Termination Analysis by Inductive Evaluation [⋆]

Jürgen Brauburger and Jürgen Giesl

FB Informatik, TU Darmstadt, Alexanderstraße 10, 64283 Darmstadt, Germany
E-mail: {brauburger, giesl}@informatik.tu-darmstadt.de

Abstract. We present a new approach for automatic termination analysis of functional programs. Several methods have been presented which try to find a well-founded ordering such that the arguments in the recursive calls are smaller than the corresponding inputs. However, previously developed approaches for automated termination analysis often disregard the *conditions* under which the recursive calls are evaluated. Hence, the existing methods fail for an important class of algorithms where the necessary information for proving termination is 'hidden' in the conditions. In this paper we develop the *inductive evaluation* method which analyzes the auxiliary functions occurring in the conditions of the recursive calls. We also discuss an extension of our method to *partial* functions in order to determine their domains automatically. The proposed technique proved successful for termination analysis of numerous algorithms in functional as well as imperative programming languages.

1 Introduction

Proving termination is a central problem in the development of correct software. While most work on the automation of termination proofs has been done for *term rewriting systems* (for surveys see e.g. [Der87, Ste95]) and for *logic programs* (e.g. [UvG88, Plü90, SD94]), in this paper we consider *functional programs.*

A functional program $f(x)$ is terminating if the arguments of each recursive call $f(r)$ decrease w.r.t. a well-founded ordering. A well-known method for termination proofs of LISP functions has been implemented in the NQTHM system of R. S. Boyer and J S. Moore [BM79]. In their approach, for each recursive call $f(r)$ an *induction lemma* $\Delta \rightarrow r \prec x$ is required, which asserts that the argument decreases w.r.t. a well-founded ordering $\prec$ if some condition Δ is satisfied. Then it remains to verify $\psi \rightarrow \Delta$ where ψ is the condition under which the recursive call $f(r)$ is performed. While in [BM79] the user has to supply all induction lemmata, the methods in [Wal94b, Gie95c, GWB98] synthesize a certain class of induction lemmata automatically. The technique in [Wal94b] is restricted to one fixed ordering $\prec$, but the approach of [Gie95c, GWB98] also allows an automatic generation of suitable well-founded orderings $\prec$ by incorporating techniques from the area of *term rewriting systems.*

To synthesize an induction lemma for a recursive call $f(r)$ under the condition ψ, these methods analyze the auxiliary functions occurring in the recursive argument r. However, auxiliary functions in the *condition* ψ are ignored at this point. Consequently, the previous approaches often fail if the necessary information for the termination proof is given by the functions called in the condition.

We illustrate this problem in Sect. 2. In Sect. 3 we present the *inductive evaluation* technique which overcomes this drawback by combining termination analysis with methods for *induction theorem proving.* While in Sect. 3 our aim is to show that a procedure terminates for *each* input, in Sect. 4 the method is generalized for termination analysis of functions which terminate for *some* inputs only. In Sect. 5 the techniques are extended for analysis of more complex procedures and Sect. 6 draws some conclusions. Finally, the appendix contains a collection of examples to demonstrate the power of our approach.

[⋆] Technical Report IBN-98-47, Technische Universität Darmstadt.
This work was supported by the Deutsche Forschungsgemeinschaft under grant no. Wa 652/7-2 as part of the focus program "Deduktion".

2 Termination of Functional Programs

We regard an eager first-order functional language with free algebraic data types[1] and pattern matching where the patterns have to be exhaustive and exclusive. As an example consider the data types bool and nat (for natural numbers). The type bool has the nullary *constructors* true and false and the objects of type nat are built with the constructors 0 and $\mathsf{s} : \mathsf{nat} \to \mathsf{nat}$. The following procedures compute the less than or equal relation for naturals and the subtraction function.

```
function le : nat × nat → bool          function minus : nat × nat → nat
  le(0, v)        = true                   minus(x, y) = if( le(x, y),
  le(s(u), 0)     = false                                   0,
  le(s(u), s(v))  = le(u, v)                                s(minus(x, s(y))) )
```

For each data type τ there is a pre-defined conditional function $\mathsf{if} : \mathsf{bool} \times \tau \times \tau \to \tau$. These conditionals are the only functions with non-eager semantics, i.e. when evaluating $\mathsf{if}(\psi, t_1, t_2)$, the boolean term[2] ψ is evaluated first and depending on the result of its evaluation either t_1 or t_2 is evaluated afterward yielding the result of the whole conditional.

An algorithm f is terminating if there is a well-founded relation $\prec$ such that the inputs are greater than the arguments of the recursive calls. For instance, termination of le can be shown by inventing a well-founded relation which satisfies the following *termination hypothesis* for le.

$$\langle u, v\rangle \prec \langle \mathsf{s}(u), \mathsf{s}(v)\rangle \tag{1}$$

In the area of term rewriting systems, several techniques have been developed for the generation of well-founded orderings. For example, termination of le can be proved by using a suitable *polynomial ordering* [Lan79]. A polynomial ordering $\prec$ is defined by mapping each n-ary constructor to an n-ary polynomial with integer coefficients. Then for two data objects r and t, we have $r \prec t$ iff the number corresponding to r is smaller than the number corresponding to t. We always demand that the choice of the coefficients ensures that data objects are only mapped to non-negative integers. For example, if the constructors 0 and $\mathsf{s}(u)$ are associated with the polynomials 0 and $u + 1$, respectively, then each data object of type nat corresponds to a natural number, and we have $u \prec \mathsf{s}(u)$ since $u < u + 1$ holds for each natural number u.

To compare *n-tuples* of terms we use an additional n-ary polynomial to map n-tuples of integers to one integer, where tuples of data objects must again be mapped to non-negative numbers. So if $\langle u, v\rangle$ is associated with $u + v$, then we have $\langle u, v\rangle \prec \langle \mathsf{s}(u), \mathsf{s}(v)\rangle$, as $u + v < u + 1 + v + 1$ holds for all naturals u and v.

The above polynomial ordering satisfies the termination hypothesis (1). Consequently, as all polynomial orderings are well founded, termination of le is proved. Techniques to generate suitable polynomial orderings automatically have been developed in [Ste94, Gie95a], for instance.

In the following, let t^* and r^* denote tuples of terms $t_1, .., t_n$ and $r_1, .., r_n$. To prove termination of a function f, for every recursive call in a defining equation $f(t^*) = \ldots f(r^*) \ldots$ we build a *termination hypothesis* $\psi \to r^* \prec t^*$ where ψ is the condition under which the recursive call $f(r^*)$ is evaluated. For instance, the recursive call of minus is evaluated under the condition $\neg\mathsf{le}(x, y)$. Thus we obtain the termination hypothesis

$$\neg\mathsf{le}(x, y) \to \langle x, \mathsf{s}(y)\rangle \prec \langle x, y\rangle. \tag{2}$$

Termination of a functional program f is proved if one finds a polynomial ordering that satisfies *all* termination hypotheses of f.

As the termination hypothesis (1) of le only contains terms built with constructors, a polynomial ordering satisfying it can easily be generated automatically. However, this is not possible for the termination hypothesis (2) of minus. The reason is that minus calls another algorithm, viz. le. In contrast to (1), the inequality $\langle t_1, \mathsf{s}(t_2)\rangle \prec \langle t_1, t_2\rangle$ in minus' termination hypothesis does not have

[1] See [NN96, PS97, Sen96] for extensions of termination analysis to higher-order languages, languages with lazy evaluation strategy, and to non-free algebraic data types.

[2] We use Greek letters to denote boolean terms and often refer to them as 'formulas', where $\neg, \wedge, \vee$, and $\to$ are pre-defined boolean functions with obvious semantics.

to be satisfied for *all* data objects t_1 and t_2, but only for those objects where $\neg\mathsf{le}(t_1, t_2)$ evaluates to true. To determine these data objects we have to consider the *semantics* of the algorithm le. However, the existing methods for the automated generation of polynomial orderings can only be used for termination proofs if the termination hypotheses do not contain any *defined* symbols, i.e. function symbols defined by algorithms.

3 Termination Proofs with Inductive Evaluation

To enable automatic termination proofs for algorithms like minus, in this section we develop a calculus which transforms termination hypotheses like (2) into formulas *without* defined symbols. Our calculus operates on pairs $H;\ C$ where H is a set of formulas (the hypotheses) possibly containing defined symbols and C is a set of inequalities (the constraints) without defined symbols. The soundness of our calculus guarantees that if $H;\ C$ can be transformed into $H';\ C'$, then every polynomial ordering satisfying $H' \cup C'$ also satisfies $H \cup C$.To use the calculus for termination proofs, we initialize H to be the set of termination hypotheses and we let C be empty. Then rules of the calculus are applied repeatedly until we result in a pair $H';\ C'$ where the first component H' is empty. By the soundness of the calculus, now it suffices to find a polynomial ordering satisfying the constraints C' in order to prove the termination of the algorithm.

An obvious solution to eliminate the defined symbol le from minus' termination hypothesis is to omit its premise, i.e. we could use the following rule[3].

Premise Elimination $$\frac{H \cup \{\psi \to \omega\}\ ;\ \ C}{H\ ;\ \ C \cup \{\omega\}}$$
if ω does not contain any defined symbols.

This is a sound transformation technique, because every polynomial ordering satisfying ω for *all* instantiations of its variables will also satisfy ω for those instantiations which meet the condition ψ, i.e. $\psi \to r^* \prec t^*$ may indeed be transformed into $r^* \prec t^*$. Moreover, we also allow the application of this rule if the premise ψ is missing, i.e. for an algorithm like le, the unconditional termination hypothesis (1) can be directly inserted into the set of constraints.

For the termination proof of minus, we would initialize H to be $\{(2)\}$ and C to be empty. Then one application of the Premise Elimination rule transforms H into the empty set and C into $\{\langle x, \mathsf{s}(y)\rangle \prec \langle x, y\rangle\}$. However, in our example this naive solution cannot be used, because there does not exist any well-founded ordering satisfying this constraint. Hence, the termination of minus cannot be proved if the premise of its termination hypothesis is neglected. For that reason, all previous approaches for automatic termination proofs fail with this example.

To enable termination proofs for algorithms like minus we now introduce a new rule which *evaluates* the auxiliary functions in the premises of termination hypotheses. To construct a set of constraints sufficient for the termination hypothesis (2) we use an *induction* w.r.t. the definition of the algorithm le. The base cases of this inductive construction correspond to le's non-recursive defining equations and the step case results from le's recursive (third) equation.

First we perform a case analysis w.r.t. le, i.e. the variables x, y in (2) are instantiated by the patterns of le's defining equations. Instead of (2) we demand

$$\neg\mathsf{le}(0, v) \ \to\ \langle 0, \mathsf{s}(v)\rangle \prec \langle 0, v\rangle, \tag{3}$$
$$\neg\mathsf{le}(\mathsf{s}(u), 0) \ \to\ \langle \mathsf{s}(u), \mathsf{s}(0)\rangle \prec \langle \mathsf{s}(u), 0\rangle, \tag{4}$$
$$\neg\mathsf{le}(\mathsf{s}(u), \mathsf{s}(v)) \ \to\ \langle \mathsf{s}(u), \mathsf{s}(\mathsf{s}(v))\rangle \prec \langle \mathsf{s}(u), \mathsf{s}(v)\rangle. \tag{5}$$

[3] In order to obtain constraints without defined symbols, this rule may only be applied if ω contains no calls of auxiliary algorithms. Hence, in this paper we restrict ourselves to termination hypotheses $\psi \to \omega$ where defined symbols may only occur in the *condition* ψ. For algorithms with defined symbols in the arguments of recursive calls, the technique of the present paper is extended by the calculus of [Gie95c, Gie97, GWB98] to eliminate the remaining defined symbols from the *conclusion* ω.

In order to detect redundant cases, in each resulting formula (3)-(5), we now check whether the premise is unsatisfiable. For example, the first case (3) may be omitted as the negated premise $\neg\neg\mathsf{le}(0, v)$ can be verified[4] by evaluation of the algorithms le and $\neg$. As the premises of the second and third case are satisfiable, the corresponding proofs must fail.

In the third case (5) we use that le is a terminating procedure, i.e. each le-call produces a finite sequence of recursive calls. Hence, we may assume as an *induction hypothesis* that the termination hypothesis (2) is true for the arguments u and v of the recursive call in le, i.e.

$$\neg\mathsf{le}(u, v) \;\rightarrow\; \langle u, \mathsf{s}(v)\rangle \prec \langle u, v\rangle. \tag{6}$$

In order to apply the induction hypothesis, we have to check whether the premise of the induction conclusion (5) entails the premise of the induction hypothesis (6), i.e. we have to prove the formula

$$\neg\mathsf{le}(\mathsf{s}(u), \mathsf{s}(v)) \rightarrow \neg\mathsf{le}(u, v). \tag{7}$$

Again, the proof is trivial since $\mathsf{le}(\mathsf{s}(u), \mathsf{s}(v))$ evaluates to $\mathsf{le}(u, v)$. For that reason we may now *apply* the induction hypothesis (6), i.e. instead of (5) we demand

$$\neg\mathsf{le}(\mathsf{s}(u), \mathsf{s}(v)) \;\wedge\; \langle u, \mathsf{s}(v)\rangle \prec \langle u, v\rangle \quad\rightarrow\quad \langle \mathsf{s}(u), \mathsf{s}(\mathsf{s}(v))\rangle \prec \langle \mathsf{s}(u), \mathsf{s}(v)\rangle. \tag{8}$$

The existing techniques for generating polynomial orderings expect a set of *inequalities* as ordering constraints, i.e. they cannot treat constraints like $r_1^* \prec t_1^* \rightarrow r_2^* \prec t_2^*$. To eliminate and to exploit the inequality in the premise of (8) we use that the occurring terms are interpreted as polynomials over integers which are compared by $<$. For arbitrary integers k, l, m, n the conjecture $[m + l \leq n + k] \rightarrow [k < l \rightarrow m < n]$ holds. Hence, instead of (8) we may demand

$$\neg\mathsf{le}(\mathsf{s}(u), \mathsf{s}(v)) \quad\rightarrow\quad \langle \mathsf{s}(u), \mathsf{s}(\mathsf{s}(v))\rangle + \langle u, v\rangle \;\preceq\; \langle \mathsf{s}(u), \mathsf{s}(v)\rangle + \langle u, \mathsf{s}(v)\rangle. \tag{9}$$

Here, $+$ is a new function symbol and we require that all polynomial orderings map $+$ to the addition. Moreover, for any polynomial ordering $\prec$ the corresponding non-strict polynomial ordering $\preceq$ is defined as $r \preceq t$ iff the number corresponding to r is smaller than or equal to the number corresponding to t.

In this way, the termination hypothesis (2) can be transformed into the formulas (4) and (9). By eliminating their premises (using the Premise Elimination rule), we obtain two constraints without defined symbols. Therefore we can now use the existing techniques to generate a polynomial ordering satisfying these constraints. For example, they are satisfied by the polynomial ordering where 0, $\mathsf{s}(u)$, and $\langle u, v\rangle$ are associated with 0, $u + 1$, and $(u - v)^2$, respectively. (Note that this is a legal polynomial ordering, because all tuples of data objects are mapped to non-negative numbers[5].) Hence, termination of minus is proved.

Recall that during the transformation of minus' termination hypothesis we had to verify the formulas $\neg\neg\mathsf{le}(0, v)$ and (7). To perform the required proofs, we simply applied *symbolic evaluation*, i.e. we used the defining equations as rewrite rules. In general this verification could require an *induction theorem proving system*, e.g. [BM79, ZKK88, Bun$^+$89, Wal94a, BR95, HS96]. However, when testing our method on numerous algorithms, we found that in almost all examples the required conjectures could already be proved by symbolic evaluation.

Let $\psi \rightarrow r^* \prec t^*$ be a termination hypothesis containing at least the pairwise different variables $y_1, \ldots, y_n$ of the data types $\tau_1, \ldots, \tau_n$. Moreover, let $g : \tau_1 \times \ldots \times \tau_n \rightarrow \tau$ be defined by a terminating algorithm with k defining equations. Then we use the following rule for induction and subsequent evaluation. In this rule, for any terms $p, s_1, \ldots, s_n$ let $p[s^*]$ be an abbreviation for

[4] Throughout the paper 'verification of a boolean term φ' means proving that φ evaluates to true for all instantiations of its variables with data objects.

[5] In contrast to conventional termination proofs of *term rewriting systems*, for *functional programs* one may use orderings which are not even weakly monotonic, cf. [AG97]. In fact, termination of minus cannot be proved by any monotonic well-founded ordering. For that reason, in our approach we restricted ourselves to the generation of polynomial orderings, as most other classes of orderings (that are amenable to automation) possess the monotonicity property.

$p[y_1/s_1, \ldots, y_n/s_n]$. Moreover, throughout the paper we always assume that the variables occurring in different algorithms are disjoint.

Inductive Evaluation $\dfrac{H \cup \{\psi \to r^* \prec t^*\}\ ;\ \ C}{H \cup \{\varphi_1, .., \varphi_k\}\ ;\ \ C}$

If $g(s^*) = q$ is the i-th defining equation of g, then φ_i is defined as follows:

- $\varphi_i :=$ true,	if $\neg\psi[s^*]$ can be verified,
- $\varphi_i := \psi[s^*] \to r^*[s^*] + t^*[q^*] \preceq t^*[s^*] + r^*[q^*]$,	otherwise, if q contains a term $g(q^*)$ where $\psi[s^*] \to \psi[q^*]$ can be verified,
- $\varphi_i := \psi[s^*] \to r^*[s^*] \prec t^*[s^*]$,	otherwise.

This rule performs a Noetherian induction, since it only allows inductions w.r.t. functions g whose termination has been proved before. To ease the presentation, in the above rule we restricted ourselves to functions g which are defined without using the conditional if (for an extension see Sect. 5).

For the choice of a suitable induction relation we use a well-known heuristic from the area of induction theorem proving. Given a termination hypothesis $\psi \to r^* \prec t^*$, we check whether ψ contains a subterm of the form $g(y_1, .., y_n)$ where the y_i are pairwise different variables. Such a term suggests a plausible induction w.r.t. g using $y_1, .., y_n$ as induction variables, cf. e.g. [BM79, ZKK88, Bun$^+$89, Wal94a]. Hence, for the termination hypothesis of minus this heuristic suggests inductive evaluation w.r.t. le using the induction variables x and y.

Inductive evaluation is used for algorithms where the *conditions* of recursive calls have to be analyzed in order to prove termination. In particular, this holds for algorithms like minus where some value is repeatedly increased until it reaches some bound. This class of algorithms is also used extensively in *imperative* programming languages. A straightforward approach to prove termination of imperative programs is to transform them into functional ones and to verify termination of the resulting functions, cf. e.g. [Hen80, GWB98]. For example, the imperative program '$r := 0;\ \ $ *while* $x > y$ *do* $y := y + 1;\ \ r := r + 1$ *od*' is transformed into a function whose termination can be proved analogously to minus. Hence, inductive evaluation is particularly useful when extending termination analysis to imperative programs, cf. the appendix.

4 Termination Analysis for Partial Functions

Up to now we tried to prove that an algorithm terminates *totally*, i.e. for *each input*. In the following, we also regard procedures which terminate for *some* inputs only. For example, consider the data type list with the constructors empty and $\bullet$: nat $\times$ list $\to$ list where $x \bullet y$ represents the insertion of the number x in front of the list y. Then nextindex(x, y, z) returns the smallest index $i \geq x$ such that z is the i-th element of the list y (where the first element has index 0).

```
function nth : nat × list → nat          function eq : nat × nat → bool
  nth(u, empty)  = 0                        eq(0, 0)        = true
  nth(0, v•w))   = v                        eq(0, s(v))     = false
  nth(s(u), v•w) = nth(u, w)                eq(s(u), 0)     = false
                                            eq(s(u), s(v))  = eq(u, v)
function nextindex : nat × list × nat → nat
  nextindex(x, y, z) = if( eq(nth(x, y), z),  x,  nextindex(s(x), y, z) )
```

Let '$u \bullet v \bullet w$' abbreviate '$u \bullet (v \bullet w)$'. Hence, nextindex$(2, 5 \bullet 6 \bullet 3 \bullet 5 \bullet 7 \bullet 5 \bullet \text{empty}, 5) = 3$. While termination of nth and eq can easily be proved, nextindex(x, y, z) only terminates iff z occurs in y at a position whose index is greater than or equal to x or if $z = 0$ (as nth$(x, y) = 0$ whenever x is not an index of y). Thus, evaluation of nextindex$(2, 5 \bullet 6 \bullet 3 \bullet 5 \bullet 7 \bullet 5 \bullet \text{empty}, 6)$ does not halt.

4.1 Termination Predicates

To represent subsets of inputs where procedures like nextindex terminate, in [BG96] we introduced termination predicates. An n-ary boolean function θ_f is a *termination predicate* for an n-ary function f iff θ_f is total and if $\theta_f(t_1,..,t_n) = \mathsf{true}$ implies that evaluation of $f(t_1,..,t_n)$ halts. Our aim is to synthesize termination predicates which return true as often as possible, but of course in general this goal cannot be reached as the domains of functions are undecidable.

In [BG96, GWB98], rules for the synthesis of termination predicates are developed. Given a procedure for f and a well-founded ordering $\prec$ these rules generate the defining equations of a procedure for θ_f such that $\theta_f(t_1,..,t_n)$ returns true *iff* for $f(t_1,..,t_n)$

(i) the sequence of arguments of (recursive) f-calls decreases w.r.t. $\prec$ and

(ii) $\theta_g(r_1,..,r_n)$ holds for each resulting auxiliary function call $g(r_1,..,r_n)$.

For example, given $\prec$ the following procedure is synthesized for $\theta_{\mathsf{nextindex}}$.

function $\theta_{\mathsf{nextindex}}$: nat × list × nat → bool
$\theta_{\mathsf{nextindex}}(x,y,z) = \mathsf{if}(\,\mathsf{eq}(\mathsf{nth}(x,y),z),\ \mathsf{true},\ \mathsf{if}(\,\langle \mathsf{s}(x),y,z\rangle \prec \langle x,y,z\rangle,$
$\qquad \theta_{\mathsf{nextindex}}(\mathsf{s}(x),y,z),$
$\qquad \mathsf{false}\,)\,)$

The procedure $\theta_{\mathsf{nextindex}}$ satisfies (i), since under the condition $\neg\mathsf{eq}(\mathsf{nth}(x,y),z)$ of the only recursive call in nextindex, $\theta_{\mathsf{nextindex}}$ returns true iff the arguments of this recursive call decrease (i.e. $\langle \mathsf{s}(x),y,z\rangle \prec \langle x,y,z\rangle$) and if the arguments of the subsequent recursive nextindex-calls decrease w.r.t. $\prec$, too (i.e. $\theta_{\mathsf{nextindex}}(\mathsf{s}(x),y,z)$). Furthermore, (ii) is satisfied since the only auxiliary functions, nth and eq, are total. As the constructors also denote total functions, we may neglect their termination predicates. Note that $\theta_{\mathsf{nextindex}}$ terminates totally by construction since it is called recursively only if the arguments decrease w.r.t. $\prec$.

