3:] * r + [:a0, a2:])*
 ⟨*proof*⟩

lemma *degree-combination*: **assumes** *nneg-poly r*
shows *degree ([:a1, a3:] * r + [:a0, a2:])* = *Suc (degree r)*
 ⟨*proof*⟩

lemma *degree-eval-encode-num*: **assumes** *c: c \geq 0*
shows \exists *p*. *mpoly-of-poly x p = poly-inter.eval I (encode-num x c) \wedge nneg-poly*
p \wedge int (degree p) = c
 ⟨*proof*⟩

lemma *degree-eval-encode-monom*: **assumes** *c: c > 0*
and α : $\alpha = (\lambda i. \text{int } (\text{degree } (v i)))$
shows \exists *p*. *mpoly-of-poly y p = poly-inter.eval I (encode-monom y m c) \wedge nneg-poly*
p \wedge
int (degree p) = insertion α (mmonom m c) \wedge degree p > 0
 ⟨*proof*⟩

Lemma 6.2

lemma *degree-eval-encode-poly-generic*: **assumes** *positive-poly r*
and α : $\alpha = (\lambda i. \text{int } (\text{degree } (v i)))$
shows \exists *p*. *poly-to-mpoly x p = poly-inter.eval I (encode-poly x r) \wedge nneg-poly p*
 \wedge

$int (degree\ p) = insertion\ \alpha\ r$
 <proof>
end
end

context *delta-poly-inter*
begin

lemma *transp-gt-delta*: $transp\ (\lambda\ x\ y.\ x \geq y + \delta)$ <proof>

lemma *gt-delta-imp-ge*: $y + \delta \leq x \implies y \leq x$ <proof>

lemma *weakly-monotone-insertion*: **assumes** *mono*: *monotone-poly* (*vars* *p*) *p*
and *a*: *assignment* ($a :: - \implies 'a$)
and *gt*: $\bigwedge x. x \in vars\ p \implies a\ x + \delta \leq b\ x$
shows $insertion\ a\ p \leq insertion\ b\ p$
 <proof>

Lemma 6.5

lemma *degree-partial-insertion-stays-constant*: **assumes** *mono*: *monotone-poly* (*vars* *p*) *p*
shows $\exists a. assignment\ a \wedge$
 $(\forall b. (\forall y. a\ y + \delta \leq b\ y) \longrightarrow degree\ (partial-insertion\ a\ x\ p) = degree$
 $(partial-insertion\ b\ x\ p))$
 <proof>

lemma *degree-mono*: **assumes** *pos*: $lead-coeff\ p \geq (0 :: 'a)$
and *le*: $\bigwedge x. x \geq c \implies poly\ p\ x \leq poly\ q\ x$
shows $degree\ p \leq degree\ q$
 <proof>

lemma *degree-mono'*: **assumes** $\bigwedge x. x \geq c \implies (bnd :: 'a) \leq poly\ p\ x \wedge poly\ p\ x$
 $\leq poly\ q\ x$
shows $degree\ p \leq degree\ q$
 <proof>

Lemma 6.6

lemma *subst-same-var-monotone-imp-same-degree*:
assumes *mono*: *monotone-poly* (*vars* *p*) (*p* :: $'a\ mpoly$)
and *qp*: *poly-to-mpoly* *x* *q* = *substitute* ($\lambda i. PVar\ x$) *p*
shows $total-degree\ p = degree\ q$
 <proof>