4.2 Inductive Evaluation for Partial Functions

The synthesis of termination predicates described in [BG96] requires the user to provide a well-founded ordering $\prec$. Our aim is to get independent from this input by generating suitable polynomial orderings for termination predicates automatically. Here, our goal is to synthesize an ordering which satisfies the termination hypotheses 'as often as possible'[6]. In order to find a suitable choice for the ordering $\prec$ in the termination predicate algorithm $\theta_{\mathsf{nextindex}}$, consider the termination hypothesis of nextindex,

$$\neg\mathsf{eq}(\mathsf{nth}(x,y),z) \rightarrow \langle \mathsf{s}(x),y,z\rangle \prec \langle x,y,z\rangle. \tag{10}$$

The formula (10) cannot be transformed into constraints that are satisfied by a polynomial ordering, since then total termination of nextindex would be falsely proved. However, we can use inductive evaluation to generate a polynomial ordering that satisfies (10) for a *maximal* number of instances. In this way, we finally obtain a termination predicate for nextindex that is true as often as possible.

In general, the set H of termination hypotheses is transformed into a (possibly empty) set C of inequalities that are satisfied by a polynomial ordering. For that purpose we also use *unsound* transformation rules. However, based on the faulty transformation, for each termination hypothesis $\psi \rightarrow r^* \prec t^*$ containing the variables x^*, a *soundness predicate* λ is generated such that every polynomial ordering satisfying C also satisfies the restricted termination hypothesis $\lambda(x^*) \wedge \psi \rightarrow r^* \prec t^*$. So the soundness predicate λ indicates for which data objects the transformation of $\psi \rightarrow r^* \prec t^*$ into the constraints C is correct[7]. Hence, if we choose $\prec$ to be a polynomial ordering satisfying the obtained constraints C, then we can modify the termination predicate algorithm and replace the inequality $r^* \prec t^*$ by the corresponding soundness predicate $\lambda(x^*)$.

For instance, to find a suitable polynomial ordering for $\theta_{\mathsf{nextindex}}$ we transform the termination hypothesis (10). To exploit the premise $\neg\mathsf{eq}(\mathsf{nth}(x,y),z)$ our heuristic suggests inductive evaluation w.r.t. nth. According to the definition of nth we have to consider two base cases and one step case. For none of these cases the premise is unsatisfiable. In the third case the induction hypothesis may be applied as the formula $\neg\mathsf{eq}(\mathsf{nth}(\mathsf{s}(u),v\centerdot w),z) \rightarrow \neg\mathsf{eq}(\mathsf{nth}(u,w),z)$ can be proved by symbolic

[6] For algorithms with auxiliary functions in the recursive *arguments* (instead of the *conditions*), the techniques developed for total termination [Gie95c] can be adapted to partial functions [Bra97], cf. [GWB98].

[7] This is similar to the approach of [Pro96] where a *proof predicate* is generated from an unsound induction proof in order to extend faulty conjectures to valid ones.

evaluation. Thus inductive evaluation and subsequent premise elimination transform (10) into the following inequalities.

$$\langle \mathsf{s}(u), \mathsf{empty}, z\rangle \prec \langle u, \mathsf{empty}, z\rangle \tag{11}$$

$$\langle \mathsf{s}(0), v\bullet w, z\rangle \prec \langle 0, v\bullet w, z\rangle \tag{12}$$

$$\langle \mathsf{s}(\mathsf{s}(u)), v\bullet w, z\rangle + \langle u, w, z\rangle \preceq \langle \mathsf{s}(u), v\bullet w, z\rangle + \langle \mathsf{s}(u), w, z\rangle \tag{13}$$

Of course, (11)-(13) are not satisfied by any polynomial ordering, since nextindex is not totally terminating. Hence, we do not demand that *all* inequalities but only that *some* inequalities are satisfied by a polynomial ordering. For that purpose we select a subset of (11)-(13) and check if it is satisfied by a polynomial ordering. For instance, if (11) is rejected, then (12) and (13) are satisfied if we associate empty and 0 with 0, $\mathsf{s}(u)$ with $u+1$, $v\bullet w$ with $w+1$, and $\langle x, y, z\rangle$ with $(y-x)^2$.

In general, our aim is to find a maximal subset of the inequalities that is satisfied by a polynomial ordering and to reject as few inequalities as possible. As the number of hypotheses is always finite (and small), exhaustive search could be used to determine the inequalities that should be rejected. However, efficiency can be improved if 'probably polynomially satisfiable' inequalities are selected by the heuristics developed in [Gie95b] which have proved successful in practice.

In our example, we associate the following soundness predicate λ with the faulty transformation of (10) into (12) and (13), where $\lambda(x, y, z)$ is true for all those instantiations of x, y, and z where this transformation was correct.

$$\begin{aligned}
&\textbf{function } \lambda : \mathsf{nat} \times \mathsf{list} \times \mathsf{nat} \to \mathsf{bool}\\
&\lambda(u, \mathsf{empty}, z) = \mathsf{false}\\
&\lambda(0, v\bullet w, z) \quad = \mathsf{true}\\
&\lambda(\mathsf{s}(u), v\bullet w, z) = \lambda(u, w, z)
\end{aligned}$$

The case analysis of λ is given by the case analysis of the algorithm nth which has been used for inductive evaluation of (10). The results of λ are created depending on the transformation steps performed during the inductive evaluation. Since for $y = \mathsf{empty}$ inequality (11) has been *rejected*, λ returns false for that case. Analogously, as (12) was kept as a constraint, this results in the value true for $x = 0$ and $y = v\bullet w$. For inputs of the form $\mathsf{s}(u), v\bullet w, z$, the soundness of the transformation depends on the soundness of the transformation for u, w, z, because inequality (13) of the third case has been created by applying the *induction hypothesis*. Hence, in this case the result $\lambda(u, w, z)$ is generated. The procedure λ terminates totally by construction as it is called recursively under the same condition as nth whose total termination has already been verified.

The algorithm λ returns true iff the first argument is less than the length of the second argument. If the inequality $\langle \mathsf{s}(x), y, z\rangle \prec \langle x, y, z\rangle$ in the termination predicate $\theta_{\mathsf{nextindex}}$ is replaced by $\lambda(x, y, z)$, then $\theta_{\mathsf{nextindex}}(x, y, z)$ is true iff z occurs in y at a position $i \geq x$ or if $z = 0$, i.e. we have indeed generated a termination predicate that returns true as often as possible.

To formalize the generation of soundness predicates λ_φ for termination hypotheses φ, we modify the calculus of Sect. 3. The resulting calculus operates on triples H; C; E where the third component E contains the defining equations of the newly synthesized soundness predicates. The correctness of the calculus guarantees that if H; C; E can be transformed into H', C', E', then every polynomial ordering satisfying C' and $\lambda_\varphi(x^*) \to \varphi$ for all $\varphi \in H'$ also satisfies C and $\lambda_\varphi(x^*) \to \varphi$ for all $\varphi \in H$. Here, the semantics of λ_φ is given by E'[8].

To use our calculus for termination proofs, we again initialize H with the termination hypotheses and let C and E be empty. Then the rules of the calculus are applied repeatedly until we have obtained a triple of the form $\emptyset$; C'; E'. Now the defining equations E' of the generated soundness predicates are added to our specification. Then the correctness of the calculus guarantees that every polynomial ordering satisfying the constraints C' also satisfies the original termination hypotheses $\varphi \in H$ for those inputs where the corresponding soundness predicates λ_φ return true. Hence, if there exists a polynomial ordering satisfying C', then in the definitions of termination predicates

[8] For those soundness predicates λ_φ where E' does not yet contain any defining equations, the semantics can be chosen arbitrarily. As our calculus guarantees $E \subseteq E'$, every semantics chosen according to E' also satisfies the defining equations in E.

each inequality $r^* \prec t^*$ may be replaced by the soundness predicate for the termination hypothesis $\psi \to r^* \prec t^*$.

In the following two rules, let $x_1, .., x_l$ (x^* for short) be the variables in $\psi \to \omega$ of types $\delta_1, .., \delta_l$ and let $\lambda_{\psi\to\omega}$ be a new boolean function symbol with the argument types $\delta_1 \times .. \times \delta_l$. As in Sect. 3 we also allow an application of the next two rules if the condition ψ is missing.

Premise Elimination $$\frac{H \cup \{\psi \to \omega\}\,;\quad C \quad;\, E}{H\;;\; C \cup \{\omega\}\,;\; E \cup \{\lambda_{\psi\to\omega}(x^*) = \mathsf{true}\}}$$
if ω does not contain any defined symbols.

Rejection $$\frac{H \cup \{\psi \to \omega\}\,;\; C\,;\; E}{H\;;\; C\;;\; E \cup \{\lambda_{\psi\to\omega}(x^*) = \mathsf{false}\}}$$

In the third rule, let x^* be the variables of $\psi \to r^* \prec t^*$ and to ease readability we write λ instead of $\lambda_{\psi\to r^*\prec t^*}$. Moreover, let the variables $y_1, .., y_n$ of the data types $\tau_1, .., \tau_n$ be contained in x^* and let $g : \tau_1 \times .. \times \tau_n \to \tau$ be a terminating function. Again $p[s^*]$ is an abbreviation for $p[y^*/s^*]$ and moreover, $\lambda[s^*]$ is used as an abbreviation for $\lambda(x^*[y^*/s^*])$, i.e. each y_i is instantiated with s_i and the remaining variables of x^* remain unchanged.

Inductive Evaluation $$\frac{H \cup \{\psi \to r^* \prec t^*\}\,;\;\; C\,;\;\; E}{H \cup \{\varphi_1, .., \varphi_k\}\,;\;\; C\,;\;\; E \cup \{e_1, .., e_k\}}$$
If $g(s^*) = q$ is the i-th defining equation of g, then φ_i and e_i are defined as

$$\left.\begin{array}{lll} \text{-}\,\varphi_i & := & \mathsf{true} \\ \;e_i & := & \lambda[s^*] = \mathsf{true} \end{array}\right\} \text{if } \neg\psi[s^*] \text{ can be verified,}$$

$$\left.\begin{array}{lll} \text{-}\,\varphi_i & := & \psi[s^*] \to r^*[s^*] + t^*[q^*] \preceq t^*[s^*] + r^*[q^*] \\ \;e_i & := & \lambda[s^*] = \lambda[q^*] \wedge \lambda_{\varphi_i}(z^*) \end{array}\right\} \begin{array}{l}\text{otherwise, if } q \text{ contains } g(q^*) \text{ and} \\ \psi[s^*] \to \psi[q^*] \text{ can be verified,}\end{array}$$

$$\left.\begin{array}{lll} \text{-}\,\varphi_i & := & \psi[s^*] \to r^*[s^*] \prec t^*[s^*], \\ \;e_i & := & \lambda[s^*] = \lambda_{\varphi_i}(z^*) \end{array}\right\} \text{otherwise.}$$

Here, z^* are the variables occurring in φ_i.

Using this calculus, for the termination hypothesis (10) a soundness predicate $\lambda_{(10)}$ is generated with the defining equations $\lambda_{(10)}(u, \mathsf{empty}, z) = \lambda_{(11)}(u, z)$, $\lambda_{(10)}(0, v{\bullet}w, z) = \lambda_{(12)}(v, w, z)$, and $\lambda_{(10)}(\mathsf{s}(u), v{\bullet}w, z) = \lambda_{(10)}(u, w, z) \wedge \lambda_{(13)}(u, v, w, z)$, where $\lambda_{(11)}, \lambda_{(12)}, \lambda_{(13)}$ have the defining equations $\lambda_{(11)}(u, z) = \mathsf{false}, \lambda_{(12)}(v, w, z) = \mathsf{true}, \lambda_{(13)}(u, v, w, z) = \mathsf{true}$. Hence, by symbolic evaluation one obtains the algorithm λ given at the beginning of the section.

This extension of inductive evaluation for termination analysis of partial functions generalizes our first approach, i.e. whenever total termination of f can be verified by the technique of Sect. 3, the technique of the present section can generate a termination predicate θ_f that returns true for each input.

The handling of partial functions is also necessary for termination analysis of *imperative* programs, because when translating imperative programs into functional ones, *while*-loops are often transformed into *partial* functions, as termination of *while*-loops often depends on their contexts, cf. [GWB98].

5 Refinements

In this section we present extensions of our approach which increase its power considerably. As an example consider the following algorithms.

```
function max : list → nat          function add_if_member : nat × list × list → list
 max(empty)     = 0                 add_if_member(x, empty, z) = z
 max(u•empty)   = u                 add_if_member(x, u•y, z)   = if( eq(x, u),
 max(u•v•w)     = if( le(u, v),                                      x•z,
                      max(v•w),                                      add_if_member(x, y, z) )
                      max(u•w) )

function sort : nat × list → list
 sort(x, y) = if( eq(x, max(y)), x•empty, add_if_member(x, y, sort(s(x), y)) )
```

Total termination of both max and add_if_member can easily be proved (where add_if_member(x, y, z) returns x•z if x occurs in y and z otherwise[9]). The function sort(x, y) sorts all elements of y which are greater than or equal to x where multiple occurrences of elements are removed. Hence, sort$(0, y)$ sorts the entire list y. This procedure terminates iff x is less than or equal to the maximal element of y (where the maximum of empty is 0). Consider sort's termination hypothesis,

$$\neg \mathsf{eq}(x, \mathsf{max}(y)) \rightarrow \langle \mathsf{s}(x), y\rangle \prec \langle x, y\rangle. \tag{14}$$

To construct a soundness predicate for (14) according to our heuristic we have to use inductive evaluation w.r.t. the algorithm max. However, for inductive evaluation w.r.t. functions like max which are defined using the conditional if, we now have to refine the Inductive Evaluation rule.

In the following we assume that all terms are '*normalized*' w.r.t. if, i.e. the only functions on positions above an if are also conditionals. As usual, positions in terms are denoted by sequences of naturals where the empty sequence ϵ denotes the root position. Then r is the *result* of the term t under the *condition* ω at the *position* π, iff

- t contains no if's, $\omega = \mathsf{true}$, $\quad \pi = \epsilon$, $\quad r = t$

or - $t = \mathsf{if}(\psi, t_1, t_2)$, $\quad \omega = \psi \wedge \omega'$, $\pi = 2\pi'$, r is the result of t_1 under ω' at π'

or - $t = \mathsf{if}(\psi, t_1, t_2)$, $\quad \omega = \neg\psi \wedge \omega'$, $\pi = 3\pi'$, r is the result of t_2 under ω' at π'.

Let the right-hand side of g's i-th defining equation contain m_i results. Then by performing a case analysis w.r.t. their conditions, the Inductive Evaluation rule is extended to functions g defined by conditionals. Here, $q[\pi \leftarrow t]$ denotes the term obtained by replacing the subterm of q at position π by t.

Inductive Evaluation
$$\frac{H \cup \{\psi \rightarrow r^* \prec t^*\}\ ;\ C\ ;\ E}{H \cup \{\varphi_{11}, .., \varphi_{1m_1}, .., \varphi_{k1}, .., \varphi_{km_k}\}\ ;\ C\ ;\ E \cup \{e_1, .., e_k\}}$$

If $g(s^*) = q$ is the i-th defining equation of g where q has the results $q_1, .., q_m$ under the conditions $\omega_1, .., \omega_m$ at the positions $\pi_1, .., \pi_m$, then we define $e_i \ :=\ \lambda[s^*] = q[\pi_1 \leftarrow \xi_1] \ldots [\pi_m \leftarrow \xi_m]$, where

- $\varphi_{ij} := \mathsf{true}$, $\ \xi_j := \mathsf{true}$ $\quad$ if $\neg(\psi[s^*] \wedge \omega_j[s^*])$ can be verified,
- $\varphi_{ij} := \psi[s^*] \wedge \omega_j[s^*] \rightarrow r^*[s^*] + t^*[q^*] \preceq t^*[s^*] + r^*[q^*]$, $\ \xi_j := \lambda[q^*] \wedge \lambda_{\varphi_{ij}}(z^*)$ $\quad$ otherwise, if q_j contains $g(q^*)$ where $\psi[s^*] \wedge \omega_j[s^*] \rightarrow \psi[q^*]$ can be verified,
- $\varphi_{ij} := \psi[s^*] \wedge \omega_j[s^*] \rightarrow r^*[s^*] \prec t^*[s^*]$, $\ \xi_j := \lambda_{\varphi_{ij}}(z^*)$ $\quad$ otherwise.

Here, z^* are the variables occurring in φ_{ij}.

With this rule we can perform inductive evaluation of (14) w.r.t. max using y as induction variable. In each resulting case the premise is satisfiable and in both step cases (resulting from

[9] In the algorithm sort, we use add_if_member$(x, y, \mathsf{sort}(\mathsf{s}(x), y))$ instead of if(member(x, y), x•sort$(\mathsf{s}(x), y)$, sort$(\mathsf{s}(x), y)$) to ease the readability of our presentation.

the *two* results of max' last defining equation) the induction hypothesis can be applied. Hence we obtain the following new hypotheses and the soundness predicate $\lambda_{(14)}(x,y)$ (omitting true in conjunctions).

$$\neg\mathsf{eq}(x,\mathsf{max}(\mathsf{empty})) \to \langle \mathsf{s}(x),\mathsf{empty}\rangle \prec \langle x,\mathsf{empty}\rangle \tag{15}$$

$$\neg\mathsf{eq}(x,\mathsf{max}(u\bullet\mathsf{empty})) \to \langle \mathsf{s}(x),u\bullet\mathsf{empty}\rangle \prec \langle x,u\bullet\mathsf{empty}\rangle \tag{16}$$

$$\neg\mathsf{eq}(x,\mathsf{max}(u\bullet v\bullet w)) \wedge \mathsf{le}(u,v) \to \langle \mathsf{s}(x),u\bullet v\bullet w\rangle + \langle x,v\bullet w\rangle \preceq \langle x,u\bullet v\bullet w\rangle + \langle \mathsf{s}(x),v\bullet w\rangle \tag{17}$$

$$\neg\mathsf{eq}(x,\mathsf{max}(u\bullet v\bullet w)) \wedge \neg\mathsf{le}(u,v) \to \langle \mathsf{s}(x),u\bullet v\bullet w\rangle + \langle x,u\bullet w\rangle \preceq \langle x,u\bullet v\bullet w\rangle + \langle \mathsf{s}(x),u\bullet w\rangle \tag{18}$$

function $\lambda_{(14)}$: nat × list → bool
$\lambda_{(14)}(x,\mathsf{empty}) \;=\; \lambda_{(15)}(x)$
$\lambda_{(14)}(x,u\bullet\mathsf{empty}) = \lambda_{(16)}(x,u)$
$\lambda_{(14)}(x,u\bullet v\bullet w) \;=\; \mathsf{if}(\,\mathsf{le}(u,v), \lambda_{(14)}(x,v\bullet w) \wedge \lambda_{(17)}(x,u,v,w),$
$\qquad\qquad \lambda_{(14)}(x,u\bullet w) \wedge \lambda_{(18)}(x,u,v,w)\,)$

Our heuristic suggests no further inductive evaluation for (15) and (16), since no term $g(y_1,..,y_n)$ with pairwise different y_i occurs in their premises. But then the inequalities in (15) and (16) have to be rejected, since they are not satisfied by any polynomial ordering. Thus, both $\lambda_{(15)}$ and $\lambda_{(16)}$ would always be false and hence, the soundness predicate $\lambda_{(14)}$ for sort's termination hypothesis would also return false for each input. Thus, we would obtain an unsatisfiable soundness predicate although evaluation halts for some recursive calls of sort.

To construct a better soundness predicate, we should again perform inductive evaluation on the obtained hypothesis (16). For that purpose the occurring max-term is *symbolically evaluated.* If we replace the term max(u•empty) by its *symbolic value* u then instead of (16) we get

$$\neg\mathsf{eq}(x,u) \to \langle \mathsf{s}(x),u\bullet\mathsf{empty}\rangle \prec \langle x,u\bullet\mathsf{empty}\rangle. \tag{19}$$

Hence, we extend our calculus by an additional *Symbolic Evaluation rule* which allows to replace a premise $\psi[\pi \leftarrow g(t^*)]$ by $\psi[\pi \leftarrow r]$ whenever $g(t^*)$ can be evaluated to r, where C and E do not change.

Now for (19) our heuristic suggests inductive evaluation with respect to eq. As the negated premise of the resulting first case can easily be verified and the induction hypothesis can be applied in the last case, (19) is replaced by

$$\neg\mathsf{eq}(0,\mathsf{s}(v)) \to \langle \mathsf{s}(0),\mathsf{s}(v)\bullet\mathsf{empty}\rangle \prec \langle 0,\mathsf{s}(v)\bullet\mathsf{empty}\rangle \tag{20}$$

$$\neg\mathsf{eq}(\mathsf{s}(u),0) \to \langle \mathsf{s}(\mathsf{s}(u)),0\bullet\mathsf{empty}\rangle \prec \langle \mathsf{s}(u),0\bullet\mathsf{empty}\rangle \tag{21}$$

$$\neg\mathsf{eq}(\mathsf{s}(u),\mathsf{s}(v)) \to \langle \mathsf{s}(\mathsf{s}(u)),\mathsf{s}(v)\bullet\mathsf{empty}\rangle + \langle u,v\bullet\mathsf{empty}\rangle \preceq \langle \mathsf{s}(u),\mathsf{s}(v)\bullet\mathsf{empty}\rangle + \langle \mathsf{s}(u),v\bullet\mathsf{empty}\rangle. \tag{22}$$

To ensure that inductive evaluation is only applied a finite number of times, we use the heuristic never to perform inductive evaluation w.r.t. the same algorithm twice. So even if the eq-terms in (20)-(22) are evaluated symbolically, we do not apply another inductive evaluation w.r.t. eq. On the other hand, for the hypotheses (17) and (18) we could perform another inductive evaluation w.r.t. le. For that purpose we also introduce a non-strict version of the Inductive Evaluation rule where all occurrences of $\prec$ are replaced by $\preceq$. If we finally reject the hypotheses (15) and (21) and eliminate the premises of all remaining inequalities, then the resulting constraints are satisfied by the polynomial ordering where empty and 0 are mapped to 0 and s(x), x•y, and $\langle x,y\rangle$ are mapped to $x+1$, $x+y$, and $(y-x)^2$, respectively. In fact, in this example the last inductive evaluation of (17) and (18) w.r.t. le turns out not to be necessary, because this polynomial ordering already satisfies the inequalities in (17), (18), (20), and (22). Hence, for the hypothesis (15) we generate the soundness predicate $\lambda_{(15)}(x) = \mathsf{false}$ and for (16) we obtain the soundness predicate

function $\lambda_{(16)}$: $\mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$$\begin{array}{llll}\lambda_{(16)}(\mathsf{0},\mathsf{0}) & = \mathsf{true} & \lambda_{(16)}(\mathsf{s}(u),\mathsf{0}) & = \mathsf{false}\\ \lambda_{(16)}(\mathsf{0},\mathsf{s}(v)) & = \mathsf{true} & \lambda_{(16)}(\mathsf{s}(u),\mathsf{s}(v)) & = \lambda_{(16)}(u,v).\end{array}$$

Thus, $\lambda_{(16)}$ computes the less than or equal relation on naturals, and hence, the soundness predicate $\lambda_{(14)}(x,y)$ for sort'*s* termination hypothesis is true iff y is non-empty and if x is less than or equal to the maximal element of y. Using this soundness predicate we finally obtain the termination predicate procedure

function θ_{sort} : $\mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$

$$\theta_{\mathsf{sort}}(x,y) = \mathsf{if}(\,\mathsf{eq}(x,\mathsf{max}(y)),\ \mathsf{true},\ \mathsf{if}(\lambda_{(14)}(x,y),\ \theta_{\mathsf{sort}}(\mathsf{s}(x),y),\ \mathsf{false})\,).$$

The procedure θ_{sort} defines the exact domain of sort, i.e. it returns true iff x is less than or equal to the maximal element of y. Hence, in this way a predicate describing the domain of sort can be generated automatically.

6 Conclusion

We have illustrated that termination of many interesting algorithms cannot be verified if the premises of the termination hypotheses are neglected. Therefore, in this paper we presented the *inductive evaluation* method which analyzes auxiliary functions occurring in the conditions of recursive calls. Our calculus transforms termination hypotheses into inequalities such that existing automated methods can be used to check whether they are satisfied by a polynomial ordering. In this way, total termination of algorithms can be proved automatically.

Subsequently,we have generalized our approach for analyzing partially terminating procedures. For that purpose our calculus is extended in order to synthesize *soundness predicates* which are used for the construction of termination predicates describing the domain of the function under consideration.

We combined our method to handle auxiliary functions in the *conditions* with techniques to deal with defined functions in the *arguments* of recursive calls [Gie95b, Gie95c, Gie97, GWB98] and implemented it within the induction theorem prover INKA [HS96]. In this way we obtained an extremely powerful approach for automated termination analysis which performed successfully on a large collection of benchmarks (including all 82 algorithms from [BM79], all 60 examples from[Wal94b], and all 92 examples in [Gie95b] and [BG96]).

See the appendix for a collection of 36 algorithms whose termination behaviour could not be analyzed with any other automatic method up to now, but where inductive evaluation enables termination analysis without user interaction. For all these examples, termination predicates describing the *exact* domains of the functions could be synthesized. We also applied our approach to *imperative programs* by translating them into equivalent functional programs. In this way, in 33 of 45 examples from [Gri81] the exact domain could be determined automatically.

Examples

This appendix contains 37 functional and 21 imperative programs to illustrate the power of our method. For functional procedures marked with * total termination is provable without inductive evaluation; they are required as auxiliary procedures for the other examples. Hence their termination can also be proved with existing techniques for proving total termination. For all other procedures in this appendix an automated termination analysis is not possible with previous methods. Note that in all following examples the termination predicates generated by our method are the weakest possible ones, i.e. they return true *iff* the procedure under consideration terminates. For an overview, the procedures and their generated termination predicates are listed in Table 1.

The data types used in the examples are nat (with the constructors 0 and s), list (with the constructors empty and add), sexpr (with the constructors nil : sexpr and cons : sexpr $\times$ sexpr $\to$ sexpr), and tree for binary trees (with the constructors lf : tree ('leaf') and nd : tree$\times$nat$\times$tree $\to$ tree ('node'), where nd(a,b,c) represents the tree whose root is labelled with the natural number b and which has the direct subtrees a and c). Moreover, we use a data type llist for lists of lists with the constructors lempty $:\longrightarrow$ llist and ladd(v, w) : list $\times$ llist $\longrightarrow$ llist.

Procedure	**Termination Predicate**
minus(x, y)	true
nthtail(x, y)	true
tree_minimum(n, t)	true
countsort(n, l)	true
round_3(x)	true
leaf_difference(x, y)	true
bin_log(x, y)	true
mod(x, y)	true
gcd1(x, y)	true
gcd2(x, y)	true
next_non_element(x, y)	true
minus3(x, y)	$x \geq y$
nextindex(i, b, z)	$z = 0 \vee \exists j \,.\, j \geq i \wedge z = b[j]$
sort(x, y)	$x \leq \mathsf{max}(y)$
half_diff(x, y)	$x \geq y \wedge \mathsf{even}(x - y)$
fibonacci(n, i, a, b, h)	true
list_max(n, i, k, b)	true
search(i, b, z)	$z \in b \vee z = 0$
insert(z, k, p, b)	$k < \mathsf{len}(b)$
sum(i, x, n, b)	true
llist_search1(b, n, m, x, i, j)	$m = 0 \vee n \neq 0 \vee x = b[0, 0]$
4_tuplesort(w, x, y, z, h)	true
sqrt1(n, a)	true
plateau(n, i, p, b)	$n > 0$
common_elem(f, g, h, i, j, k)	true
swap_equals(b, i, j, k, n, h)	true
horner(a, n, i, x, y, z)	$n \neq 0$
perm_to_code(x, j, k, n)	true
code_to_perm(x, j, k, n)	true
list_min(n, i, x, b)	$n \neq 0$
llist_search2(b, n, x, i, j)	$x = 0 \vee \exists i, j \,.\, 0 \leq i < \mathsf{len}(b) \wedge 0 \leq j < n \wedge b[i, j] = x$
partition(b, m, n, x, q, r, h)	true
llist_search3(b, n, x, i, j)	$(b = \mathsf{empty} \wedge x = 0) \vee$ $(\exists i, j \,.\, 0 \leq i < \mathsf{len}(b) \wedge 0 \leq j < n \wedge b[i, j] = x \wedge$ $\exists k_1, .., k_r, l_1, .., l_r \,.\, k_1 = 0 \wedge l_1 = n - 1 \wedge k_r = i \wedge l_r = j \wedge$ $\forall p \in \{1, .., r - 1\} \,.\, (x < b[k_p, l_p] \wedge k_{p+1} = k_p \wedge l_{p+1} = l_p - 1) \vee$ $(x > b[k_p, l_p] \wedge k_{p+1} = k_p + 1 \wedge l_{p+1} = l_p))$
sqr_sums(r, a, b, i, x, y)	true
sqrt2(n, p, q, r)	true
singles(h, k, c, n, m, f, g)	true

Table 1. Functional and imperative procedures whose termination is analyzed with inductive evaluation

A Functional Procedures

For each functional procedure we describe its semantics. Then we list the termination hypotheses which are subsequently transformed using Inductive Evaluation, Symbolic Evaluation, Premise Elimination, and Rejection.

We mention a polynomial ordering which satisfies the resulting inequalities. Then we give the soundness predicates which are synthesized according to the previous transformation. Finally we show the resulting termination predicate procedure. For each generated procedure we describe its semantics. In the following, we omit termination predicates for constructors because these predicates always return true.

1 le^*

The functional procedure le computes the less than or equal relation for naturals.

function $\mathsf{le} : \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
$\quad\mathsf{le}(0, v) = \mathsf{true}$
$\quad\mathsf{le}(\mathsf{s}(u), 0) = \mathsf{false}$
$\quad\mathsf{le}(\mathsf{s}(u), \mathsf{s}(v)) = \mathsf{le}(u, v)$

Termination Hypothesis

$$\langle u, v\rangle \prec \langle \mathsf{s}(u), \mathsf{s}(v)\rangle \tag{23}$$

Transformation
No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \langle x, y\rangle \mapsto x$

Soundness Predicate
function $\lambda_{(23)} : \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
$\quad\lambda_{(23)}(u, v) = \mathsf{true}$

Termination Predicate
function $\theta_{\mathsf{le}} : \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
$\quad\theta_{\mathsf{le}}(0, v) = \mathsf{true}$
$\quad\theta_{\mathsf{le}}(\mathsf{s}(u), 0) = \mathsf{true}$
$\quad\theta_{\mathsf{le}}(\mathsf{s}(u), \mathsf{s}(v)) = \theta_{\mathsf{le}}(u, v)$

In [BG96] we introduced a technique for automated simplification of termination (or soundness) predicate algorithms. This technique would immediately transform the above algorithm into the following one.

function $\theta_{\mathsf{le}} : \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
$\quad\theta_{\mathsf{le}}(u, v) = \mathsf{true}$

In most of the following examples, we will only mention the simplified termination and soundness predicate algorithms.

2 lt^*

The functional procedure lt computes the less than relation for naturals.

function $\mathsf{lt} : \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
$\quad\mathsf{lt}(u, 0) = \mathsf{false}$
$\quad\mathsf{lt}(0, \mathsf{s}(v)) = \mathsf{true}$
$\quad\mathsf{lt}(\mathsf{s}(u), \mathsf{s}(v)) = \mathsf{lt}(u, v)$

Termination analysis analogously to le yields $\theta_{\mathsf{lt}}(x, y) = \mathsf{true}$.

3 minus

The functional procedure minus computes the subtraction for naturals.

function minus : nat × nat → nat
  $\mathsf{minus}(x, y) = \mathsf{if}(\mathsf{le}(x, y),\ 0,\ \mathsf{s}(\mathsf{minus}(x, \mathsf{s}(y))))$

Termination Hypothesis

$$\neg\mathsf{le}(x, y) \rightarrow \langle x, \mathsf{s}(y)\rangle \prec \langle x, y\rangle \tag{24}$$

Transformation

1. Inductive Evaluation w.r.t. $\mathsf{le}(x, y)$, Symbolic Evaluation:

$$\mathsf{true} \rightarrow \langle \mathsf{s}(u), \mathsf{s}(0)\rangle \prec \langle \mathsf{s}(u), 0\rangle \tag{25}$$
$$\neg\mathsf{le}(u, v) \rightarrow \langle \mathsf{s}(u), \mathsf{s}(\mathsf{s}(v))\rangle + \langle u, v\rangle \preceq \langle \mathsf{s}(u), \mathsf{s}(v)\rangle + \langle u, \mathsf{s}(v)\rangle \tag{26}$$

2. Premise Elimination in (25) and (26)

Polynomial Ordering
  $0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \langle x, y\rangle \mapsto (x - y)^2$

Soundness Predicate
  function $\lambda_{(24)}$: nat × nat → bool
    $\lambda_{(24)}(x, y) = \mathsf{true}$

Termination Predicate
  function θ_{minus} : nat × nat → bool
    $\theta_{\mathsf{minus}}(x, y) = \mathsf{true}$

4 longer_than*

The functional procedure longer_than returns true iff the length of a list is greater than or equal to a given natural.

function longer_than : list × nat → bool
  $\mathsf{longer\_than}(w, 0) = \mathsf{true}$
  $\mathsf{longer\_than}(\mathsf{empty}, \mathsf{s}(u)) = \mathsf{false}$
  $\mathsf{longer\_than}(\mathsf{add}(v, w), \mathsf{s}(u)) = \mathsf{longer\_than}(w, u)$

Termination Hypothesis

$$\langle w, u\rangle \prec \langle \mathsf{add}(v, w), \mathsf{s}(u)\rangle \tag{27}$$

Transformation
  No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering
  $0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \mathsf{empty} \mapsto 0,\ \mathsf{add}(v, w) \mapsto w + 1,\ \langle x, y\rangle \mapsto x$

Soundness Predicate
  function $\lambda_{(27)}$: list × nat × nat → bool
    $\lambda_{(27)}(w, u, v) = \mathsf{true}$

Termination Predicate
  function $\theta_{\mathsf{longer\_than}}$: list × nat → bool
    $\theta_{\mathsf{longer\_than}}(w, u) = \mathsf{true}$

5 tl*

$\mathsf{tl}(y)$ returns the list y without its first element.

function $\mathsf{tl} : \mathsf{list} \to \mathsf{list}$
$$\begin{aligned}\mathsf{tl}(\mathsf{empty}) &= \mathsf{empty}\\ \mathsf{tl}(\mathsf{add}(x,y)) &= y\end{aligned}$$

As tl has no recursive calls, there is no termination hypothesis for tl and hence, its termination is trivially proved.

6 nthtail

$\mathsf{nthtail}(x,y)$ returns the list containing the last x elements of the list y.

function $\mathsf{nthtail} : \mathsf{nat} \times \mathsf{list} \to \mathsf{nat}$
$$\mathsf{nthtail}(x,y) = \mathsf{if}(\mathsf{longer\_than}(y,x),\ \mathsf{tl}(\mathsf{nthtail}(\mathsf{s}(x),y)),\ y)$$

Termination Hypothesis

$$\neg\mathsf{longer\_than}(y,x) \to \langle \mathsf{s}(x), y\rangle \prec \langle x, y\rangle \tag{28}$$

Transformation

1. Inductive Evaluation w.r.t. $\mathsf{longer\_than}(y,x)$, Symbolic Evaluation:

$$\mathsf{true} \to \langle \mathsf{s}(\mathsf{s}(u)), \mathsf{empty}\rangle \prec \langle \mathsf{s}(u), \mathsf{empty}\rangle \tag{29}$$
$$\neg\mathsf{longer\_than}(w,u) \to \langle \mathsf{s}(\mathsf{s}(u)), \mathsf{add}(v,w)\rangle + \langle u, w\rangle \preceq \langle \mathsf{s}(u), \mathsf{add}(v,w)\rangle + \langle \mathsf{s}(u), w\rangle \tag{30}$$

2. Premise Elimination in (29) and (30)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \mathsf{empty} \mapsto 0,\ \mathsf{add}(v,w) \mapsto w+1,\ \langle x,y\rangle \mapsto (x-y)^2$

Soundness Predicate

function $\lambda_{(28)} : \mathsf{list} \times \mathsf{nat} \to \mathsf{bool}$
$$\lambda_{(28)}(y,x) = \mathsf{true}$$

Termination Predicate

function $\theta_{\mathsf{nthtail}} : \mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$
$$\theta_{\mathsf{nthtail}}(x,y) = \mathsf{true}$$

7 lb* (“lower bound”)

$\mathsf{lb}(n,t)$ returns true iff the binary tree t contains at least one inner node and n is less than each natural occurring in t.

function $\mathsf{lb} : \mathsf{nat} \times \mathsf{tree} \to \mathsf{bool}$
$$\begin{aligned}
\mathsf{lb}(n,\mathsf{lf}) &= \mathsf{false}\\
\mathsf{lb}(n,\mathsf{nd}(\mathsf{lf},m,\mathsf{lf})) &= \mathsf{if}(\mathsf{le}(m,n),\ \mathsf{false},\ \mathsf{true})\\
\mathsf{lb}(n,\mathsf{nd}(\mathsf{lf},m,\mathsf{nd}(a,b,c))) &= \mathsf{if}(\mathsf{le}(m,n),\ \mathsf{false},\ \mathsf{lb}(n,\mathsf{nd}(a,b,c)))\\
\mathsf{lb}(n,\mathsf{nd}(\mathsf{nd}(a,b,c),m,\mathsf{lf})) &= \mathsf{if}(\mathsf{le}(m,n),\ \mathsf{false},\ \mathsf{lb}(n,\mathsf{nd}(a,b,c)))\\
\mathsf{lb}(n,\mathsf{nd}(\mathsf{nd}(a,b,c),m,\mathsf{nd}(d,e,f))) &= \mathsf{if}(\mathsf{le}(m,n),\ \mathsf{false},\ \mathsf{lb}(n,\mathsf{nd}(a,b,c))\wedge\\
&\qquad \mathsf{lb}(n,\mathsf{nd}(d,e,f)))
\end{aligned}$$

Termination Hypotheses

$$\neg\mathsf{le}(m,n) \rightarrow \langle n, \mathsf{nd}(a,b,c)\rangle \prec \langle n, \mathsf{nd}(\mathsf{lf}, m, \mathsf{nd}(a,b,c))\rangle \qquad (31)$$
$$\neg\mathsf{le}(m,n) \rightarrow \langle n, \mathsf{nd}(a,b,c)\rangle \prec \langle n, \mathsf{nd}(\mathsf{nd}(a,b,c), m, \mathsf{lf})\rangle \qquad (32)$$
$$\neg\mathsf{le}(m,n) \rightarrow \langle n, \mathsf{nd}(a,b,c)\rangle \prec \langle n, \mathsf{nd}(\mathsf{nd}(a,b,c), m, \mathsf{nd}(d,e,f))\rangle \qquad (33)$$
$$\neg\mathsf{le}(m,n) \rightarrow \langle n, \mathsf{nd}(d,e,f)\rangle \prec \langle n, \mathsf{nd}(\mathsf{nd}(a,b,c), m, \mathsf{nd}(d,e,f))\rangle \qquad (34)$$

Transformation
No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \mathsf{lf} \mapsto 0,\ \mathsf{nd}(a,b,c) \mapsto 1+a+c,\ \langle x,y\rangle \mapsto y$

Soundness Predicates
function $\lambda_{(31)}$: $\mathsf{nat} \times \mathsf{nat} \times \mathsf{tree} \times \mathsf{nat} \times \mathsf{tree} \rightarrow \mathsf{bool}$
$\lambda_{(31)}(m,n,a,b,c) = \mathsf{true}$
function $\lambda_{(32)}$: $\mathsf{nat} \times \mathsf{nat} \times \mathsf{tree} \times \mathsf{nat} \times \mathsf{tree} \rightarrow \mathsf{bool}$
$\lambda_{(32)}(m,n,a,b,c) = \mathsf{true}$
function $\lambda_{(33)}$: $\mathsf{nat} \times \mathsf{nat} \times \mathsf{tree} \times \mathsf{nat} \times \mathsf{tree} \times \mathsf{tree} \times \mathsf{nat} \times \mathsf{tree} \rightarrow \mathsf{bool}$
$\lambda_{(33)}(m,n,a,b,c,d,e,f) = \mathsf{true}$
function $\lambda_{(34)}$: $\mathsf{nat} \times \mathsf{nat} \times \mathsf{tree} \times \mathsf{nat} \times \mathsf{tree} \times \mathsf{tree} \times \mathsf{nat} \times \mathsf{tree} \rightarrow \mathsf{bool}$
$\lambda_{(34)}(m,n,d,e,f,a,b,c) = \mathsf{true}$

Termination Predicate
function θ_{lb} : $\mathsf{nat} \times \mathsf{tree} \rightarrow \mathsf{bool}$
$\theta_{\mathsf{longer\_than}}(n,a) = \mathsf{true}$

8 tree_minimum

tree_minimum(t) returns the minimal element of a binary tree t iff t contains at least one inner node and the minimum is greater than or equal to n. Otherwise n is returned. So in particular, tree_minimum$(0,t)$ returns the minimum of the tree t.

function tree_minimum : $\mathsf{nat} \times \mathsf{tree} \rightarrow \mathsf{nat}$
tree_minimum(n,t) = if(lb(n,t), tree_minimum(s(n), t), n)

Termination Hypothesis

$$\mathsf{lb}(n,t) \rightarrow \langle \mathsf{s}(n), t\rangle \prec \langle n, t\rangle \qquad (35)$$

Transformation

1. Inductive Evaluation w.r.t. lb(n,t), Symbolic Evaluation:

$$\neg\mathsf{le}(m,n) \rightarrow \langle \mathsf{s}(n), \mathsf{nd}(\mathsf{lf}, m, \mathsf{lf})\rangle \prec \langle n, \mathsf{nd}(\mathsf{lf}, m, \mathsf{lf})\rangle \qquad (36)$$
$$\neg\mathsf{le}(m,n) \rightarrow \langle \mathsf{s}(n), \mathsf{nd}(\mathsf{lf}, m, \mathsf{nd}(a,b,c))\rangle + \langle n, \mathsf{nd}(a,b,c)\rangle \preceq \langle n, \mathsf{nd}(\mathsf{lf}, m, \mathsf{nd}(a,b,c))\rangle + \langle \mathsf{s}(n), \mathsf{nd}(a,b,c)\rangle \qquad (37)$$
$$\neg\mathsf{le}(m,n) \rightarrow \langle \mathsf{s}(n), \mathsf{nd}(\mathsf{nd}(a,b,c), m, \mathsf{lf})\rangle + \langle n, \mathsf{nd}(a,b,c)\rangle \preceq \langle n, \mathsf{nd}(\mathsf{nd}(a,b,c), m, \mathsf{lf})\rangle + \langle \mathsf{s}(n), \mathsf{nd}(a,b,c)\rangle \qquad (38)$$
$$\neg\mathsf{le}(m,n) \rightarrow \langle \mathsf{s}(n), \mathsf{nd}(\mathsf{nd}(a,b,c), m, \mathsf{nd}(d,e,f))\rangle + \langle n, \mathsf{nd}(a,b,c)\rangle \preceq \langle n, \mathsf{nd}(\mathsf{nd}(a,b,c), m, \mathsf{nd}(d,e,f))\rangle + \langle \mathsf{s}(n), \mathsf{nd}(a,b,c)\rangle \qquad (39)$$

Note that the last defining equation of lb has two recursive calls. According to the inductive evaluation rule, we can choose an arbitrary recursive call for the construction of the induction hypotheses. Hence, in formula (39) we used the arguments of the first recursive call $n, \mathsf{nd}(a,b,c)$ to generate the induction hypothesis. But the termination proof would also succeed if we used the arguments of the second recursive call $n, \mathsf{nd}(d,e,f)$ instead.