lemma *monotone-poly-partial-insertion*:
assumes *x*: $x \in xs$
and *mono*: *monotone-poly* *xs* *p*
and *ass*: *assignment* *a*
shows $0 < degree\ (partial-insertion\ a\ x\ p)$
 $lead-coeff\ (partial-insertion\ a\ x\ p) > 0$

```

    valid-poly p  $\implies$  y  $\geq$  0  $\implies$  poly (partial-insertion a x p) y  $\geq$  y -  $\delta$ 
    valid-poly p  $\implies$  insertion a p  $\geq$  a x -  $\delta$ 
  <proof>
end

context solvable-poly-problem
begin

context
  assumes SORT-CONSTRAINT('a :: floor-ceiling)
begin

context
  fixes h :: 'a
begin

fun IQ :: symbol  $\Rightarrow$  'a mpoly where
  IQ f-sym = PVar 0 + PVar 1 + PVar 2 + PVar 3 + PVar 4 + PVar 5 + PVar
6 + PVar 7 + PVar 8
| IQ a-sym = PVar 0 * PVar 1 + PVar 0 + PVar 1
| IQ z-sym = 0
| IQ (v-sym i) = PVar 0  $\wedge$  (nat ( $\alpha$  i))
| IQ q-sym = PVar 0 * PVar 0 + Const 2 * PVar 0
| IQ g-sym = PVar 0 + PVar 1
| IQ h-sym = Const h * PVar 0 + Const h
| IQ o-sym = 0

interpretation interQ: poly-inter F-Q IQ ( $\lambda$ x y. x  $\geq$  y + (1 :: 'a)) <proof>

Lemma 6.2 specialized for this interpretation

lemma degree-eval-encode-poly: assumes positive-poly r
  shows  $\exists$  p. poly-to-mpoly y9 p = interQ.eval (encode-poly y9 r)  $\wedge$  nneg-poly p  $\wedge$ 
int (degree p) = insertion  $\alpha$  r
  <proof>

definition pp where pp = (SOME pp. poly-to-mpoly y9 pp = interQ.eval (encode-poly
y9 p)  $\wedge$  nneg-poly pp  $\wedge$  int (degree pp) = insertion  $\alpha$  p)

lemma pp: interQ.eval (encode-poly y9 p) = poly-to-mpoly y9 pp
  nneg-poly pp int (degree pp) = insertion  $\alpha$  p
  <proof>

definition qq where qq = (SOME qq. poly-to-mpoly y9 qq = interQ.eval (encode-poly
y9 q)  $\wedge$  nneg-poly qq  $\wedge$  int (degree qq) = insertion  $\alpha$  q)

lemma qq: interQ.eval (encode-poly y9 q) = poly-to-mpoly y9 qq
  nneg-poly qq int (degree qq) = insertion  $\alpha$  q
  <proof>

```

definition $ppp = pp * [:1,1:] + [:0,1:]$

definition $qqq = qq * [:1,1:] + [:0,1:]$

lemma *degree-ppp*: $\text{int} (\text{degree } ppp) = 1 + \text{insertion } \alpha p$
<proof>

lemma *degree-qqq*: $\text{int} (\text{degree } qqq) = 1 + \text{insertion } \alpha q$
<proof>

lemma *ppp-qqq*: $\text{degree } ppp \geq \text{degree } qqq$
<proof>

lemma *nneg-ppp*: *nneg-poly ppp*
<proof>

definition *H* **where** $H = (\text{SOME } H. \forall h \geq H. \forall x \geq 0. \text{poly } qqq x \leq h * \text{poly } ppp x + h)$

lemma *H*: $h \geq H \implies x \geq 0 \implies \text{poly } qqq x \leq h * \text{poly } ppp x + h$
<proof>

end

definition *h* **where** $h = \max 9 (H 1)$

lemma *h*: $h \geq 1$ *<proof>*

abbreviation *I-Q* **where** $I-Q \equiv IQ h$

interpretation *inter-Q*: *poly-inter F-Q I-Q* $(\lambda x y. x \geq y + (1 :: 'a))$ *<proof>*

Well-definedness of Interpretation in Theorem 6.4

lemma *valid-monotone-inter-Q*:
inter-Q.valid-monotone-poly-inter
<proof>

lemma *I-Q-delta-poly-inter*: *delta-poly-inter F-Q I-Q* $(1 :: 'a)$
<proof>

interpretation *inter-Q*: *delta-poly-inter F-Q I-Q* $1 :: 'a$ *<proof>*

Orientation part of Theorem 6.4

lemma *orient-Q*: *inter-Q.orient-rule* $(\text{lhs-}Q, \text{rhs-}Q)$
<proof>

end

end

context *poly-input*

begin

Theorem 6.4

theorem *solution-impl-delta-termination-of-Q*:
assumes *positive-poly-problem* $p\ q$
shows *termination-by-delta-poly-interpretation* ($TYPE('a :: \text{floor-ceiling})$) $F\ Q$
 Q
 $\langle \text{proof} \rangle$

end

context *delta-poly-inter*
begin

lemma *insertion-eval-pos*: **assumes** *funas-term* $t \subseteq F$
and *assignment* α
shows *insertion* α ($\text{eval } t$) ≥ 0
 $\langle \text{proof} \rangle$