2. Inductive evaluation of (36) w.r.t. $\mathsf{le}(m,n)$, Symbolic Evaluation:

$$\begin{aligned} \mathsf{true} &\rightarrow \langle \mathsf{s}(0), \mathsf{nd}(\mathsf{lf}, \mathsf{s}(u), \mathsf{lf})\rangle \prec \langle 0, \mathsf{nd}(\mathsf{lf}, \mathsf{s}(u), \mathsf{lf})\rangle && (40)\\ \neg\mathsf{le}(u,v) &\rightarrow \langle \mathsf{s}(\mathsf{s}(v)), \mathsf{nd}(\mathsf{lf}, \mathsf{s}(u), \mathsf{lf})\rangle + \langle v, \mathsf{nd}(\mathsf{lf}, u, \mathsf{lf})\rangle \preceq \\ &\quad \langle \mathsf{s}(v), \mathsf{nd}(\mathsf{lf}, \mathsf{s}(u), \mathsf{lf})\rangle + \langle \mathsf{s}(v), \mathsf{nd}(\mathsf{lf}, u, \mathsf{lf})\rangle && (41)\end{aligned}$$

3. Inductive evaluation of (37) w.r.t. $\mathsf{le}(m,n)$, Symbolic Evaluation:

$$\begin{aligned} \mathsf{true} &\rightarrow \langle \mathsf{s}(0), \mathsf{nd}(\mathsf{lf}, \mathsf{s}(u), \mathsf{nd}(a,b,c))\rangle + \langle 0, \mathsf{nd}(a,b,c)\rangle \preceq \\ &\quad \langle 0, \mathsf{nd}(\mathsf{lf}, \mathsf{s}(u), \mathsf{nd}(a,b,c))\rangle + \langle \mathsf{s}(0), \mathsf{nd}(a,b,c)\rangle && (42)\\ \neg\mathsf{le}(u,v) &\rightarrow \langle \mathsf{s}(\mathsf{s}(v)), \mathsf{nd}(\mathsf{lf}, \mathsf{s}(u), \mathsf{nd}(a,b,c))\rangle + \langle \mathsf{s}(v), \mathsf{nd}(a,b,c)\rangle + \\ &\quad \langle v, \mathsf{nd}(\mathsf{lf}, u, \mathsf{nd}(a,b,c))\rangle + \langle \mathsf{s}(v), \mathsf{nd}(a,b,c)\rangle \preceq \\ &\quad \langle \mathsf{s}(v), \mathsf{nd}(\mathsf{lf}, \mathsf{s}(u), \mathsf{nd}(a,b,c))\rangle + \langle \mathsf{s}(\mathsf{s}(v)), \mathsf{nd}(a,b,c)\rangle + \\ &\quad \langle \mathsf{s}(v), \mathsf{nd}(\mathsf{lf}, u, \mathsf{nd}(a,b,c))\rangle + \langle v, \mathsf{nd}(a,b,c)\rangle && (43)\end{aligned}$$

4. Inductive evaluation of (38) w.r.t. $\mathsf{le}(m,n)$, Symbolic Evaluation:

$$\begin{aligned} \mathsf{true} &\rightarrow \langle \mathsf{s}(0), \mathsf{nd}(\mathsf{nd}(a,b,c), \mathsf{s}(u), \mathsf{lf})\rangle + \langle 0, \mathsf{nd}(a,b,c)\rangle \preceq \\ &\quad \langle 0, \mathsf{nd}(\mathsf{nd}(a,b,c), \mathsf{s}(u), \mathsf{lf})\rangle + \langle \mathsf{s}(0), \mathsf{nd}(a,b,c)\rangle && (44)\\ \neg\mathsf{le}(u,v) &\rightarrow \langle \mathsf{s}(\mathsf{s}(v)), \mathsf{nd}(\mathsf{nd}(a,b,c), \mathsf{s}(u), \mathsf{lf})\rangle + \langle \mathsf{s}(v), \mathsf{nd}(a,b,c)\rangle + \\ &\quad \langle v, \mathsf{nd}(\mathsf{nd}(a,b,c), u, \mathsf{lf})\rangle + \langle \mathsf{s}(v), \mathsf{nd}(a,b,c)\rangle \preceq \\ &\quad \langle \mathsf{s}(v), \mathsf{nd}(\mathsf{nd}(a,b,c), \mathsf{s}(u), \mathsf{lf})\rangle + \langle \mathsf{s}(\mathsf{s}(v)), \mathsf{nd}(a,b,c)\rangle + \\ &\quad \langle \mathsf{s}(v), \mathsf{nd}(\mathsf{nd}(a,b,c), u, \mathsf{lf})\rangle + \langle v, \mathsf{nd}(a,b,c)\rangle && (45)\end{aligned}$$

5. Inductive evaluation of (39) w.r.t. $\mathsf{le}(m,n)$, Symbolic Evaluation:

$$\begin{aligned} \mathsf{true} &\rightarrow \langle \mathsf{s}(0), \mathsf{nd}(\mathsf{nd}(a,b,c), \mathsf{s}(u), \mathsf{nd}(d,e,f))\rangle + \langle 0, \mathsf{nd}(a,b,c)\rangle \preceq \\ &\quad \langle 0, \mathsf{nd}(\mathsf{nd}(a,b,c), \mathsf{s}(u), \mathsf{nd}(d,e,f))\rangle + \langle \mathsf{s}(0), \mathsf{nd}(a,b,c)\rangle && (46)\\ \neg\mathsf{le}(u,v) &\rightarrow \langle \mathsf{s}(\mathsf{s}(v)), \mathsf{nd}(\mathsf{nd}(a,b,c), \mathsf{s}(u), \mathsf{nd}(d,e,f))\rangle + \langle \mathsf{s}(v), \mathsf{nd}(a,b,c)\rangle + \\ &\quad \langle v, \mathsf{nd}(\mathsf{nd}(a,b,c), u, \mathsf{nd}(d,e,f))\rangle + \langle \mathsf{s}(v), \mathsf{nd}(a,b,c)\rangle \preceq \\ &\quad \langle \mathsf{s}(v), \mathsf{nd}(\mathsf{nd}(a,b,c), \mathsf{s}(u), \mathsf{nd}(d,e,f))\rangle + \langle \mathsf{s}(\mathsf{s}(v)), \mathsf{nd}(a,b,c)\rangle + \\ &\quad \langle \mathsf{s}(v), \mathsf{nd}(\mathsf{nd}(a,b,c), u, \mathsf{nd}(d,e,f))\rangle + \langle v, \mathsf{nd}(a,b,c)\rangle && (47)\end{aligned}$$

6. Premise Elimination in (40)-(47)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \mathsf{lf} \mapsto 0,\ \mathsf{nd}(a,b,c) \mapsto a+b+c,\ \langle x,y\rangle \mapsto (x-y)^2$

Note that the inductive evaluations of (37), (38), and (39) turn out not to be necessary for the termination proof, as the mentioned polynomial ordering already satisfies the conclusions of (37) - (39) (i.e. one could have applied premise elimination on these hypotheses directly).

Soundness Predicate

function $\lambda_{(35)}$: $\mathsf{nat} \times \mathsf{tree} \rightarrow \mathsf{bool}$

$\lambda_{(35)}(n,t) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{tree\_minimum}}$: $\mathsf{nat} \times \mathsf{tree} \rightarrow \mathsf{bool}$

$\theta_{\mathsf{tree\_minimum}}(n,t) = \mathsf{true}$

9 eq*

The functional procedure eq computes the equality for naturals.

function $\mathsf{eq} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$$\begin{aligned}
&\mathsf{eq}(0,0) &&= \mathsf{true}\\
&\mathsf{eq}(0,\mathsf{s}(y)) &&= \mathsf{false}\\
&\mathsf{eq}(\mathsf{s}(x),0) &&= \mathsf{false}\\
&\mathsf{eq}(\mathsf{s}(x),\mathsf{s}(y)) &&= \mathsf{eq}(x,y)
\end{aligned}$$

Termination Hypothesis

$$\langle x, y\rangle \prec \langle \mathsf{s}(x), \mathsf{s}(y)\rangle \tag{48}$$

Transformation

No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(x) \mapsto x+1,\ \langle x, y\rangle \mapsto x$

Soundness Predicate

function $\lambda_{(48)} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$\lambda_{(48)}(x,y) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{eq}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$\theta_{\mathsf{eq}}(x,y) = \mathsf{true}$

10 mem*

The functional procedure mem checks if a natural occurs in a given list.

function $\mathsf{mem} : \mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$

$$\begin{aligned}
&\mathsf{mem}(n,\mathsf{empty}) &&= \mathsf{false}\\
&\mathsf{mem}(n,\mathsf{add}(m,v)) &&= \mathsf{if}(\mathsf{eq}(n,m),\ \mathsf{true},\ \mathsf{mem}(n,v))
\end{aligned}$$

Termination Hypothesis

$$\langle n, v\rangle \prec \langle n, \mathsf{add}(m, v)\rangle \tag{49}$$

Transformation

No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \mathsf{empty} \mapsto 0,\ \mathsf{add}(m,v) \mapsto v+1,\ \langle x, y\rangle \mapsto y$

Soundness Predicate

function $\lambda_{(49)} : \mathsf{nat} \times \mathsf{list} \times \mathsf{nat} \to \mathsf{bool}$

$\lambda_{(49)}(n,v,m) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{mem}} : \mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$

$\theta_{\mathsf{mem}}(n,v) = \mathsf{true}$

11 max^*

The functional procedure max computes the maximal element of a list.

function $\mathsf{max} : \mathsf{list} \to \mathsf{nat}$

$$\begin{aligned}
&\mathsf{max}(\mathsf{empty}) &&= 0\\
&\mathsf{max}(\mathsf{add}(u, \mathsf{empty})) &&= u\\
&\mathsf{max}(\mathsf{add}(u, \mathsf{add}(v, w))) &&= \mathsf{if}(\mathsf{le}(u, v),\ \mathsf{max}(\mathsf{add}(v, w)),\ \mathsf{max}(\mathsf{add}(u, w)))
\end{aligned}$$

Termination Hypotheses

$$\mathsf{le}(u, v) \to \mathsf{add}(v, w) \prec \mathsf{add}(u, \mathsf{add}(v, w)) \tag{50}$$

$$\neg\mathsf{le}(u, v) \to \mathsf{add}(u, w) \prec \mathsf{add}(u, \mathsf{add}(v, w)) \tag{51}$$

Transformation

No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \mathsf{empty} \mapsto 0,\ \mathsf{add}(v, w) \mapsto w + 1$

Soundness Predicates

function $\lambda_{(50)} : \mathsf{nat} \times \mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$
$\lambda_{(50)}(u, v, w) = \mathsf{true}$

function $\lambda_{(51)} : \mathsf{nat} \times \mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$
$\lambda_{(51)}(u, v, w) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{max}} : \mathsf{list} \to \mathsf{bool}$
$\theta_{\mathsf{max}}(u, w) = \mathsf{true}$

12 $\mathsf{countsort}$

$\mathsf{countsort}(n, l)$ returns the sorted list of all elements of l which are greater than or equal to n where duplicates are removed. Hence $\mathsf{countsort}(0, l)$ sorts l.

function $\mathsf{countsort} : \mathsf{nat} \times \mathsf{list} \to \mathsf{list}$

$$\begin{aligned}
\mathsf{countsort}(n, l) = \mathsf{if}(&\mathsf{mem}(n, l),\\
&\mathsf{add}(n, \mathsf{countsort}(\mathsf{s}(n), l)),\\
&\mathsf{if}(\mathsf{le}(\mathsf{max}(l), n),\ \mathsf{empty},\ \mathsf{countsort}(\mathsf{s}(n), l)))
\end{aligned}$$

Termination Hypotheses

$$\mathsf{mem}(n, l) \to \langle \mathsf{s}(n), l\rangle \prec \langle n, l\rangle \tag{52}$$

$$\neg\mathsf{mem}(n, l) \wedge \neg\mathsf{le}(\mathsf{max}(l), n) \to \langle \mathsf{s}(n), l\rangle \prec \langle n, l\rangle \tag{53}$$

Transformation

1. Inductive Evaluation of (52) w.r.t. $\mathsf{mem}(n, l)$, Symbolic Evaluation:

$$\mathsf{eq}(n, m) \to \langle \mathsf{s}(n), \mathsf{add}(m, v)\rangle \prec \langle n, \mathsf{add}(m, v)\rangle \tag{54}$$

$$\begin{aligned}\neg\mathsf{eq}(n, m) \wedge \mathsf{mem}(n, v) \to\ & \langle \mathsf{s}(n), \mathsf{add}(m, v)\rangle + \langle n, v\rangle \preceq\\ & \langle n, \mathsf{add}(m, v)\rangle + \langle \mathsf{s}(n), v\rangle\end{aligned} \tag{55}$$

2. Inductive Evaluation of (54) w.r.t. $\mathsf{eq}(n, m)$, Symbolic Evaluation:

$$\mathsf{true} \to \langle \mathsf{s}(0), \mathsf{add}(0, v)\rangle \prec \langle 0, \mathsf{add}(0, v)\rangle \tag{56}$$

$$\begin{aligned}\mathsf{eq}(x, y) \to\ & \langle \mathsf{s}(\mathsf{s}(x)), \mathsf{add}(\mathsf{s}(y), v)\rangle + \langle x, \mathsf{add}(y, v)\rangle \preceq\\ & \langle \mathsf{s}(x), \mathsf{add}(\mathsf{s}(y), v)\rangle + \langle \mathsf{s}(x), \mathsf{add}(y, v)\rangle\end{aligned} \tag{57}$$

3. Inductive Evaluation of (55) w.r.t. $\mathsf{eq}(n,m)$, Symbolic Evaluation:

$$\begin{aligned}\mathsf{true} \wedge \mathsf{mem}(0,v) &\to \langle \mathsf{s}(0), \mathsf{add}(\mathsf{s}(y),v)\rangle + \langle 0,v\rangle \preceq \\ &\quad \langle 0, \mathsf{add}(\mathsf{s}(y),v)\rangle + \langle \mathsf{s}(0),v\rangle & (58)\\ \mathsf{true} \wedge \mathsf{mem}(\mathsf{s}(x),v) &\to \langle \mathsf{s}(\mathsf{s}(x)), \mathsf{add}(0,v)\rangle + \langle \mathsf{s}(x),v\rangle \preceq \\ &\quad \langle \mathsf{s}(x), \mathsf{add}(0,v)\rangle + \langle \mathsf{s}(\mathsf{s}(x)),v\rangle & (59)\\ \neg\mathsf{eq}(x,y) \wedge \mathsf{mem}(\mathsf{s}(x),v) &\to \langle \mathsf{s}(\mathsf{s}(x)), \mathsf{add}(\mathsf{s}(y),v)\rangle + \langle \mathsf{s}(x),v\rangle + \\ &\quad \langle x, \mathsf{add}(y,v)\rangle + \langle \mathsf{s}(x),v\rangle \preceq \\ &\quad \langle \mathsf{s}(x), \mathsf{add}(\mathsf{s}(y),v)\rangle + \langle \mathsf{s}(\mathsf{s}(x)),v\rangle + \\ &\quad \langle \mathsf{s}(x), \mathsf{add}(y,v)\rangle + \langle x,v\rangle & (60)\end{aligned}$$

4. Inductive Evaluation of (53) w.r.t. $\mathsf{max}(l)$, Symbolic Evaluation. (Note that according to our heuristic there are two inductive evaluations possible, i.e. we could also perform inductive evaluation w.r.t. $\mathsf{mem}(n,l)$. To decide between several suggested inductive evaluations we use the heuristic to apply the one with minimal number of recursive cases where the induction hypothesis cannot be applied. In our example, if one would use inductive evaluation w.r.t. mem, then the induction hypothesis would not be applicable because the premise of the induction conclusion would not imply the premise of the induction hypothesis. Hence, this heuristic decides to use inductive evaluation w.r.t. max instead.)

$$\begin{aligned}&\neg\mathsf{mem}(n,\mathsf{add}(u,\mathsf{empty})) \wedge \neg\mathsf{le}(u,n) \to \\ &\qquad \langle \mathsf{s}(n), \mathsf{add}(u,\mathsf{empty})\rangle \prec \langle n, \mathsf{add}(u,\mathsf{empty})\rangle & (61)\\ &\neg\mathsf{mem}(n,\mathsf{add}(u,\mathsf{add}(v,w))) \wedge \neg\mathsf{le}(\mathsf{max}(\mathsf{add}(v,w)),n) \wedge \mathsf{le}(u,v) \to \\ &\qquad \langle \mathsf{s}(n), \mathsf{add}(u,\mathsf{add}(v,w))\rangle + \langle n, \mathsf{add}(v,w)\rangle \preceq \\ &\qquad \langle n, \mathsf{add}(u,\mathsf{add}(v,w))\rangle + \langle \mathsf{s}(n), \mathsf{add}(v,w)\rangle & (62)\\ &\neg\mathsf{mem}(n,\mathsf{add}(u,\mathsf{add}(v,w))) \wedge \neg\mathsf{le}(\mathsf{max}(\mathsf{add}(u,w)),n) \wedge \neg\mathsf{le}(u,v) \to \\ &\qquad \langle \mathsf{s}(n), \mathsf{add}(u,\mathsf{add}(v,w))\rangle + \langle n, \mathsf{add}(u,w)\rangle \preceq \\ &\qquad \langle n, \mathsf{add}(u,\mathsf{add}(v,w))\rangle + \langle \mathsf{s}(n), \mathsf{add}(u,w)\rangle & (63)\end{aligned}$$

5. Inductive Evaluation of (61) w.r.t. $\mathsf{le}(u,n)$, Symbolic Evaluation.

$$\begin{aligned}&\mathsf{true} \to \langle \mathsf{s}(0), \mathsf{add}(\mathsf{s}(x),\mathsf{empty})\rangle \prec \langle 0, \mathsf{add}(\mathsf{s}(x),\mathsf{empty})\rangle & (64)\\ &\neg\mathsf{mem}(\mathsf{s}(y),\mathsf{add}(\mathsf{s}(x),\mathsf{empty})) \wedge \neg\mathsf{le}(x,y) \to \\ &\qquad \langle \mathsf{s}(\mathsf{s}(y)), \mathsf{add}(\mathsf{s}(x),\mathsf{empty})\rangle + \langle y, \mathsf{add}(x,\mathsf{empty})\rangle \preceq \\ &\qquad \langle \mathsf{s}(y), \mathsf{add}(\mathsf{s}(x),\mathsf{empty})\rangle + \langle \mathsf{s}(y), \mathsf{add}(x,\mathsf{empty})\rangle & (65)\end{aligned}$$

6. Inductive Evaluation of (62) w.r.t. $\mathsf{le}(u,v)$, Symbolic Evaluation. (This transforms (62) into two new formulas. The first formula is (62) where u is instantiated with 0 and v is instantiated with y and the other one is (62) where u is instantiated with $\mathsf{s}(x)$ and v is instantiated with $\mathsf{s}(y)$, as the induction hypothesis cannot be applied.)
7. Inductive Evaluation of (63) w.r.t. $\mathsf{le}(u,v)$, Symbolic Evaluation. (This transforms (63) into two new formulas. The first formula is (63) where u is instantiated with $\mathsf{s}(x)$ and v is instantiated with 0 and the other one is (63) where u is instantiated with $\mathsf{s}(x)$ and v is instantiated with $\mathsf{s}(y)$, as the induction hypothesis cannot be applied.)
8. Premise Elimination in all resulting inequalities.

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(x) \mapsto x+1,\ \mathsf{empty} \mapsto 0,\ \mathsf{add}(v,w) \mapsto v+w+1,\ \langle x,y\rangle \mapsto (x-y)^2$

Soundness Predicates

function $\lambda_{(52)}$: nat × list → bool
$\lambda_{(52)}(n,l) = \mathsf{true}$

function $\lambda_{(53)}$: nat $\times$ list $\rightarrow$ bool
$\lambda_{(53)}(n, l) =$ true

Termination Predicate

function $\theta_{\mathsf{countsort}}$: nat $\times$ list $\rightarrow$ bool
$\theta_{\mathsf{countsort}}(n, l) =$ true

13 divisible_3*

The functional procedure divisible_3 returns true iff the input is a multiple of 3.

function divisible_3 : nat $\rightarrow$ nat

$$\begin{aligned}
&\mathsf{divisible\_3}(0) &&= \mathsf{true}\\
&\mathsf{divisible\_3}(\mathsf{s}(0)) &&= \mathsf{false}\\
&\mathsf{divisible\_3}(\mathsf{s}(\mathsf{s}(0))) &&= \mathsf{false}\\
&\mathsf{divisible\_3}(\mathsf{s}(\mathsf{s}(\mathsf{s}(y)))) &&= \mathsf{divisible\_3}(y)
\end{aligned}$$

Termination Hypothesis

$$y \prec \mathsf{s}(\mathsf{s}(\mathsf{s}(y))) \tag{66}$$

Transformation

No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(y) \mapsto y + 1$

Soundness Predicate

function $\lambda_{(66)}$: nat $\rightarrow$ bool
$\lambda_{(66)}(y) =$ true

Termination Predicate

function $\theta_{\mathsf{divisible\_3}}$: nat $\rightarrow$ bool
$\theta_{\mathsf{divisible\_3}}(y) =$ true

14 round_3

round_3(x) is the least natural greater than or equal to x which is a multiple of 3.

function round_3 : nat $\rightarrow$ nat
round_3(x) = if(divisible_3(x), x, round_3(s(x)))

Termination Hypothesis

$$\neg\mathsf{divisible\_3}(x) \ \rightarrow\ \mathsf{s}(x) \prec x \tag{67}$$

Transformation

1. Inductive Evaluation w.r.t. divisible_3(x), Symbolic Evaluation:

$$\mathsf{true} \ \rightarrow\ \mathsf{s}(\mathsf{s}(0)) \prec \mathsf{s}(0) \tag{68}$$

$$\mathsf{true} \ \rightarrow\ \mathsf{s}(\mathsf{s}(\mathsf{s}(0))) \prec \mathsf{s}(\mathsf{s}(0)) \tag{69}$$

$$\mathsf{divisible\_3}(y) \ \rightarrow\ \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(y)))) + y \preceq \mathsf{s}(\mathsf{s}(\mathsf{s}(y))) + \mathsf{s}(y) \tag{70}$$

2. Premise Elimination in (68)-(70)

Polynomial Ordering

For this example we need a polynomial ordering with *rational* (instead of integer) coefficients, cf. [Gie95b]. To ensure that such polynomial orderings are still well founded, for each polynomial ordering there must be a number $\epsilon > 0$ and we say that a term r is smaller than a term t w.r.t. the polynomial ordering iff the difference between the number corresponding to t and the number corresponding to r is at least ϵ. The methods for the automated generation of polynomial orderings presented in [Gie95a, Gie95b] can also generate such rational polynomial orderings. In this way, termination of round_3 can be proved using the following association: $0 \mapsto 0,\ \mathsf{s}(x) \mapsto \frac{3}{2}x^2 - \frac{7}{2}x + 2$. Now data objects $\mathsf{s}^n(0)$ with $n \bmod 3 = 0$ are mapped to 0, data objects $\mathsf{s}^n(0)$ with $n \bmod 3 = 1$ are mapped to 2, and data objects $\mathsf{s}^n(0)$ with $n \bmod 3 = 2$ are mapped to 1.

Soundness Predicate

function $\lambda_{(67)}$: nat → bool
$\lambda_{(67)}(x) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{round\_3}}$: nat → bool
$\theta_{\mathsf{round\_3}}(x) = \mathsf{true}$

15 app*

The algorithm $\mathsf{app}(x, y)$ appends the tree y to the rightmost leaf of x.

function app : sexpr × sexpr → sexpr
$\mathsf{app}(\mathsf{nil}, y) = y$
$\mathsf{app}(\mathsf{cons}(u, v), y) = \mathsf{cons}(u, \mathsf{app}(v, y))$

Termination Hypothesis

$$\langle v, y\rangle \prec \langle \mathsf{cons}(u, v), y\rangle \tag{71}$$

Transformation

No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering

$\mathsf{nil} \mapsto 0,\ \mathsf{cons}(u, v) \mapsto v + 1,\ \langle u, y\rangle \mapsto u$

Soundness Predicate

function $\lambda_{(71)}$: sexpr × sexpr × sexpr → bool
$\lambda_{(71)}(v, y, u) = \mathsf{true}$

Termination Predicate

function θ_{app} : sexpr × sexpr × sexpr → bool
$\theta_{\mathsf{app}}(v, y, u) = \mathsf{true}$

16 less_leaves*

The following procedure $\mathsf{less\_leaves}(x, y)$ returns true iff x has less leaves than y.

function less_leaves : sexpr × sexpr → bool
$\mathsf{less\_leaves}(x, \mathsf{nil}) = \mathsf{false}$
$\mathsf{less\_leaves}(\mathsf{nil}, \mathsf{cons}(u, v)) = \mathsf{true}$
$\mathsf{less\_leaves}(\mathsf{cons}(x, y), \mathsf{cons}(u, v)) = \mathsf{less\_leaves}(\mathsf{app}(x, y), \mathsf{app}(u, v))$

Termination Hypothesis

$$\langle \mathsf{app}(x,y), \mathsf{app}(u,v)\rangle \prec \langle \mathsf{cons}(x,y), \mathsf{cons}(u,v)\rangle \tag{72}$$

Transformation

1. The termination hypothesis of less_leaves contains the defined symbol app in its *conclusion.* To eliminate defined symbols from conclusions (instead of premises), we use the method presented in [Gie95a, Gie95b, GWB98] and we refer the reader to these papers for a detailed explanation. In this method there are two rules to eliminate defined symbols from conclusions, viz. *generalization* and *estimation.* For example, we can eliminate the first app-term $\mathsf{app}(x,y)$ by generalization, i.e. we replace it by a fresh variable. In this way, (72) is transformed into
$$\langle z, \mathsf{app}(u,v)\rangle \prec \langle \mathsf{cons}(x,y), \mathsf{cons}(u,v)\rangle. \tag{73}$$

2. To eliminate the other occurrence of app we use the estimation rule. Here, the central idea is an *estimation* of defined function symbols by new *free* function symbols (i.e. function symbols like constructors which are not defined by an algorithm). Therefore app is replaced by a new free function symbol $\overline{\mathsf{app}}$ and we demand that the result of $\overline{\mathsf{app}}$ is always greater or equal than the result of app, i.e. $\mathsf{app}(x,y) \preceq \overline{\mathsf{app}}(x,y)$. More precisely, we compute a set of *estimation inequalities* $\mathcal{E}_{\mathsf{app} \preceq \overline{\mathsf{app}}}$ (without defined function symbols) which imply $\mathsf{app}(x,y) \preceq \overline{\mathsf{app}}(x,y)$ and we also compute a strictness predicate $\delta_{\mathsf{app} \prec \overline{\mathsf{app}}}$ which returns true iff the result of app is strictly smaller than the result of $\overline{\mathsf{app}}$. Then we can replace (73) by[10]
$$\langle z, \overline{\mathsf{app}}(u,v)\rangle \preceq \langle \mathsf{cons}(x,y), \mathsf{cons}(u,v)\rangle, \tag{74}$$
$$u \prec v \rightarrow \langle z, u\rangle \prec \langle z, v\rangle, \tag{75}$$
$$\mathcal{E}_{\mathsf{app} \preceq \overline{\mathsf{app}}}, \tag{76}$$
$$\delta_{\mathsf{app} \prec \overline{\mathsf{app}}}(u,v) \vee \langle z, \overline{\mathsf{app}}(u,v)\rangle \prec \langle \mathsf{cons}(x,y), \mathsf{cons}(u,v)\rangle. \tag{77}$$

3. The estimation inequalities $\mathcal{E}_{\mathsf{app} \preceq \overline{\mathsf{app}}}$ are constructed inductively w.r.t. the algorithm app. For that purpose instead of $\mathsf{app}(x,y) \preceq \overline{\mathsf{app}}(x,y)$ we consider each case separately and in the induction step (i.e. in the case $\mathsf{app}(\mathsf{cons}(u,v),y) \preceq \overline{\mathsf{app}}(\mathsf{cons}(u,v),y))$ we use $\mathsf{app}(v,y) \preceq \overline{\mathsf{app}}(v,y)$ as induction hypothesis. In this way, $\mathcal{E}_{\mathsf{app} \preceq \overline{\mathsf{app}}}$ is automatically computed as follows:
$$y \preceq \overline{\mathsf{app}}(\mathsf{nil},y), \tag{78}$$
$$\mathsf{cons}(u, \overline{\mathsf{app}}(v,y)) \preceq \overline{\mathsf{app}}(\mathsf{cons}(u,v),y), \tag{79}$$
$$y \prec z \rightarrow \mathsf{cons}(u,y) \prec \mathsf{cons}(u,z). \tag{80}$$

4. In a similar way, one can compute the corresponding strictness predicate.
```
**function** $\delta_{\mathsf{app} \prec \overline{\mathsf{app}}} : \mathsf{sexpr} \times \mathsf{sexpr} \rightarrow \mathsf{bool}$
  $\delta_{\mathsf{app} \prec \overline{\mathsf{app}}}(\mathsf{nil},y) = y \prec \overline{\mathsf{app}}(\mathsf{nil},y)$
  $\delta_{\mathsf{app} \prec \overline{\mathsf{app}}}(\mathsf{cons}(u,v),y) = \delta_{\mathsf{app} \prec \overline{\mathsf{app}}}(v,y) \vee \mathsf{cons}(u,\overline{\mathsf{app}}(v,y)) \prec \overline{\mathsf{app}}(\mathsf{cons}(u,v),y)$
```
In general, there are two possibilities to eliminate the defined symbol $\delta_{\mathsf{app} \prec \overline{\mathsf{app}}}$ from a formula $\psi \rightarrow \delta_{\mathsf{app} \prec \overline{\mathsf{app}}}(q^*) \vee \varphi$. Either we omit the first part of the disjunction, i.e. we transform $\psi \rightarrow \delta_{\mathsf{app} \prec \overline{\mathsf{app}}}(q^*) \vee \varphi$ into φ or else we replace this formula by a subset of the inequalities from the algorithm $\delta_{\mathsf{app} \prec \overline{\mathsf{app}}}$ that is sufficient for $\psi \rightarrow \delta_{\mathsf{app} \prec \overline{\mathsf{app}}}(q^*)$. In our example, we will use the first possibility and transform (77) into
$$\langle z, \overline{\mathsf{app}}(u,v)\rangle \prec \langle \mathsf{cons}(x,y), \mathsf{cons}(u,v)\rangle. \tag{81}$$

[10] Note that while the existing methods for the generation of polynomial orderings cannot deal with arbitrary conditions, they can nevertheless handle monotonicity formulas like (75), cf. [Gie95b, AG97]. Hence, when performing premise elimination, these formulas are always left unchanged.

Polynomial Ordering

$\mathsf{nil} \mapsto 0,\ \mathsf{cons}(u,v) \mapsto u+v+1,\ \overline{\mathsf{app}}(u,v) \mapsto u+v, \langle u,v\rangle \mapsto v$

Soundness Predicate

function $\lambda_{(72)}$: sexpr × sexpr × sexpr × sexpr → bool
$\lambda_{(72)}(x,y,u,v) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{less\_leaves}}$: sexpr × sexpr → bool
$\theta_{\mathsf{less\_leaves}}(x,u) = \mathsf{true}$

17 leaf_difference

The functional procedure leaf_difference(x,y) computes the difference between the number of leaves in x and the number of leaves in y.

```
function leaf_difference : sexpr × sexpr → nat
  leaf_difference(x, y) = if( less_leaves(x, y),
                              s(leaf_difference(cons(nil, x), y)),
                              0)
```

Termination Hypothesis

In this example we have to use a polynomial ordering where tuples of data objects may also be mapped to negative numbers. However, we will demand that all those tuples of data objects *which lead to recursive calls* are still mapped to non-negative numbers. In this way, such (possibly non-well-founded) polynomial orderings can still be used for termination proofs, cf. [Gie95b]. For that purpose we add a second termination hypothesis where zero is a new function symbol which must be associated with the number 0 by all polynomial orderings.

$$\mathsf{less\_leaves}(x,y) \rightarrow \langle \mathsf{cons}(\mathsf{nil},x), y\rangle \prec \langle x,y\rangle \tag{82}$$

$$\mathsf{less\_leaves}(x,y) \rightarrow \mathsf{zero} \preceq \langle x,y\rangle \tag{83}$$

In the following examples we will only mention this second termination hypotheses where it is needed, i.e. to ease readability we omit it in those examples where we use a polynomial ordering which maps all tuples of data objects to natural numbers.

Transformation

1. Inductive Evaluation of (82) w.r.t. less_leaves(x,y), Symbolic Evaluation:

$$\mathsf{true} \rightarrow \langle \mathsf{cons}(\mathsf{nil},\mathsf{nil}), \mathsf{cons}(u,v)\rangle \prec \langle \mathsf{nil}, \mathsf{cons}(u,v)\rangle \tag{84}$$

$$\begin{aligned}\mathsf{less\_leaves}(\mathsf{app}(x,y),\mathsf{app}(u,v)) \rightarrow \\ \langle \mathsf{cons}(\mathsf{nil},\mathsf{cons}(x,y)), \mathsf{cons}(u,v)\rangle + \langle \mathsf{app}(x,y), \mathsf{app}(u,v)\rangle \preceq \\ \langle \mathsf{cons}(x,y), \mathsf{cons}(u,v)\rangle + \langle \mathsf{cons}(\mathsf{nil},\mathsf{app}(x,y)), \mathsf{app}(u,v)\rangle\end{aligned} \tag{85}$$

2. Inductive Evaluation of (83) w.r.t. less_leaves(x,y), Symbolic Evaluation:

$$\mathsf{true} \rightarrow \mathsf{zero} \preceq \langle \mathsf{nil}, \mathsf{cons}(u,v)\rangle \tag{86}$$

$$\mathsf{less\_leaves}(\mathsf{app}(x,y),\mathsf{app}(u,v)) \rightarrow \langle \mathsf{app}(x,y), \mathsf{app}(u,v)\rangle \preceq \langle \mathsf{cons}(x,y), \mathsf{cons}(u,v)\rangle \tag{87}$$

To ease readability we omitted additions of zero in these inequalities.

3. Premise Elimination in (84)-(87).

$$\langle \mathsf{cons}(\mathsf{nil},\mathsf{nil}),\mathsf{cons}(u,v)\rangle \prec \langle \mathsf{nil},\mathsf{cons}(u,v)\rangle \tag{88}$$

$$\langle \mathsf{cons}(\mathsf{nil},\mathsf{cons}(x,y)),\mathsf{cons}(u,v)\rangle + \langle \mathsf{app}(x,y),\mathsf{app}(u,v)\rangle \preceq \langle \mathsf{cons}(x,y),\mathsf{cons}(u,v)\rangle + \langle \mathsf{cons}(\mathsf{nil},\mathsf{app}(x,y)),\mathsf{app}(u,v)\rangle \tag{89}$$

$$\mathsf{zero} \preceq \langle \mathsf{nil},\mathsf{cons}(u,v)\rangle \tag{90}$$

$$\langle \mathsf{app}(x,y),\mathsf{app}(u,v)\rangle \preceq \langle \mathsf{cons}(x,y),\mathsf{cons}(u,v)\rangle \tag{91}$$

4. Estimation of app in (89) and (91). Similar to the estimation of app by an upper bound $\overline{\mathsf{app}}$, [Gie95b] also introduced estimation by lower bounds $\underline{\mathsf{app}}$. In our example, the heuristics of [Gie95b] suggest to estimate the first argument of tuples by lower bounds and the second argument by upper bounds. Hence, (89) and (91) are transformed into

$$\langle \mathsf{cons}(\mathsf{nil},\mathsf{cons}(x,y)),\mathsf{cons}(u,v)\rangle + \langle \underline{\mathsf{app}}(x,y),\overline{\mathsf{app}}(u,v)\rangle \preceq \langle \mathsf{cons}(x,y),\mathsf{cons}(u,v)\rangle + \langle \mathsf{cons}(\mathsf{nil},\underline{\mathsf{app}}(x,y)),\overline{\mathsf{app}}(u,v)\rangle \tag{92}$$

$$\langle \underline{\mathsf{app}}(x,y),\overline{\mathsf{app}}(u,v)\rangle \preceq \langle \mathsf{cons}(x,y),\mathsf{cons}(u,v)\rangle \tag{93}$$

$$u \prec v \rightarrow \langle v,y\rangle \prec \langle u,y\rangle \tag{94}$$

$$u \prec v \rightarrow \langle x,u\rangle \prec \langle x,v\rangle \tag{95}$$

$$\mathcal{E}_{\mathsf{app} \preceq \overline{\mathsf{app}}} \tag{96}$$

$$\mathcal{E}_{\underline{\mathsf{app}} \preceq \mathsf{app}} \tag{97}$$

where the lower-bound estimation inequalities $\mathcal{E}_{\underline{\mathsf{app}} \preceq \mathsf{app}}$ are constructed analogously to the upper-bound estimation inequalities.

Polynomial Ordering

$\mathsf{nil} \mapsto 0$, $\mathsf{cons}(u,v) \mapsto u+v+1$, $\overline{\mathsf{app}}(u,v) \mapsto u+v$, $\underline{\mathsf{app}}(u,v) \mapsto u+v$, $\langle x,y\rangle \mapsto y-x$

Soundness Predicates

function $\lambda_{(82)}$: sexpr × sexpr → bool
$\lambda_{(82)}(x,y)$ = true

The construction of soundness predicates for the new types of termination hypotheses

$$\ldots \rightarrow \mathsf{zero} \preceq \ldots$$

is completely analogous to the generation of soundness predicates for the other termination hypotheses. Hence, we obtain the following soundness predicate for the second termination hypothesis (83).

function $\lambda_{(83)}$: sexpr × sexpr → bool
$\lambda_{(83)}(x,y)$ = true

Termination Predicate

Now the termination predicates contain inequalities corresponding to both termination hypotheses, because a termination predicate may only be called recursively, if both termination hypotheses hold for the current arguments. Hence, both termination hypotheses are replaced by their soundness predicates (which corresponds to the 'conjunction' of the two soundness predicates).

```
function $\theta_{\mathsf{leaf\_difference}}$ : sexpr × sexpr → bool
  $\theta_{\mathsf{leaf\_difference}}(x, y)$ = if( less_leaves($x, y$),
                                  if( $\lambda_{(82)}(x, y)$,
                                      if( $\lambda_{(83)}(x, y)$,
                                          $\theta_{\mathsf{leaf\_difference}}$(cons(nil, $x$), $y$),
                                          false),
                                      false),
                                  true)
```

Of course, this algorithm can be simplified to

function $\theta_{\mathsf{leaf\_difference}}$: $\mathsf{sexpr} \times \mathsf{sexpr} \to \mathsf{bool}$
  $\theta_{\mathsf{leaf\_difference}}(x, y) = \mathsf{true}$.

18 dbl*

The following functional procedure doubles a natural.

function dbl : $\mathsf{nat} \to \mathsf{nat}$
  $\mathsf{dbl}(0) \quad = 0$
  $\mathsf{dbl}(\mathsf{s}(v)) = \mathsf{s}(\mathsf{s}(\mathsf{dbl}(v)))$

Termination Hypothesis

$$v \prec \mathsf{s}(v) \tag{98}$$

Transformation
  No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering
  $0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1$

Soundness Predicate
  function $\lambda_{(98)}$: $\mathsf{nat} \to \mathsf{bool}$
    $\lambda_{(98)}(v) = \mathsf{true}$

Termination Predicate
  function θ_{dbl} : $\mathsf{nat} \to \mathsf{bool}$
    $\theta_{\mathsf{dbl}}(v) = \mathsf{true}$

19 plus*

The functional procedure plus computes the addition for naturals.

function plus : $\mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$
  $\mathsf{plus}(x, 0) \quad = x$
  $\mathsf{plus}(x, \mathsf{s}(y)) = \mathsf{s}(\mathsf{plus}(x, y))$

Termination Hypothesis

$$\langle x, y\rangle \prec \langle x, \mathsf{s}(y)\rangle \tag{99}$$

Transformation
  No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering
  $0 \mapsto 0,\ \mathsf{s}(x) \mapsto x + 1,\ \langle x, y\rangle \mapsto y$

Soundness Predicate
  function $\lambda_{(99)}$: $\mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
    $\lambda_{(99)}(x, y) = \mathsf{true}$

Termination Predicate
  function θ_{plus} : $\mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
    $\theta_{\mathsf{plus}}(x, y) = \mathsf{true}$

20 sqr*

The functional procedure sqr multiplies a natural with itself.

function $\mathsf{sqr} : \mathsf{nat} \to \mathsf{nat}$
$$\begin{aligned} \mathsf{sqr}(0) &= 0 \\ \mathsf{sqr}(\mathsf{s}(v)) &= \mathsf{s}(\mathsf{plus}(\mathsf{sqr}(v), \mathsf{dbl}(v))) \end{aligned}$$

Termination Hypothesis

$$v \prec \mathsf{s}(v) \tag{100}$$

Transformation
No transformation necessary (i.e. direct application of Premise Elimination).

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1$

Soundness Predicate
function $\lambda_{(100)} : \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(100)}(v) = \mathsf{true}$

Termination Predicate
function $\theta_{\mathsf{sqr}} : \mathsf{nat} \to \mathsf{bool}$
$\theta_{\mathsf{sqr}}(v) = \mathsf{true}$

21 bin_log

$\mathsf{bin\_log}(x, y)$ computes the least power of 2 greater than or equal to $\log_y(x)$ where we define $\mathsf{bin\_log}(x, y) = 1$ for $x \leq 1$ or $y \leq 1$.

function $\mathsf{bin\_log} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$
$$\begin{aligned} \mathsf{bin\_log}(x, 0) &= \mathsf{s}(0) \\ \mathsf{bin\_log}(x, \mathsf{s}(0)) &= \mathsf{s}(0) \\ \mathsf{bin\_log}(x, \mathsf{s}(\mathsf{s}(z))) &= \mathsf{if}(\,\mathsf{le}(x, \mathsf{s}(\mathsf{s}(z))), \\ &\qquad \mathsf{s}(0), \\ &\qquad \mathsf{dbl}(\mathsf{bin\_log}(x, \mathsf{sqr}(\mathsf{s}(\mathsf{s}(z)))))) \end{aligned}$$

Termination Hypotheses

$$\neg\mathsf{le}(x, \mathsf{s}(\mathsf{s}(z))) \ \to\ \langle x, \mathsf{sqr}(\mathsf{s}(\mathsf{s}(z)))\rangle \prec \langle x, \mathsf{s}(\mathsf{s}(z))\rangle \tag{101}$$
$$\neg\mathsf{le}(x, \mathsf{s}(\mathsf{s}(z))) \ \to\ \mathsf{zero} \preceq \langle x, \mathsf{s}(\mathsf{s}(z))\rangle \tag{102}$$

Transformation

1. Our heuristics do not suggest inductive evaluation, as no algorithm is applied to pairwise different variables in the premises of the termination hypotheses. Hence, while termination analysis of all other algorithms in this section could be done fully automatically, here we need one user interaction to perform a generalization, i.e. to replace the term $\mathsf{s}(\mathsf{s}(z))$ in the second termination hypothesis (102) by a new variable y. So (102) is transformed into

$$\neg\mathsf{le}(x, y) \ \to\ \mathsf{zero} \preceq \langle x, y\rangle. \tag{103}$$

   Now the heuristic suggests inductive evaluation of (103) w.r.t. $\mathsf{le}(x, y)$ and Symbolic Evaluation:

$$\mathsf{true} \ \to\ \mathsf{zero} \preceq \langle \mathsf{s}(u), 0\rangle \tag{104}$$
$$\neg\mathsf{le}(u, v) \ \to\ \langle u, v\rangle \preceq \langle \mathsf{s}(u), \mathsf{s}(v)\rangle. \tag{105}$$

2. Estimation of sqr by a lower bound in (101).

$$\langle x, \underline{\mathsf{sqr}}(\mathsf{s}(\mathsf{s}(z)))\rangle \preceq \langle x, \mathsf{s}(\mathsf{s}(z))\rangle \tag{106}$$

$$u \prec v \rightarrow \langle x, v\rangle \prec \langle x, u\rangle \tag{107}$$

$$\mathcal{E}_{\underline{\mathsf{sqr}} \preceq \mathsf{sqr}} \tag{108}$$

$$\neg\mathsf{le}(x, \mathsf{s}(\mathsf{s}(z))) \rightarrow \delta_{\underline{\mathsf{sqr}} \prec \mathsf{sqr}}(\mathsf{s}(\mathsf{s}(z))) \vee \langle x, \underline{\mathsf{sqr}}(\mathsf{s}(\mathsf{s}(z)))\rangle \prec \langle x, \mathsf{s}(\mathsf{s}(z))\rangle. \tag{109}$$

3. Computing the estimation inequalities $\mathcal{E}_{\underline{\mathsf{sqr}} \preceq \mathsf{sqr}}$.

$$\underline{\mathsf{sqr}}(0) \preceq 0, \tag{110}$$

$$\underline{\mathsf{sqr}}(\mathsf{s}(v)) \preceq \mathsf{s}(\underline{\mathsf{plus}}(\underline{\mathsf{sqr}}(v), w)) \tag{111}$$

$$u \prec v \rightarrow \mathsf{s}(u) \prec \mathsf{s}(v) \tag{112}$$

$$u \prec v \rightarrow \mathsf{s}(\underline{\mathsf{plus}}(u, w)) \prec \mathsf{s}(\underline{\mathsf{plus}}(v, w)) \tag{113}$$

$$\mathcal{E}_{\underline{\mathsf{plus}} \preceq \mathsf{plus}} \tag{114}$$

where $\mathcal{E}_{\underline{\mathsf{plus}} \preceq \mathsf{plus}}$ is

$$\underline{\mathsf{plus}}(x, 0) \preceq x \tag{115}$$

$$\underline{\mathsf{plus}}(x, \mathsf{s}(y)) \preceq \mathsf{s}(\underline{\mathsf{plus}}(x, y)) \tag{116}$$

$$u \prec v \rightarrow \mathsf{s}(u) \prec \mathsf{s}(v) \tag{117}$$

4. Elimination of the strictness predicate call from (109). The strictness predicate for sqr is computed as

```
**function** $\delta_{\underline{\mathsf{sqr}} \prec \mathsf{sqr}} : \mathsf{nat} \rightarrow \mathsf{bool}$
    $\delta_{\underline{\mathsf{sqr}} \prec \mathsf{sqr}}(0) = \underline{\mathsf{sqr}}(0) \prec 0$
    $\delta_{\underline{\mathsf{sqr}} \prec \mathsf{sqr}}(\mathsf{s}(v)) = \delta_{\underline{\mathsf{sqr}} \prec \mathsf{sqr}}(v) \vee \delta_{\underline{\mathsf{plus}} \prec \mathsf{plus}}(\mathsf{sqr}(v), \mathsf{dbl}(v)) \vee$
                                  $\underline{\mathsf{sqr}}(\mathsf{s}(v)) \prec \mathsf{s}(\underline{\mathsf{plus}}(\underline{\mathsf{sqr}}(v), w))$
```

and the strictness predicate for plus is

```
**function** $\delta_{\underline{\mathsf{plus}} \prec \mathsf{plus}} : \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
    $\delta_{\underline{\mathsf{plus}} \prec \mathsf{plus}}(x, 0) = \underline{\mathsf{plus}}(x, 0) \prec x$
    $\delta_{\underline{\mathsf{plus}} \prec \mathsf{plus}}(x, \mathsf{s}(y)) = \delta_{\underline{\mathsf{plus}} \prec \mathsf{plus}}(x, y) \vee \underline{\mathsf{plus}}(x, \mathsf{s}(y)) \prec \mathsf{s}(\underline{\mathsf{plus}}(x, y))$
```

In this example, we omit the second part of the disjunction (109), i.e. we replace this formula by a set of inequalities from $\delta_{\underline{\mathsf{sqr}} \prec \mathsf{sqr}}$ and $\delta_{\underline{\mathsf{plus}} \prec \mathsf{plus}}$ that is sufficient for $\neg\mathsf{le}(x, \mathsf{s}(\mathsf{s}(z))) \rightarrow \delta_{\underline{\mathsf{sqr}} \prec \mathsf{sqr}}(\mathsf{s}(\mathsf{s}(z)))$. For example, we can replace (109) by the inequality $\underline{\mathsf{plus}}(x, \mathsf{s}(y)) \prec \mathsf{s}(\underline{\mathsf{plus}}(x, y))$ from $\delta_{\underline{\mathsf{plus}} \prec \mathsf{plus}}$'s second defining equation.

5. Premise Elimination in all remaining formulas.

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \underline{\mathsf{sqr}}(x) \mapsto x,\ \underline{\mathsf{plus}}(x, y) \mapsto x,\ \langle x, y\rangle \mapsto x - y$

Soundness Predicates

```
**function** $\lambda_{(101)} : \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
    $\lambda_{(101)}(x, z) = \mathsf{true}$
**function** $\lambda_{(102)} : \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
    $\lambda_{(102)}(x, z) = \mathsf{true}$
```

Termination Predicate

```
**function** $\theta_{\mathsf{bin\_log}} : \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
    $\theta_{\mathsf{bin\_log}}(x, z) = \mathsf{true}$
```

22 p*

The following functional procedure computes the predecessor of a natural.

function $\mathsf{p} : \mathsf{nat} \to \mathsf{nat}$
  $\mathsf{p}(0) \quad = 0$
  $\mathsf{p}(\mathsf{s}(u)) = u$

As p has no recursive calls, there is no termination hypothesis for p and hence, its termination is trivially proved.