lemma *monotone-poly-eval*: **assumes** *funas-term* $t \subseteq F$
shows *monotone-poly* (*vars-term* t) ($\text{eval } t$) *vars* ($\text{eval } t$) = *vars-term* t
 $\langle \text{proof} \rangle$

lemma *monotone-linear-poly-to-coeffs*: **fixes** $p :: 'a\ \text{mpoly}$
assumes *linear*: *total-degree* $p \leq 1$
and *poly*: *valid-poly* p
and *mono*: *monotone-poly* $\{..<n\}$ p
and *vars*: *vars* $p = \{..<n\}$
shows $\exists\ c\ a.\ p = \text{Const } c + (\sum\ i \leftarrow [0..<n]. \text{Const } (a\ i) * \text{PVar } i)$
 $\wedge\ c \geq 0 \wedge (\forall\ i < n.\ a\ i \geq 1)$
 $\langle \text{proof} \rangle$

end

Lemma 6.7

lemma *criterion-for-degree-2*: **assumes** *qq-def*: $qq = q \circ_p [c, a] - \text{smult } a\ q$
and *dq*: *degree* $q \geq 2$
and *ineq*: $\bigwedge\ x :: 'a :: \text{linordered-field}.\ x \geq 0 \implies \text{poly } qq\ x \leq \text{poly } p\ x$
and *dp*: *degree* $p \leq 1$
and *a1*: $a \geq 1$
and *lq0*: *lead-coeff* $q > 0$
and *c*: $c > 0$
shows *degree* $q = 2\ a = 1$
 $\langle \text{proof} \rangle$

locale *term-delta-poly-input* = *poly-input* $p\ q$ **for** $p\ q +$
fixes *type-of-field* $:: 'a :: \text{floor-ceiling itself}$
assumes *terminating-delta-poly*: *termination-by-delta-poly-interpretation* $TYPE('a)$
 $F\ Q\ Q$

begin

definition *I* **where** $I = (\text{SOME } I. \exists \delta. \text{delta-poly-inter } F\text{-}Q\text{ } I (\delta :: 'a) \wedge$
 $\text{delta-poly-inter.termination-by-delta-interpretation } F\text{-}Q\text{ } I\ \delta\ Q)$

definition δ **where** $\delta = (\text{SOME } \delta. \text{delta-poly-inter } F\text{-}Q\text{ } I (\delta :: 'a) \wedge$
 $\text{delta-poly-inter.termination-by-delta-interpretation } F\text{-}Q\text{ } I\ \delta\ Q)$

lemma *I*: $\text{delta-poly-inter } F\text{-}Q\text{ } I\ \delta\ \text{delta-poly-inter.termination-by-delta-interpretation}$
 $F\text{-}Q\text{ } I\ \delta\ Q$
<proof>

sublocale $\text{delta-poly-inter } F\text{-}Q\text{ } I\ \delta\ \langle\text{proof}\rangle$

lemma *orient*: *orient-rule* (*lhs-Q*,*rhs-Q*)
<proof>

lemma *eval-t-t-gt-0*: **assumes** *Ig*: $I\ g\text{-sym} = \text{Const } g0 + \text{Const } g1 * \text{PVar } 0 +$
 $\text{Const } g2 * \text{PVar } 1$
and *Iz*: $I\ z\text{-sym} = \text{Const } z0$
and *z0*: $z0 \geq 0$
and *g0*: $g0 \geq 0$
and *g12*: $g1 > 0\ g2 > 0$
shows *insertion* $\beta\ (\text{eval } t\text{-}t) > 0$
<proof>

Theorem 6.8

theorem *solution*: *positive-poly-problem* *p* *q*
<proof>
end

context *poly-input*
begin

corollary *polynomial-termination-with-delta-orders-undecidable*:
positive-poly-problem *p* *q* \longleftrightarrow
termination-by-delta-poly-interpretation (*TYPE*('a :: *floor-ceiling*)) *F-Q* *Q*
<proof>

end

end

References

- [1] D. Lankford. On proving term rewrite systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.

- [2] Y. Y. Matijasevic. Enumerable sets are diophantine (translated from Russian). In *Soviet Mathematics Doklady*, volume 11, pages 354–358, 1970.
- [3] F. Mitterwallner, A. Middeldorp, and R. Thiemann. Linear termination is undecidable. In *Proceedings of the 39th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 2024. To appear.