23 minus2*

The functional procedure minus2 computes the subtraction for naturals.

function $\mathsf{minus2} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$
  $\mathsf{minus2}(x, y) = \mathsf{if}(\mathsf{le}(x, y),\ 0,\ \mathsf{s}(\mathsf{minus2}(\mathsf{p}(x), y)))$

Termination Hypothesis

$$\neg\mathsf{le}(x, y) \ \to\ \langle \mathsf{p}(x), y\rangle \prec \langle x, y\rangle \tag{118}$$

Transformation

1. Termination of minus2 can be proved without inductive evaluation. The defined symbol p has to be eliminated by estimation.

$$\langle \overline{\mathsf{p}}(x), y\rangle \preceq \langle x, y\rangle \tag{119}$$

$$u \prec v \to \langle u, y\rangle \prec \langle v, y\rangle \tag{120}$$

$$\mathcal{E}_{\mathsf{p}\,\preceq\,\overline{\mathsf{p}}} \tag{121}$$

$$\neg\mathsf{le}(x, y) \to \delta_{\mathsf{p}\,\prec\,\overline{\mathsf{p}}}(x) \vee \langle \overline{\mathsf{p}}(x), y\rangle \prec \langle x, y\rangle \tag{122}$$

where $\mathcal{E}_{\mathsf{p}\,\preceq\,\overline{\mathsf{p}}}$ is

$$0 \preceq \overline{\mathsf{p}}(0) \tag{123}$$

$$u \preceq \overline{\mathsf{p}}(\mathsf{s}(u)) \tag{124}$$

and $\delta_{\mathsf{p}\,\prec\,\overline{\mathsf{p}}}$ is computed as

function $\delta_{\mathsf{p}\,\prec\,\overline{\mathsf{p}}} : \mathsf{nat} \to \mathsf{bool}$
  $\delta_{\mathsf{p}\,\prec\,\overline{\mathsf{p}}}(0) \quad = 0 \prec \overline{\mathsf{p}}(0)$
  $\delta_{\mathsf{p}\,\prec\,\overline{\mathsf{p}}}(\mathsf{s}(u)) = u \prec \overline{\mathsf{p}}(\mathsf{s}(u))$

As the inequality $u \prec \overline{\mathsf{p}}(\mathsf{s}(u))$ from $\delta_{\mathsf{p}\,\prec\,\overline{\mathsf{p}}}$'s second defining equation is sufficient for $\neg_{\mathsf{le}}(x, y) \to \delta_{\mathsf{p}\,\prec\,\overline{\mathsf{p}}}(x)$, the formula (122) is transformed into

$$u \prec \overline{\mathsf{p}}(\mathsf{s}(u)) \tag{125}$$

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \overline{\mathsf{p}}(x) \mapsto x,\ \langle u, v\rangle \mapsto u$

Soundness Predicate

function $\lambda_{(118)} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
  $\lambda_{(118)}(x, y) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{minus2}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
  $\theta_{\mathsf{minus2}}(x, y) = \mathsf{true}$

24 mod

The functional procedure mod computes the remainder function for naturals.

function mod : nat $\times$ nat $\rightarrow$ nat
$\mathsf{mod}(x,y) = \mathsf{if}(\mathsf{le}(y,x),\ \mathsf{if}(\mathsf{eq}(y,0),\ 0,\ \mathsf{mod}(\mathsf{minus2}(x,y),y)),\ x)$

Termination Hypothesis

$$\mathsf{le}(y,x) \wedge \neg\mathsf{eq}(y,0)\ \rightarrow\ \langle \mathsf{minus2}(x,y), y\rangle \prec \langle x,y\rangle \tag{126}$$

Transformation

1. No (direct) inductive evaluation is needed for the termination hypothesis (126) of mod. The defined symbol minus2 is eliminated by estimation.

$$\langle \overline{\mathsf{minus2}}(x,y), y\rangle \preceq \langle x,y\rangle \tag{127}$$
$$u \prec v \rightarrow \langle u,y\rangle \prec \langle v,y\rangle \tag{128}$$
$$\mathcal{E}_{\mathsf{minus2}\ \preceq\ \overline{\mathsf{minus2}}} \tag{129}$$
$$\mathsf{le}(y,x) \wedge \neg\mathsf{eq}(y,0) \rightarrow \delta_{\mathsf{minus2}\ \prec\ \overline{\mathsf{minus2}}}(x,y) \vee \langle \overline{\mathsf{minus2}}(x,y), y\rangle \prec \langle x,y\rangle \tag{130}$$

2. However, inductive evaluation has to be used to compute the estimation inequalities of minus2. The estimation inequalities $\mathcal{E}_{\mathsf{minus2}\ \preceq\ \overline{\mathsf{minus2}}}$ have to ensure $\mathsf{minus2}(x,y) \preceq \overline{\mathsf{minus2}}(x,y)$. As we already illustrated, to compute these estimation inequalities one performs a case analysis w.r.t. the algorithm minus2, i.e. one obtains the inequalities

$$\mathsf{le}(x,y) \rightarrow 0 \preceq \overline{\mathsf{minus2}}(x,y), \tag{131}$$
$$\neg\mathsf{le}(x,y) \rightarrow \mathsf{s}(\mathsf{minus2}(\mathsf{p}(x),y)) \preceq \overline{\mathsf{minus2}}(x,y). \tag{132}$$

   Now for the first inequality (131) we can apply inductive evaluation w.r.t. $\mathsf{le}(x,y)$ and eliminate the premise afterwards.

$$0 \preceq \overline{\mathsf{minus2}}(0,v) \tag{133}$$
$$\mathsf{minus2}(u,v) \preceq \overline{\mathsf{minus2}}(\mathsf{s}(u),\mathsf{s}(v)) \tag{134}$$

   In (134) we performed a simplification by removing $+0$ on both sides of the inequality.

3. Similarly, we also apply inductive evaluation w.r.t. $\mathsf{le}(x,y)$ for (132). However, for formulas in estimation inequalities which result from recursive cases (such as (132)), we do not allow an application of the induction hypothesis during the inductive evaluation, i.e. we only perform a case analysis w.r.t. le. (The reason is that otherwise one could obtain non-terminating strictness predicates afterwards. For further details see [Gie95b].) By inductive evaluation and subsequent symbolic evaluation (in the premise and in the conclusion), (132) is transformed into

$$\mathsf{true}\ \rightarrow\ \mathsf{s}(\mathsf{minus2}(u,0)) \preceq \overline{\mathsf{minus2}}(\mathsf{s}(u),0) \tag{135}$$
$$\neg\mathsf{le}(u,v)\ \rightarrow\ \mathsf{s}(\mathsf{minus2}(u,\mathsf{s}(v))) \preceq \overline{\mathsf{minus2}}(\mathsf{s}(u),\mathsf{s}(v)). \tag{136}$$

   Now we continue as usual, i.e. we eliminate the premise and apply the induction hypotheses for the estimation of minus2 which results in

$$\mathsf{s}(\overline{\mathsf{minus2}}(u,0)) \preceq \overline{\mathsf{minus2}}(\mathsf{s}(u),0) \tag{137}$$
$$\mathsf{s}(\overline{\mathsf{minus2}}(u,\mathsf{s}(v))) \preceq \overline{\mathsf{minus2}}(\mathsf{s}(u),\mathsf{s}(v)) \tag{138}$$
$$u \prec v \rightarrow \mathsf{s}(u) \prec \mathsf{s}(v). \tag{139}$$

4. In a similar way one can also use inductive evaluation to compute the strictness predicate $\delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}$. Instead of

   function $\delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
   $$\begin{aligned}\delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(x,y) = \mathsf{if}(\,&\mathsf{le}(x,y),\\ &0 \prec \overline{\mathsf{minus2}}(x,y),\\ &\delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(\mathsf{p}(x),y) \vee \delta_{\mathsf{p} \prec \overline{\mathsf{p}}}(x)\\ &\qquad\qquad \vee \mathsf{s}(\overline{\mathsf{minus2}}(\overline{\mathsf{p}}(x),y)) \prec \overline{\mathsf{minus2}}(x,y))\end{aligned}$$

   we can replace the result $0 \prec \overline{\mathsf{minus2}}(x,y)$ by the call of a new function $\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(x,y)$ which reflects the results of the inductive evaluation of $\mathsf{le}(x,y) \to 0 \prec \overline{\mathsf{minus2}}(x,y)$. In the first case, the result of $\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(0,v)$ is $0 \prec \overline{\mathsf{minus2}}(0,v)$, i.e. the strict version of (133). In the second case, the result of $\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(\mathsf{s}(u),0)$ is true, as the premise of the corresponding formula could be proved to be unsatisfiable during the inductive evaluation process. Finally, in the (third) step case, $\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(\mathsf{s}(u),\mathsf{s}(v))$ should be true iff $0 \prec \overline{\mathsf{minus2}}(\mathsf{s}(u),\mathsf{s}(v))$ is true. Note that the estimation inequalities $\mathcal{E}_{\mathsf{minus2} \preceq \overline{\mathsf{minus2}}}$ already guarantee

   $$0 \preceq \overline{\mathsf{minus2}}(u,v) \preceq \overline{\mathsf{minus2}}(\mathsf{s}(u),\mathsf{s}(v)).$$

   Hence, $\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(\mathsf{s}(u),\mathsf{s}(v))$ should return true if the first or the second inequality above is strict. By an inductive argument, the strictness of the first inequality (i.e. $0 \preceq \overline{\mathsf{minus2}}(u,v)$) is implied by $\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(u,v)$. We proceed in a similar way for the result of $\delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}$ in the 'else' case and obtain the following definitions for minus' strictness predicate.

   function $\delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
   $$\begin{aligned}\delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(x,y) = \mathsf{if}(\,&\mathsf{le}(x,y),\\ &\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(x,y),\\ &\delta''_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(x,y))\end{aligned}$$

   function $\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
   $$\begin{aligned}\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(0,v) &= 0 \prec \overline{\mathsf{minus2}}(0,v)\\ \delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(\mathsf{s}(u),0) &= \mathsf{true}\\ \delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(\mathsf{s}(u),\mathsf{s}(v)) &= \delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(u,v) \vee \overline{\mathsf{minus2}}(u,v) \prec \overline{\mathsf{minus2}}(\mathsf{s}(u),\mathsf{s}(v)).\end{aligned}$$

   function $\delta''_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
   $$\begin{aligned}\delta''_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(0,v) &= \mathsf{true}\\ \delta''_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(\mathsf{s}(u),0) &= \delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(u,0) \vee \mathsf{s}(\overline{\mathsf{minus2}}(u,0)) \prec \overline{\mathsf{minus2}}(\mathsf{s}(u),0))\\ \delta''_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(\mathsf{s}(u),\mathsf{s}(v)) &= \delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(u,\mathsf{s}(v)) \vee \mathsf{s}(\overline{\mathsf{minus2}}(u,\mathsf{s}(v))) \prec \overline{\mathsf{minus2}}(\mathsf{s}(u),\mathsf{s}(v))).\end{aligned}$$

   So inductive evaluation can also be used for the computation of estimation inequalities and strictness predicates.

5. Now we eliminate the second part of the disjunction in (130) and replace it by a set of inequalities from $\delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}$, $\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}$, and $\delta''_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}$ that is sufficient for $\mathsf{le}(y,x) \wedge \neg_{\mathsf{eq}}(y,0) \to \delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(x,y)$. For example, we can use the inequality from the third defining equation of $\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}$, i.e. we transform (132) into

   $$\overline{\mathsf{minus2}}(u,v) \prec \overline{\mathsf{minus2}}(\mathsf{s}(u),\mathsf{s}(v)). \tag{140}$$

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \overline{\mathsf{minus2}}(x,y) \mapsto x,\ \langle u,v\rangle \mapsto u$

Soundness Predicate
function $\lambda_{(126)} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(126)}(y,x) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{mod}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\theta_{\mathsf{mod}}(x, y) = \mathsf{true}$

25 gcd1

The functional procedure gcd1 from [MW78] computes the greatest common divisor of two naturals.

function $\mathsf{gcd1} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$
$\mathsf{gcd1}(0, y) = y$
$\mathsf{gcd1}(\mathsf{s}(v), y) = \mathsf{gcd1}(\mathsf{mod}(y, \mathsf{s}(v)), \mathsf{s}(v))$

Termination Hypothesis

$$\langle \mathsf{mod}(y, \mathsf{s}(v)), \mathsf{s}(v)\rangle \prec \langle \mathsf{s}(v), y\rangle \tag{141}$$

Transformation

1. No (direct) inductive evaluation is needed for the termination hypothesis (126) of gcd1. The defined symbol mod is eliminated by estimation.

$$\langle \overline{\mathsf{mod}}(y, \mathsf{s}(v)), \mathsf{s}(v)\rangle \preceq \langle \mathsf{s}(v), y\rangle \tag{142}$$
$$u \prec v \to \langle u, y\rangle \prec \langle v, y\rangle \tag{143}$$
$$\mathcal{E}_{\mathsf{mod} \preceq \overline{\mathsf{mod}}} \tag{144}$$
$$\delta_{\mathsf{mod} \prec \overline{\mathsf{mod}}}(y, \mathsf{s}(v)) \vee \langle \overline{\mathsf{mod}}(y, \mathsf{s}(v)), \mathsf{s}(v)\rangle \prec \langle \mathsf{s}(v), y\rangle \tag{145}$$

2. However, to compute the estimation inequalities $\mathcal{E}_{\mathsf{mod} \preceq \overline{\mathsf{mod}}}$ of mod we use inductive evaluation w.r.t. $\mathsf{le}(y, x)$ (on the formula $\neg\mathsf{le}(y, x) \to x \preceq \overline{\mathsf{mod}}(x, y)$). Moreover, the function symbol minus2 is eliminated by generalization. (To ease readability, we did not use inductive evaluation for the other cases during the computation of mod's estimation inequalities.) In this way, $\mathcal{E}_{\mathsf{mod} \preceq \overline{\mathsf{mod}}}$ is computed as

$$0 \preceq \overline{\mathsf{mod}}(x, y), \tag{146}$$
$$\overline{\mathsf{mod}}(z, y) \preceq \overline{\mathsf{mod}}(x, y), \tag{147}$$
$$0 \preceq \overline{\mathsf{mod}}(0, \mathsf{s}(u)) \tag{148}$$
$$\mathsf{s}(v) + \overline{\mathsf{mod}}(v, u) \preceq \overline{\mathsf{mod}}(\mathsf{s}(v), \mathsf{s}(u)) + v. \tag{149}$$

Analogously, inductive evaluation is also used to compute the strictness predicate of mod.

```
function $\delta_{\mathsf{mod} \prec \overline{\mathsf{mod}}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
  $\delta_{\mathsf{mod} \prec \overline{\mathsf{mod}}}(x, y) = \mathsf{if}(\mathsf{le}(y, x),$
                         $\mathsf{if}(\mathsf{eq}(y, 0),$
                            $0 \prec \overline{\mathsf{mod}}(x, y),$
                            $\delta_{\mathsf{mod} \prec \overline{\mathsf{mod}}}(\mathsf{minus2}(x, y), y) \vee \overline{\mathsf{mod}}(z, y) \prec \overline{\mathsf{mod}}(x, y)),$
                         $\delta'_{\mathsf{mod} \prec \overline{\mathsf{mod}}}(x, y))$
function $\delta'_{\mathsf{mod} \prec \overline{\mathsf{mod}}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
  $\delta'_{\mathsf{mod} \prec \overline{\mathsf{mod}}}(v, 0) = \mathsf{true}$
  $\delta'_{\mathsf{mod} \prec \overline{\mathsf{mod}}}(0, \mathsf{s}(u)) = 0 \prec \overline{\mathsf{mod}}(0, \mathsf{s}(u))$
  $\delta'_{\mathsf{mod} \prec \overline{\mathsf{mod}}}(\mathsf{s}(v), \mathsf{s}(u)) = \delta'_{\mathsf{mod} \prec \overline{\mathsf{mod}}}(v, u) \vee \mathsf{s}(v) + \overline{\mathsf{mod}}(v, u) \prec \overline{\mathsf{mod}}(\mathsf{s}(v), \mathsf{s}(u)) + v$
```

3. Now we eliminate the second part of the disjunction in (145) and replace it by a set of inequalities from $\delta_{\mathsf{mod} \prec \overline{\mathsf{mod}}}$ and $\delta'_{\mathsf{mod} \prec \overline{\mathsf{mod}}}$. For example, the inequality from the second defining equation of $\delta'_{\mathsf{mod} \prec \overline{\mathsf{mod}}}$ is sufficient for $\delta_{\mathsf{mod} \prec \overline{\mathsf{mod}}}(y, \mathsf{s}(v))$. Hence, (145) is transformed into

$$0 \prec \overline{\mathsf{mod}}(0, \mathsf{s}(u)). \tag{150}$$

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \overline{\mathsf{mod}}(x,y) \mapsto y,\ \langle u,v\rangle \mapsto u$

Soundness Predicate

function $\lambda_{(141)} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(141)}(y,v) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{gcd1}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\theta_{\mathsf{gcd1}}(v,y) = \mathsf{true}$

26 gcd2

The functional procedure gcd2 computes the greatest common divisor of two naturals.

function $\mathsf{gcd2} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$
$\mathsf{gcd2}(x,y) = \mathsf{if}(\mathsf{le}(y,x),\ \mathsf{if}(\mathsf{eq}(y,0),\ x,\ \mathsf{gcd2}(\mathsf{minus2}(x,y),y)),\ \mathsf{gcd2}(y,x))$

Termination Hypotheses

$$\mathsf{le}(y,x) \wedge \neg\mathsf{eq}(y,0) \ \to\ \langle \mathsf{minus2}(x,y),y\rangle \prec \langle x,y\rangle \tag{151}$$
$$\neg\mathsf{le}(y,x) \ \to\ \langle y,x\rangle \prec \langle x,y\rangle \tag{152}$$

Transformation

1. Inductive Evaluation of (152) w.r.t. $\mathsf{le}(y,x)$, Symbolic Evaluation:

$$\mathsf{true} \ \to\ \langle \mathsf{s}(u),0\rangle \prec \langle 0,\mathsf{s}(u)\rangle \tag{153}$$
$$\neg\mathsf{le}(u,v) \ \to\ \langle \mathsf{s}(u),\mathsf{s}(v)\rangle + \langle v,u\rangle \preceq \langle \mathsf{s}(v),\mathsf{s}(u)\rangle + \langle u,v\rangle. \tag{154}$$

2. Premise Elimination in (153) and (154).
3. The first termination hypothesis (151) could also be inductively evaluated w.r.t. $\mathsf{le}(y,x)$, but in the step case the induction hypothesis would not be applicable. Afterwards, the defined symbol minus2 would have to be eliminated by estimation. To ease readability, we omit this inductive evaluation of (151) and directly apply estimation.

$$\langle \overline{\mathsf{minus2}}(x,y),y\rangle \preceq \langle x,y\rangle \tag{155}$$
$$u \prec v \to \langle u,y\rangle \prec \langle v,y\rangle \tag{156}$$
$$\mathcal{E}_{\mathsf{minus2} \preceq \overline{\mathsf{minus2}}} \tag{157}$$
$$\mathsf{le}(y,x) \wedge \neg\mathsf{eq}(y,0) \to \delta_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}(x,y) \vee \langle \overline{\mathsf{minus2}}(x,y),y\rangle \prec \langle x,y\rangle. \tag{158}$$

Finally, similar to the termination proof of mod, (158) is replaced by the inequality from the third defining equation of $\delta'_{\mathsf{minus2} \prec \overline{\mathsf{minus2}}}$, i.e. we transform (158) into

$$\overline{\mathsf{minus2}}(u,v) \prec \overline{\mathsf{minus2}}(\mathsf{s}(u),\mathsf{s}(v)). \tag{159}$$

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \overline{\mathsf{minus2}}(x,y) \mapsto x,\ \langle x,y\rangle \mapsto x+2y$

Soundness Predicates

function $\lambda_{(151)} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(151)}(y,x) = \mathsf{true}$
function $\lambda_{(152)} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(152)}(y,x) = \mathsf{true}$

Termination Predicate

function θ_{gcd2} : nat × nat → bool
$\theta_{\mathsf{gcd2}}(x, y) = \mathsf{true}$

27 next_non_element

next_non_element(x, y) computes the least element greater than x which does *not* occur in the list y.

function next_non_element : nat × list → nat
next_non_element(x, y) = if(mem(x, y), next_non_element(s(x), y), x)

Termination Hypothesis

$$\mathsf{mem}(x, y) \;\rightarrow\; \langle \mathsf{s}(x), y\rangle\rangle \prec \langle x, y\rangle \tag{160}$$

Transformation

1. Inductive Evaluation w.r.t. mem(x, y), Symbolic Evaluation:

$$\mathsf{true} \wedge \mathsf{eq}(n, m) \;\rightarrow\; \langle \mathsf{s}(n), \mathsf{add}(m, v)\rangle \prec \langle n, \mathsf{add}(m, v)\rangle \tag{161}$$

$$\begin{aligned} \mathsf{mem}(n, v) \wedge \neg\mathsf{eq}(n, m) \;\rightarrow\; & \langle \mathsf{s}(n), \mathsf{add}(m, v)\rangle + \langle n, v\rangle \preceq \\ & \langle n, \mathsf{add}(m, v)\rangle + \langle \mathsf{s}(n), v\rangle \end{aligned} \tag{162}$$

2. In (161), we use a rule *Replace* that eliminates the "equation" eq(n, m) and substitutes n at each occurrence by m. As a heuristic, we apply Replace only w.r.t. an "equation" eq(x, t) or eq(t, x) if x is a variable and t is a constructor term.

$$\mathsf{true} \;\rightarrow\; \langle \mathsf{s}(m), \mathsf{add}(m, v)\rangle \prec \langle m, \mathsf{add}(m, v)\rangle \tag{163}$$

   In (162), Inductive Evaluation w.r.t. eq(n, m) is suggested by our heuristic, but does not result in a "better" termination predicate.

3. Premise Elimination in (162) and (163)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(m) \mapsto m + 1,\ \mathsf{empty} \mapsto 0,\ \mathsf{add}(m, v) \mapsto m + v + 1,\ \langle x, y\rangle \mapsto (x - y)^2$

Soundness Predicate

function $\lambda_{(160)}$: nat × list → bool
$\lambda_{(160)}(x, y) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{next\_non\_element}}$: nat × list → bool
$\theta_{\mathsf{next\_non\_element}}(x, y) = \mathsf{true}$

28 minus3

minus3(x, y) returns the difference of x and y iff $x \geq y$ and does not terminate otherwise.

function minus3 : nat × nat → nat
minus3(x, y) = if(eq(x, y), 0, s(minus3$(x$, s(y))))

Termination Hypothesis

$$\neg\mathsf{eq}(x, y) \rightarrow \langle x, \mathsf{s}(y)\rangle \prec \langle x, y\rangle \tag{164}$$

Transformation

1. Inductive Evaluation w.r.t. $\mathsf{eq}(x,y)$, Symbolic Evaluation:

$$\mathsf{true} \rightarrow \langle 0, \mathsf{s}(\mathsf{s}(v))\rangle \prec \langle 0, \mathsf{s}(v)\rangle \tag{165}$$

$$\mathsf{true} \rightarrow \langle \mathsf{s}(u), \mathsf{s}(0)\rangle \prec \langle \mathsf{s}(u), 0\rangle \tag{166}$$

$$\neg\mathsf{eq}(u,v) \rightarrow \langle \mathsf{s}(u), \mathsf{s}(\mathsf{s}(v))\rangle + \langle u,v\rangle \preceq \langle \mathsf{s}(u), \mathsf{s}(v)\rangle + \langle u, \mathsf{s}(v)\rangle \tag{167}$$

   Note that in the functional procedure of eq the variables x and y are used to define the patterns. As these variables already occur in the termination hypothesis of minus3, we rename the variables of eq's procedure by u and v. In some of the following examples we also use a variable renaming to ease readability.
2. Premise Elimination in (166) and (167), Rejection in (165)

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle x,y\rangle \mapsto (x-y)^2$

Soundness Predicate

function $\lambda_{(164)}$: nat × nat → bool

$\lambda_{(164)}(0,0) = \mathsf{true}$
$\lambda_{(164)}(0,\mathsf{s}(v)) = \mathsf{false}$
$\lambda_{(164)}(\mathsf{s}(u),0) = \mathsf{true}$
$\lambda_{(164)}(\mathsf{s}(u),\mathsf{s}(v)) = \lambda_{(164)}(u,v)$

$\lambda_{(164)}(x,y) = \mathsf{true}$ iff $x \geq y$

Termination Predicate

function θ_{minus3} : nat × nat → bool

$\theta_{\mathsf{minus3}}(x,y) = \mathsf{if}(\mathsf{eq}(x,y),\ \mathsf{true},\ \lambda_{(164)}(x,y) \wedge \theta_{\mathsf{minus3}}(x,\mathsf{s}(y)))$[11]

$\theta_{\mathsf{minus3}}(x,y) = \mathsf{true}$ iff $x \geq y$.

29 nth*

The functional procedure nth computes the element access function for lists where the first element is associated with index 0.

function nth : nat × list → nat

$\mathsf{nth}(x,\mathsf{empty}) = 0$
$\mathsf{nth}(0,\mathsf{add}(v,w)) = v$
$\mathsf{nth}(\mathsf{s}(u),\mathsf{add}(v,w)) = \mathsf{nth}(u,w)$

Termination Hypothesis

$$\langle u,w\rangle \prec \langle \mathsf{s}(u), \mathsf{add}(v,w)\rangle \tag{168}$$

Transformation by Premise Elimination

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(u) \mapsto u+1,\ \langle u,w\rangle \mapsto u$

Soundness Predicate

function $\lambda_{(168)}$: nat × list × nat → bool

$\lambda_{(168)}(u,w,v) = \mathsf{true}$

[11] We abbreviate conditionals of the form $\mathsf{if}(b,\ t,\ \mathsf{false})$ with $b \wedge t$

Termination Predicate

function θ_{nth} : nat × list → bool
$\theta_{\mathsf{nth}}(x, y) = \mathsf{true}$

30 nextindex

$\mathsf{nextindex}(i, b, z)$ computes the smallest index $j \geq i$ such that z is the j-th element of the list b. For instance, $\mathsf{nextindex}(2, [5, 1, 3, 5, 7, 5], 5) = 3$.[12] The procedure terminates iff z occurs in b at a position whose index is greater than or equal to i or $z = 0$ (since $\mathsf{nth}(i, b) = 0$ whenever i is not an index of b).

function nextindex : nat × list × nat → nat
$\mathsf{nextindex}(i, b, z) = \mathsf{if}(\mathsf{eq}(\mathsf{nth}(i, b), z),\ i,\ \mathsf{nextindex}(\mathsf{s}(i), b, z))$

Termination Hypothesis

$$\neg\mathsf{eq}(\mathsf{nth}(i, b), z) \;\rightarrow\; \langle \mathsf{s}(i), b, z\rangle \prec \langle i, b, z\rangle \tag{169}$$

Transformation

1. Inductive Evaluation w.r.t. $\mathsf{nth}(i, b)$, Symbolic Evaluation:

$$\begin{aligned}
\neg\mathsf{eq}(0, z) &\;\rightarrow\; \langle \mathsf{s}(x), \mathsf{empty}, z\rangle \prec \langle x, \mathsf{empty}, z\rangle && (170)\\
\neg\mathsf{eq}(v, z) &\;\rightarrow\; \langle \mathsf{s}(0), \mathsf{add}(v, w), z\rangle \prec \langle 0, \mathsf{add}(v, w), z\rangle && (171)\\
\neg\mathsf{eq}(\mathsf{nth}(u, w), z) &\;\rightarrow\; \langle \mathsf{s}(\mathsf{s}(u)), \mathsf{add}(v, w), z\rangle + \langle u, w, z\rangle \preceq \\
&\qquad \langle \mathsf{s}(u), \mathsf{add}(v, w), z\rangle + \langle \mathsf{s}(u), w, z\rangle && (172)
\end{aligned}$$

2. Premise Elimination in (171) and (172), Rejection in (170)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \mathsf{empty} \mapsto 0,\ \mathsf{add}(v, w) \mapsto w + 1,\ \langle i, b, z\rangle \mapsto (b - i)^2$

Soundness Predicate

function $\lambda_{(169)}$: nat × list × nat → bool

$$\begin{aligned}
\lambda_{(169)}(x, \mathsf{empty}, z) &= \mathsf{false}\\
\lambda_{(169)}(0, \mathsf{add}(v, w), z) &= \mathsf{true}\\
\lambda_{(169)}(\mathsf{s}(u), \mathsf{add}(v, w), z) &= \lambda_{(169)}(u, w, z)
\end{aligned}$$

$\lambda_{(169)}(i, b, z) = \mathsf{true}$ iff $i < \mathsf{len}(b)$

Termination Predicate

function $\theta_{\mathsf{nextindex}}$: nat × list × nat → bool
$\theta_{\mathsf{nextindex}}(i, b, z) = \mathsf{if}(\mathsf{eq}(\mathsf{nth}(i, b), z),\ \mathsf{true},\ \lambda_{(169)}(i, b, z) \wedge \theta_{\mathsf{nextindex}}(\mathsf{s}(i), b, z))$

$\theta_{\mathsf{nextindex}}(i, b, z) = \mathsf{true}$ iff z occurs in b at a position whose index is greater than or equal to i or $z = 0$.

31 add_if_member*

$\mathsf{add\_if\_member}(x, y, z)$ returns $\mathsf{add}(x, z)$ if x occurs in y and returns z otherwise.

function add_if_member : nat × list × list → list

$$\begin{aligned}
\mathsf{add\_if\_member}(x, \mathsf{empty}, z) &= z\\
\mathsf{add\_if\_member}(x, \mathsf{add}(u, y), z) &= \mathsf{if}(\mathsf{eq}(x, u),\ \mathsf{add}(x, z),\ \mathsf{add\_if\_member}(x, y, z))
\end{aligned}$$

[12] Here we abbreviate objects of data type nat by the naturals they denote and objects of data type list by [...].

Termination Hypothesis

$$\neg \mathsf{eq}(x,u) \;\rightarrow\; \langle x,y,z\rangle \prec \langle x,\mathsf{add}(u,y),z\rangle \tag{173}$$

Transformation by Premise Elimination (Inductive Evaluation is not needed.)

Polynomial Ordering
$\mathsf{empty} \mapsto 0,\ \mathsf{add}(u,y) \mapsto y+1,\ \langle x,y,z\rangle \mapsto y$

Soundness Predicate
function $\lambda_{(173)}$: $\mathsf{nat} \times \mathsf{nat} \times \mathsf{list} \times \mathsf{list} \rightarrow \mathsf{bool}$
$\lambda_{(173)}(x,u,y,z) = \mathsf{true}$

Termination Predicate
function $\theta_{\mathsf{add\_if\_member}}$: $\mathsf{nat} \times \mathsf{list} \times \mathsf{list} \rightarrow \mathsf{bool}$
$\theta_{\mathsf{add\_if\_member}}(x,y,z) = \mathsf{true}$

32 sort

$\mathsf{sort}(x,y)$ returns the sorted list of all elements of list y which are greater than or equal to x where duplicates are removed. The functional procedure $\mathsf{sort}(x,y)$ terminates iff y is empty and x is 0 or y is not empty and x is less than or equal to the maximal element of y. Hence $\mathsf{sort}(0,y)$ sorts y.

function sort : $\mathsf{nat} \times \mathsf{list} \rightarrow \mathsf{list}$
$\mathsf{sort}(x,y) = \mathsf{if}(\mathsf{eq}(x,\mathsf{max}(y)),\ \mathsf{add}(x,\mathsf{empty}),\ \mathsf{add\_if\_member}(x,y,\mathsf{sort}(\mathsf{s}(x),y)))$

Termination Hypothesis

$$\neg \mathsf{eq}(x,\mathsf{max}(y)) \;\rightarrow\; \langle \mathsf{s}(x),y\rangle \prec \langle x,y\rangle \tag{174}$$

Transformation

1. Inductive Evaluation w.r.t. $\mathsf{max}(y)$, Symbolic Evaluation:

$$\begin{aligned}
\neg \mathsf{eq}(x,0) &\;\rightarrow\; \langle \mathsf{s}(x),\mathsf{empty}\rangle \prec \langle x,\mathsf{empty}\rangle && (175)\\
\neg \mathsf{eq}(x,u) &\;\rightarrow\; \langle \mathsf{s}(x),\mathsf{add}(u,\mathsf{empty})\rangle \prec \langle x,\mathsf{add}(u,\mathsf{empty})\rangle && (176)\\
\neg \mathsf{eq}(x,\mathsf{max}(\mathsf{add}(v,w))) \wedge \mathsf{le}(u,v) &\;\rightarrow\; \langle \mathsf{s}(x),\mathsf{add}(u,\mathsf{add}(v,w))\rangle + \langle x,\mathsf{add}(v,w)\rangle \preceq \\
&\qquad \langle x,\mathsf{add}(u,\mathsf{add}(v,w))\rangle + \langle \mathsf{s}(x),\mathsf{add}(v,w)\rangle && (177)\\
\neg \mathsf{eq}(x,\mathsf{max}(\mathsf{add}(u,w))) \wedge \neg \mathsf{le}(u,v) &\;\rightarrow\; \langle \mathsf{s}(x),\mathsf{add}(u,\mathsf{add}(v,w))\rangle + \langle x,\mathsf{add}(u,w)\rangle \preceq \\
&\qquad \langle x,\mathsf{add}(u,\mathsf{add}(v,w))\rangle + \langle \mathsf{s}(x),\mathsf{add}(u,w)\rangle && (178)
\end{aligned}$$

2. In (176), Inductive Evaluation w.r.t. $\mathsf{eq}(x,u)$, Symbolic Evaluation (Inductive Evaluation in (177) and (178) is not needed.):

$$\begin{aligned}
\mathsf{true} &\;\rightarrow\; \langle \mathsf{s}(0),\mathsf{add}(\mathsf{s}(w),\mathsf{empty})\rangle \prec \langle 0,\mathsf{add}(\mathsf{s}(w),\mathsf{empty})\rangle && (179)\\
\mathsf{true} &\;\rightarrow\; \langle \mathsf{s}(\mathsf{s}(v)),\mathsf{add}(0,\mathsf{empty})\rangle \prec \langle \mathsf{s}(v),\mathsf{add}(0,\mathsf{empty})\rangle && (180)\\
\neg \mathsf{eq}(v,w) &\;\rightarrow\; \langle \mathsf{s}(\mathsf{s}(v)),\mathsf{add}(\mathsf{s}(w),\mathsf{empty})\rangle + \langle v,\mathsf{add}(w,\mathsf{empty})\rangle \preceq \\
&\qquad \langle \mathsf{s}(v),\mathsf{add}(\mathsf{s}(w),\mathsf{empty})\rangle + \langle \mathsf{s}(v),\mathsf{add}(w,\mathsf{empty})\rangle && (181)
\end{aligned}$$

3. Premise Elimination in (177), (178), (179), and (181), Rejection in (175) and (180)

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \mathsf{empty} \mapsto 0, \mathsf{add}(v,w) \mapsto v+w, \langle x,y\rangle \mapsto (y-x)^2$

Soundness Predicates

function $\lambda_{(176)}$: $\mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$$\begin{aligned}
\lambda_{(176)}(0,0) &= \mathsf{true}\\
\lambda_{(176)}(0,\mathsf{s}(w)) &= \mathsf{true}\\
\lambda_{(176)}(\mathsf{s}(v),0) &= \mathsf{false}\\
\lambda_{(176)}(\mathsf{s}(v),\mathsf{s}(w)) &= \lambda_{(176)}(v,w)
\end{aligned}$$

$\lambda_{(176)}(x,u) = \mathsf{true}$ iff $x \le u$

function $\lambda_{(174)}$: $\mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$

$$\begin{aligned}
\lambda_{(174)}(x,\mathsf{empty}) &= \mathsf{false}\\
\lambda_{(174)}(x,\mathsf{add}(u,\mathsf{empty})) &= \lambda_{(176)}(x,u)\\
\lambda_{(174)}(x,\mathsf{add}(u,\mathsf{add}(v,w))) &= \mathsf{if}(\mathsf{le}(u,v),\ \lambda_{(174)}(x,\mathsf{add}(v,w)),\ \lambda_{(174)}(x,\mathsf{add}(u,w)))
\end{aligned}$$

$\lambda_{(174)}(x,y) = \mathsf{true}$ iff $y \neq \mathsf{empty} \wedge x \le \mathsf{max}(y)$

Termination Predicate

function θ_{sort} : $\mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$

$$\theta_{\mathsf{sort}}(x,y) = \mathsf{if}(\mathsf{eq}(x,\mathsf{max}(y)),\ \mathsf{true},\ \lambda_{(174)}(x,y) \wedge \theta_{\mathsf{sort}}(\mathsf{s}(x),y))$$

$\theta_{\mathsf{sort}}(x,y) = \mathsf{true}$ iff $x \le \mathsf{max}(y)$.

33 half_diff

half_diff(x,y) computes $\frac{x-y}{2}$ iff $x \ge y$ and the difference of x and y is even. Otherwise, the functional procedure does not terminate.

function half_diff : $\mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$

$$\mathsf{half\_diff}(x,y) = \mathsf{if}(\mathsf{eq}(x,y),\ 0,\ \mathsf{s}(\mathsf{half\_diff}(x,\mathsf{s}(\mathsf{s}(y)))))$$

Termination Hypothesis

$$\neg\mathsf{eq}(x,y) \ \to\ \langle x,\mathsf{s}(\mathsf{s}(y))\rangle \prec \langle x,y\rangle \tag{182}$$

Transformation

1. Inductive Evaluation w.r.t. $\mathsf{eq}(x,y)$, Symbolic Evaluation:

$$\mathsf{true} \ \to\ \langle 0,\mathsf{s}(\mathsf{s}(\mathsf{s}(v)))\rangle \prec \langle 0,\mathsf{s}(v)\rangle \tag{183}$$
$$\mathsf{true} \ \to\ \langle \mathsf{s}(u),\mathsf{s}(\mathsf{s}(0))\rangle \prec \langle \mathsf{s}(u),0\rangle \tag{184}$$
$$\neg\mathsf{eq}(u,v) \ \to\ \langle \mathsf{s}(u),\mathsf{s}(\mathsf{s}(\mathsf{s}(v)))\rangle + \langle u,v\rangle \preceq \langle \mathsf{s}(u),\mathsf{s}(v)\rangle + \langle u,\mathsf{s}(\mathsf{s}(v))\rangle \tag{185}$$

2. Premise Elimination in (184) and (185), Rejection in (183)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle x,y\rangle \mapsto (x-y+1)^2$

Soundness Predicate

function $\lambda_{(182)}$: $\mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$$\begin{aligned}
\lambda_{(182)}(0,0) &= \mathsf{true}\\
\lambda_{(182)}(0,\mathsf{s}(v)) &= \mathsf{false}\\
\lambda_{(182)}(\mathsf{s}(u),0) &= \mathsf{true}\\
\lambda_{(182)}(\mathsf{s}(u),\mathsf{s}(v)) &= \lambda_{(182)}(u,v)
\end{aligned}$$

$\lambda_{(182)}(x,y) = \mathsf{true}$ iff $x \ge y$

Termination Predicate

function $\theta_{\mathsf{half\_diff}}$: nat × nat → bool

$\theta_{\mathsf{half\_diff}}(x, y) = \mathsf{if}(\mathsf{eq}(x, y),\ \mathsf{true},\ \lambda_{(182)}(x, y) \wedge \theta_{\mathsf{half\_diff}}(x, \mathsf{s}(\mathsf{s}(y))))$

$\theta_{\mathsf{half\_diff}}(x, y) = \mathsf{true}$ iff $x \geq y \wedge \mathsf{even}(x - y)$.

34 set*

$\mathsf{set}(x, l, y)$ replaces in list l the element with index x by y.

function set : nat × list × nat → list

$\mathsf{set}(x, \mathsf{empty}, y) = \mathsf{empty}$
$\mathsf{set}(0, \mathsf{add}(v, w), y) = \mathsf{add}(y, w)$
$\mathsf{set}(\mathsf{s}(u), \mathsf{add}(v, w), y) = \mathsf{add}(v, \mathsf{set}(u, w, y))$

Termination Hypothesis

$$\langle u, w, y\rangle \prec \langle \mathsf{s}(u), \mathsf{add}(v, w), y\rangle \tag{186}$$

Transformation by Premise Elimination

Polynomial Ordering

$\mathsf{empty} \mapsto 0,\ \mathsf{add}(v, w) \mapsto w + 1,\ \langle u, w, y\rangle \mapsto w$

Soundness Predicate

function $\lambda_{(186)}$: nat × nat × list × nat → bool

$\lambda_{(186)}(u, v, w, y) = \mathsf{true}$

Termination Predicate

function θ_{set} : nat × list × nat → bool

$\theta_{\mathsf{set}}(x, l, y) = \mathsf{true}$

35 lnth*

Consider the data type llist ("list of lists") with the constructors lempty :→ llist and $\mathsf{ladd}(v, w)$: list × llist → llist.

$\mathsf{lnth}(n, m, x)$ returns the m-th natural of the n-th list in the list of lists x where the first elements are associated with index 0.

function lnth : nat × nat × llist → nat

$\mathsf{lnth}(n, m, \mathsf{lempty}) = 0$
$\mathsf{lnth}(0, m, \mathsf{ladd}(v, w)) = \mathsf{nth}(m, v)$
$\mathsf{lnth}(\mathsf{s}(u), m, \mathsf{ladd}(v, w)) = \mathsf{lnth}(u, m, w)$

Termination Hypothesis

$$\langle u, m, w\rangle \prec \langle \mathsf{s}(u), m, \mathsf{ladd}(v, w)\rangle \tag{187}$$

Transformation by Premise Elimination

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(u) \mapsto u + 1,\ \langle u, m, w\rangle \mapsto u$

Soundness Predicate

function $\lambda_{(187)}$: nat × nat × list × llist → bool

$\lambda_{(187)}(u, m, v, w) = \mathsf{true}$

Termination Predicate

function θ_{lnth} : nat × nat × llist → bool

$\theta_{\mathsf{lnth}}(n, m, y) = \mathsf{true}$

36 half*

The following functional procedure halves a natural.

function half : nat $\rightarrow$ nat
  half(0) $= 0$
  half(s(0)) $= 0$
  half(s(s(v))) $=$ s(half(v))

Termination Hypothesis

$$v \prec \mathsf{s}(\mathsf{s}(v)) \tag{188}$$

Transformation by Premise Elimination

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1$

Soundness Predicate
function $\lambda_{(188)}$: nat $\rightarrow$ bool
  $\lambda_{(188)}(v) =$ true

Termination Predicate
function θ_{half} : nat $\rightarrow$ bool
  $\theta_{\mathsf{half}}(x) =$ true

37 times*

The functional procedure times multiplies two naturals.

function times : nat $\times$ nat $\rightarrow$ nat
  times(0, y) $= 0$
  times(s(u), y) $=$ plus(y, times(u, y))

Termination Hypothesis

$$\langle u, y\rangle \prec \langle \mathsf{s}(u), y\rangle \tag{189}$$

Transformation by Premise Elimination

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(u) \mapsto u + 1,\ \langle x, y\rangle \mapsto x$

Soundness Predicate
function $\lambda_{(189)}$: nat $\times$ nat $\rightarrow$ bool
  $\lambda_{(189)}(u, y) =$ true

Termination Predicate
function θ_{times} : nat $\times$ nat $\rightarrow$ bool
  $\theta_{\mathsf{times}}(x, y) =$ true

B Imperative Procedures

In this section we also regard imperative procedures for termination analysis. As an example consider the following procedure that stores in the variable x the sum of the elements of an n-element list b.

```
**procedure** sum($i, x, n$: nat, $b$: list)
  $i := 0$;  $x := 0$;
  **while** $i \neq n$ **do** $x := x + b[i]$;  $i := i + 1$ **od**
```

In our programming language, $i+1$ and $i-1$ abbreviate $\mathsf{s}(i)$ and $\mathsf{p}(i)$, respectively. Furthermore, we use the symbols $+, -, *, =, \neq, \leq, <$ for the (composed) functions plus, minus, times, eq, ¬eq, le, and lt. As special cases, we write $2 * t$, $t/2$, and t^2 instead of $\mathsf{dbl}(t)$,$\mathsf{half}(t)$, and $\mathsf{sqr}(t)$, respectively. For lists, we use the usual notation $b[i]$ for the i-th element of list (or *array*) b. Hence, $b[i]$ is an abbreviation for $\mathsf{nth}(i, b)$ if it does not occur on the left-hand side of an assignment. Each statement of the form $b[i] := t$ translates to the term $\mathsf{set}(i, b, t)$. In the same way terms of the form $b[i, j]$ (data type llist) are treated.

A straightforward approach to prove termination of imperative procedures is to transform them into functional ones. In this way every **while**-loop is translated into a separate 'loop'-function. If termination of the resulting functions can be proved, then termination of the original imperative procedure is verified.

For instance, for the **while**-loop occurring in the procedure sum we obtain the following functional procedure loop which takes the input values of the variables i, x, n, and b as arguments and returns the output value of x.[13] If the loop condition is satisfied, i.e. if $\mathsf{eq}(i, n)$ is false, then loop is called recursively with the new values of i, x, n, and b. Otherwise, x is returned.

```
**function** loop : nat × nat × nat × list → nat
  $\mathsf{loop}(i, x, n, b) = \mathsf{if}(\neg\mathsf{eq}(i, n),\ \mathsf{loop}(\mathsf{s}(i), \mathsf{plus}(x, \mathsf{nth}(b, i)), n, b),\ x)$
```

Using the functional procedure loop, the whole imperative procedure is translated into the function main.

```
**function** main : nat × list → nat
  $\mathsf{main}(n, b) = \mathsf{loop}(0, 0, n, b)$
```

Obviously, the resulting function main is equivalent to the original imperative procedure and in particular, termination of the function main is equivalent to termination of the imperative procedure sum. For a formal definition and an automation of this translation see e.g. [Hen80].

Note that in general the auxiliary functions resulting from such a translation are *partial* even if the original imperative procedure terminates totally. The reason is that in imperative procedures, termination of **while**-loops often depends on their contexts, i.e. on the preconditions that hold before entering the loop.

For instance, in our example the **while**-loop is entered if i and x are 0. However, this restriction on the values of i and s is not present in the function loop. Therefore main terminates totally, but its auxiliary function loop terminates only partially, namely if and only if i is less than or equal to n. Hence, to prove total termination of main one needs a method for termination analysis of partial functions.

The rest of this section contains 21 imperative procedures from [Gri81] whose termination is analyzed using inductive evaluation. For each procedure we describe its semantics. Then we mention the functional procedures resulting from its translation and continue termination analysis like in the previous section.

38 fibonacci ([Gri81] p. 147)

For $n \geq 1$, the imperative procedure fibonacci computes in a the n-th Fibonacci number f_n where $f_0 = 0$, $f_1 = 1$, and $f_n = f_{n-1} + f_{n-2}$ for $n > 1$.

[13] Of course, a similar function returning the output value of i, n, or b could also be constructed if required.

```
procedure fibonacci($n, i, a, b, h$:nat)
  $i := 1$;  $a := 1$;  $b := 0$;
  while $i < n$ do $i := i + 1$;  $h := a$;  $a := a + b$;  $b := h$ od
```

Translation

function loop : $\mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$
$\mathsf{loop}(n, i, a, b, h) = \mathsf{if}(\mathsf{lt}(i, n),\ \mathsf{loop}(n, \mathsf{s}(i), \mathsf{plus}(a, b), a, a),\ a)$

function main : $\mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$
$\mathsf{main}(n, h) = \mathsf{loop}(n, \mathsf{s}(0), \mathsf{s}(0), 0, h)$

Termination Hypothesis of loop

$$\mathsf{lt}(i, n) \to \langle n, \mathsf{s}(i), \mathsf{plus}(a, b), a, a\rangle \prec \langle n, i, a, b, h\rangle$$

Transformation

1. Generalization $\{\mathsf{plus}(a, b)/y\}$.
   As a heuristic, we apply the rule Generalization also at the beginning of a transformation replacing the recursive arguments of those variables which do not occur in the original **while**-condition. For instance, as the variable a does not occur in $\mathsf{lt}(i, n)$ we replace in (190) a's recursive argument $\mathsf{plus}(a, b)$ by a new variable y.
   $$\mathsf{lt}(i, n) \to \langle n, \mathsf{s}(i), y, a, a\rangle \prec \langle n, i, a, b, h\rangle \tag{190}$$
2. Inductive Evaluation w.r.t. $\mathsf{lt}(i, n)$, Symbolic Evaluation:
   $$\mathsf{true} \to \langle \mathsf{s}(v), \mathsf{s}(0), y, a, a\rangle \prec \langle \mathsf{s}(v), 0, a, b, h\rangle \tag{191}$$
   $$\begin{aligned}\mathsf{lt}(u, v) \to\ & \langle \mathsf{s}(v), \mathsf{s}(\mathsf{s}(u)), y, a, a\rangle + \langle v, u, a, b, h\rangle \preceq \\ & \langle \mathsf{s}(v), \mathsf{s}(u), a, b, h\rangle + \langle v, \mathsf{s}(u), y, a, a\rangle\end{aligned} \tag{192}$$
3. Premise Elimination in (191) and (192).

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \langle n, i, a, b, h\rangle \mapsto (n - i)^2$

Soundness Predicate

function $\lambda_{(190)}$: $\mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(190)}(n, i, a, b, h, y) = \mathsf{true}$

Termination Predicates

function θ_{loop} : $\mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\theta_{\mathsf{loop}}(n, i, a, b, h) = \mathsf{true}$

function θ_{main} : $\mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\theta_{\mathsf{main}}(n, h) = \mathsf{true}$

39 list_max ([Gri81] p. 148)

The imperative procedure list_max computes index k such that $b[k]$ is the maximum value of the n-element list b.

```
procedure list_max($n, i, k$ : nat, $b$ : list)
  $i := 1$;  $k := 0$;
  while $i < n$ do
     if $b[i] \le b[k]$ then skip else $k := i$ fi;
     $i := i + 1$
  od
```

Translation

```
function loop : nat × nat × nat × list → nat
  loop(n, i, k, b) = if( lt(i, n),
                         if( le(nth(i, b), nth(k, b)),
                            loop(n, s(i), k, b),
                            loop(n, s(i), i, b)),
                         k)

function main : nat × list → nat
  main(n, b) = loop(n, s(0), 0, b)
```

Termination Hypotheses

$$\mathsf{lt}(i,n) \wedge \ \mathsf{le}(\mathsf{nth}(i,b),\mathsf{nth}(k,b)) \ \rightarrow\ \langle n,\mathsf{s}(i),k,b\rangle \prec \langle n,i,k,b\rangle \tag{193}$$
$$\mathsf{lt}(i,n) \wedge \neg\mathsf{le}(\mathsf{nth}(i,b),\mathsf{nth}(k,b)) \ \rightarrow\ \langle n,\mathsf{s}(i),i,b)\rangle \prec \langle n,i,k,b\rangle \tag{194}$$

Transformation

1. We also use a rule *Inverse Weakening* to eliminate a conjunct from the premise of a termination hypotheses, cf. [Wal94a]. This rule is often needed before Inductive Evaluation to enable the application of the induction hypothesis. If $\psi_1 \wedge \psi_2 \rightarrow \omega$ is transformed into $\psi_1 \rightarrow \omega$ by Inverse Weakening and $\lambda_{\psi_1 \rightarrow \omega}$ is a soundness predicate for $\psi_1 \rightarrow \omega$, then it is also a soundness predicate for $\psi_1 \wedge \psi_2 \rightarrow \omega$. That is, the truth of $\lambda_{\psi_1 \rightarrow \omega}$ also implies the original hypothesis $\psi_1 \wedge \psi_2 \rightarrow \omega$. As a heuristic, we apply Inverse Weakening to eliminate those conjuncts which prevent the application of the induction hypotheses where we prefer Inductive Evaluation w.r.t. a term originally occurring in a **while**-condition. For instance, in (193) Inverse Weakening is applied to eliminate the conjunct $\mathsf{le}(\mathsf{nth}(i,b),\mathsf{nth}(k,b))$ thus preserving the **while**-condition $\mathsf{lt}(i,n)$.

$$\mathsf{lt}(i,n) \ \rightarrow\ \langle n,\mathsf{s}(i),k,b\rangle \prec \langle n,i,k,b\rangle \tag{195}$$

2. In (195), Inductive Evaluation w.r.t. $\mathsf{lt}(i,n)$, Symbolic Evaluation:

$$\mathsf{true} \ \rightarrow\ \langle \mathsf{s}(v),\mathsf{s}(0),k,b\rangle \prec \langle \mathsf{s}(v),0,k,b\rangle \tag{196}$$
$$\mathsf{lt}(u,v) \ \rightarrow\ \langle \mathsf{s}(v),\mathsf{s}(\mathsf{s}(u)),k,b\rangle + \langle v,u,k,b\rangle \preceq \langle \mathsf{s}(v),\mathsf{s}(u),k,b\rangle + \langle v,\mathsf{s}(u),k,b\rangle \tag{197}$$

3. In (194), Inverse Weakening eliminating $\neg\mathsf{le}(\mathsf{nth}(i,b),\mathsf{nth}(k,b))$:

$$\mathsf{lt}(i,n) \ \rightarrow\ \langle n,\mathsf{s}(i),i,b)\rangle \prec \langle n,i,k,b\rangle \tag{198}$$

4. In (198) Inductive Evaluation w.r.t. $\mathsf{lt}(i,n)$, Symbolic Evaluation:

$$\mathsf{true} \ \rightarrow\ \langle \mathsf{s}(v),\mathsf{s}(0),i,b\rangle \prec \langle \mathsf{s}(v),0,k,b\rangle \tag{199}$$
$$\mathsf{lt}(u,v) \ \rightarrow\ \langle \mathsf{s}(v),\mathsf{s}(\mathsf{s}(u)),i,b\rangle + \langle v,u,k,b\rangle \preceq \langle \mathsf{s}(v),\mathsf{s}(u),k,b\rangle + \langle v,\mathsf{s}(u),i,b\rangle \tag{200}$$

5. Premise Elimination in (196), (197),(199), and (200)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle n,i,k,b\rangle \mapsto (n-i)^2$

Soundness Predicates

function $\lambda_{(195)}$: nat × nat × nat × list → bool
  $\lambda_{(195)}(i,n,k,b) = \mathsf{true}$

function $\lambda_{(198)}$: nat × nat × nat × list → bool
  $\lambda_{(198)}(i,n,k,b) = \mathsf{true}$

Termination Predicates

function θ_{loop} : nat × nat × nat × list → bool
$\theta_{\mathsf{loop}}(n, i, k, b) = \mathsf{true}$

function θ_{main} : nat × list → bool
$\theta_{\mathsf{main}}(n, b) = \mathsf{true}$

40 search ([Gri81] p. 151)

Given a natural z and a list b of length n, where $z \in b$, the imperative procedure search determines the index i of z in b, thus establishing $z = b[i]$. It terminates iff $z \in b$ or $z = 0$.

```
procedure search($i$: nat, $b$: list, $z$: nat)
  $i := 0$;
  while $b[i] \neq z$ do $i := i + 1$ od
```

Translation

function nextindex : nat × list × nat → nat
$\mathsf{nextindex}(i, b, z) = \mathsf{if}(\mathsf{eq}(\mathsf{nth}(i, b), z),\ i,\ \mathsf{nextindex}(\mathsf{s}(i), b, z))$

function main : list × nat → nat
$\mathsf{main}(b, z) = \mathsf{nextindex}(0, b, z)$

Termination Analysis of nextindex cf. Example 30.

Termination Predicates

function $\theta_{\mathsf{nextindex}}$: nat × list × nat → bool
$\theta_{\mathsf{nextindex}}(i, b, z) = \mathsf{if}(\mathsf{eq}(\mathsf{nth}(i, b), z),\ \mathsf{true},\ \lambda_{(169)}(i, b, z) \wedge \mathsf{nextindex}(\mathsf{s}(i), b, z))$

$\theta_{\mathsf{nextindex}}(i, b, z) = \mathsf{true}$ iff z occurs in b at a position whose index is greater than or equal to i or $z = 0$.

function θ_{main} : list × nat → bool
$\theta_{\mathsf{main}}(b, z) = \theta_{\mathsf{nextindex}}(0, b, z)$

$\theta_{\mathsf{main}}(b, z) = \mathsf{true}$ iff $z \in b$ or $z = 0$.

41 insert ([Gri81] p. 162)

The following imperative procedure inserts z in the list b and stores in p the least index of z. It terminates iff $k < \mathsf{len}(b)$.

```
procedure insert($z, k, p$: nat, $b$: list)
  $p := 0$;  $b[k] := z$;
  while $z \neq b[p]$ do $p := p + 1$ od
```

Translation

function nextindex : nat × list × nat → nat
$\mathsf{nextindex}(i, b, z) = \mathsf{if}(\mathsf{eq}(\mathsf{nth}(i, b), z),\ i,\ \mathsf{nextindex}(\mathsf{s}(i), b, z))$

function main : nat × nat × list → nat
$\mathsf{main}(z, k, b) = \mathsf{nextindex}(0, \mathsf{set}(k, b, z), z)$

Termination Analysis of nextindex cf. Example 30.

Termination Predicates

function $\theta_{\mathsf{nextindex}}$: nat × list × nat → bool
$\theta_{\mathsf{nextindex}}(i, b, z) = \mathsf{if}(\mathsf{eq}(\mathsf{nth}(i, b), z),\ \mathsf{true},\ \lambda_{(169)}(i, b, z) \wedge \mathsf{nextindex}(\mathsf{s}(i), b, z))$

$\theta_{\mathsf{nextindex}}(i, b, z) = \mathsf{true}$ iff z occurs in b at a position whose index is greater than or equal to i or $z = 0$.

function θ_{main} : $\mathsf{nat} \times \mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$
$\theta_{\mathsf{main}}(z, k, b) = \theta_{\mathsf{nextindex}}(0, \mathsf{set}(k, b, z), z)$

$\theta_{\mathsf{main}}(z, k, b) = \mathsf{true}$ iff $k < \mathsf{len}(b)$.

42 sum ([Gri81] p. 180)

Given a list b of length n, the following imperative procedure stores in variable x the sum of b's elements, cf. the introduction to Section B.

```
procedure sum($i, x, n$: nat, $b$: list)
  $i := 0$;  $x := 0$;
  while $i \neq n$ do $x := x + b[i]$;  $i := i + 1$ od
```

Translation

function loop : $\mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{list} \to \mathsf{nat}$
$\mathsf{loop}(i, x, n, b) = \mathsf{if}(\neg\mathsf{eq}(i, n),\ \mathsf{loop}(\mathsf{s}(i), \mathsf{plus}(x, \mathsf{nth}(i, b)), n, b),\ x)$

function main : $\mathsf{nat} \times \mathsf{list} \to \mathsf{nat}$
$\mathsf{main}(n, b) = \mathsf{loop}(0, 0, n, b)$

Termination Hypothesis

$$\neg\mathsf{eq}(i, n) \ \to\ \langle \mathsf{s}(i), \mathsf{plus}(x, \mathsf{nth}(i, b)), n, b\rangle \prec \langle i, x, n, b\rangle \tag{201}$$

Transformation

1. Generalization $\{\mathsf{plus}(x, \mathsf{nth}(i, b))/y\}$ (since $x \notin \{i, n\}$):

$$\neg\mathsf{eq}(i, n) \ \to\ \langle \mathsf{s}(i), y, n, b\rangle \prec \langle i, x, n, b\rangle \tag{202}$$

2. Inductive Evaluation w.r.t. $\mathsf{eq}(i, n)$, Symbolic Evaluation:

$$\mathsf{true} \ \to\ \langle \mathsf{s}(0), y, \mathsf{s}(v), b\rangle \prec \langle 0, x, \mathsf{s}(v), b\rangle \tag{203}$$
$$\mathsf{true} \ \to\ \langle \mathsf{s}(\mathsf{s}(u)), y, 0, b\rangle \prec \langle \mathsf{s}(u), x, 0, b\rangle \tag{204}$$
$$\neg\mathsf{eq}(u, v) \ \to\ \langle \mathsf{s}(\mathsf{s}(u)), y, \mathsf{s}(v), b\rangle + \langle u, x, v, b\rangle \preceq \langle \mathsf{s}(u), x, \mathsf{s}(v), b\rangle + \langle \mathsf{s}(u), y, v, b\rangle \tag{205}$$

3. Premise Elimination in (203) and (205), Rejection in (204)

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \langle i, s, n, b\rangle \mapsto (n - i)^2$

Soundness Predicate
function $\lambda_{(202)}$: $\mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{list} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(202)}(0, x, 0, b, y) = \mathsf{true}$
$\lambda_{(202)}(0, x, \mathsf{s}(v), b, y) = \mathsf{true}$
$\lambda_{(202)}(\mathsf{s}(u), x, 0, b, y) = \mathsf{false}$
$\lambda_{(202)}(\mathsf{s}(u), x, \mathsf{s}(v), b, y) = \lambda_{(202)}(u, x, v, b, y)$

$\lambda_{(202)}(i, x, n, b, y) = \mathsf{true}$ iff $i \leq n$

Termination Predicates
function θ_{loop} : $\mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$
$\theta_{\mathsf{loop}}(i, x, n, b) = \mathsf{if}(\neg\mathsf{eq}(i, n),$
$\quad \lambda_{(202)}(i, x, n, b, \mathsf{plus}(x, \mathsf{nth}(i, b))) \wedge \theta_{\mathsf{loop}}(\mathsf{s}(i), \mathsf{plus}(x, \mathsf{nth}(i, b)), n, b),$
$\quad \mathsf{true})$

Notice that the soundness predicate $\lambda_{(202)}$ is called with the term $\mathsf{plus}(x, \mathsf{nth}(i, b))$. This results from the generalization where $\mathsf{plus}(x, \mathsf{nth}(i, b))$ is replaced with y.
Hence, $\lambda_{(202)}(i, x, n, b, \mathsf{plus}(x, \mathsf{nth}(i, b)))$ entails (201).

$\theta_{\mathsf{loop}}(i, x, n, b) = \mathsf{true}$ iff $i \le n$.

function θ_{main} : nat × list → bool
  $\theta_{\mathsf{main}}(n, b) = \theta_{\mathsf{loop}}(0, 0, n, b)$

$\theta_{\mathsf{main}}(n, b) = \mathsf{true}$.

43 llist_search1 ([Gri81] p. 184)

Given a natural x and a list b containing m lists of length n, $m > 0, n > 0$, the following imperative procedure determines the least indices i and j, $0 \le i < m$ and $0 \le j < n$, such that $b[i, j] = x$. It assigns m to i iff there are no such indices. It terminates iff $m = 0$ or $n \ne 0$ or $b[0, k] = x$ for some $k \ge 0$.

```
procedure llist_search1(b: llist, n, m, x, i, j: nat)
  i := 0; j := 0;
  while $i \ne m$ and $x \ne b[i, j]$ do
     j := j + 1;
     if j = n then
        i := i + 1;  j := 0
     else
        skip
     fi
  od
```

Translation

function loop : llist × nat × nat × nat × nat × nat → nat
$$\begin{aligned}\mathsf{loop}(b, m, n, x, i, j) = \mathsf{if}(&\neg\mathsf{eq}(i, m) \wedge \neg\mathsf{eq}(x, \mathsf{lnth}(i, j, b)),\\ &\mathsf{if}(\mathsf{eq}(\mathsf{s}(j), n),\\ &\quad \mathsf{loop}(b, m, n, x, \mathsf{s}(i), 0),\\ &\quad \mathsf{loop}(b, m, n, x, i, \mathsf{s}(j))),\\ &i)\end{aligned}$$

function main : llist × nat × nat × nat → nat
$$\mathsf{main}(b, m, n, x) = \mathsf{loop}(b, m, n, x, 0, 0)$$

Termination Hypotheses

$$\neg\mathsf{eq}(i, m) \wedge \neg\mathsf{eq}(x, \mathsf{lnth}(i, j, b)) \wedge \mathsf{eq}(\mathsf{s}(j), n) \rightarrow \langle b, m, n, x, \mathsf{s}(i), 0\rangle \prec \langle b, m, n, x, i, j\rangle \quad (206)$$
$$\neg\mathsf{eq}(i, m) \wedge \neg\mathsf{eq}(x, \mathsf{lnth}(i, j, b)) \wedge \neg\mathsf{eq}(\mathsf{s}(j), n) \rightarrow \langle b, m, n, x, i, \mathsf{s}(j)\rangle \prec \langle b, m, n, x, i, j\rangle \quad (207)$$

Transformation

1. In (206), Replace w.r.t. $\mathsf{eq}(\mathsf{s}(j), n)$:
$$\neg\mathsf{eq}(i, m) \wedge \neg\mathsf{eq}(x, \mathsf{lnth}(i, j, b)) \rightarrow \langle b, m, \mathsf{s}(j), x, \mathsf{s}(i), 0\rangle \prec \langle b, m, \mathsf{s}(j), x, i, j\rangle \quad (208)$$
2. In (208), Inverse Weakening eliminating $\neg\mathsf{eq}(x, \mathsf{lnth}(i, j, b))$:
$$\neg\mathsf{eq}(i, m) \rightarrow \langle b, m, \mathsf{s}(j), x, \mathsf{s}(i), 0\rangle \prec \langle b, m, \mathsf{s}(j), x, i, j\rangle \quad (209)$$
3. In (209), Inductive Evaluation w.r.t. $\mathsf{eq}(i, m)$, Symbolic Evaluation:
$$\mathsf{true} \rightarrow \langle b, \mathsf{s}(v), \mathsf{s}(j), x, \mathsf{s}(0), 0\rangle \prec \langle b, \mathsf{s}(v), \mathsf{s}(j), x, 0, j\rangle \quad (210)$$
$$\mathsf{true} \rightarrow \langle b, 0, \mathsf{s}(j), x, \mathsf{s}(\mathsf{s}(u)), 0\rangle \prec \langle b, 0, \mathsf{s}(j), x, \mathsf{s}(u), j\rangle \quad (211)$$
$$\begin{aligned}\neg\mathsf{eq}(u, v) \rightarrow\ &\langle b, \mathsf{s}(v), \mathsf{s}(j), x, \mathsf{s}(\mathsf{s}(u)), 0\rangle + \langle b, v, \mathsf{s}(j), x, u, j\rangle \preceq\\ &\langle b, \mathsf{s}(v), \mathsf{s}(j), x, \mathsf{s}(u), j\rangle + \langle b, v, \mathsf{s}(j), x, \mathsf{s}(u), 0\rangle \quad (212)\end{aligned}$$

4. In (207), Inverse Weakening eliminating $\neg\mathsf{eq}(x, \mathsf{lnth}(i,j,b))$ (even if our heuristic suggests elimination of $\neg\mathsf{eq}(\mathsf{s}(j), n)$):

$$\neg\mathsf{eq}(i,m) \wedge \neg\mathsf{eq}(\mathsf{s}(j),n) \;\rightarrow\; \langle b,m,n,x,i,\mathsf{s}(j)\rangle \prec \langle b,m,n,x,i,j\rangle \tag{213}$$

5. In (213), Inductive Evaluation w.r.t. $\mathsf{eq}(i,m)$, Symbolic Evaluation:

$$\mathsf{true} \wedge \neg\mathsf{eq}(\mathsf{s}(j),n) \;\rightarrow\; \langle b,\mathsf{s}(v),n,x,0,\mathsf{s}(j)\rangle \prec \langle b,\mathsf{s}(v),n,x,0,j\rangle \tag{214}$$
$$\mathsf{true} \wedge \neg\mathsf{eq}(\mathsf{s}(j),n) \;\rightarrow\; \langle b,0,n,x,\mathsf{s}(u),\mathsf{s}(j)\rangle \prec \langle b,0,n,x,\mathsf{s}(u),j\rangle \tag{215}$$
$$\neg\mathsf{eq}(u,v) \wedge \neg\mathsf{eq}(\mathsf{s}(j),n) \;\rightarrow\; \langle b,\mathsf{s}(v),n,x,\mathsf{s}(u),\mathsf{s}(j)\rangle + \langle b,v,n,x,u,j\rangle \preceq \langle b,\mathsf{s}(v),n,x,\mathsf{s}(u),j\rangle + \langle b,v,n,x,u,\mathsf{s}(j)\rangle \tag{216}$$

6. In (214), Inductive Evaluation w.r.t. $\mathsf{eq}(j,n)$ (even if our heuristic does not suggest Inductive Evaluation), Symbolic Evaluation:

$$\mathsf{true} \wedge \neg\mathsf{eq}(\mathsf{s}(0),0) \;\rightarrow\; \langle b,\mathsf{s}(v),0,x,0,\mathsf{s}(0)\rangle \prec \langle b,\mathsf{s}(v),0,x,0,0\rangle \tag{217}$$
$$\mathsf{true} \wedge \neg\mathsf{eq}(\mathsf{s}(0),\mathsf{s}(r)) \;\rightarrow\; \langle b,\mathsf{s}(v),\mathsf{s}(r),x,0,\mathsf{s}(0)\rangle \prec \langle b,\mathsf{s}(v),\mathsf{s}(r),x,0,0\rangle \tag{218}$$
$$\mathsf{true} \wedge \neg\mathsf{eq}(\mathsf{s}(\mathsf{s}(q)),0) \;\rightarrow\; \langle b,\mathsf{s}(v),0,x,0,\mathsf{s}(\mathsf{s}(q))\rangle \prec \langle b,\mathsf{s}(v),0,x,0,\mathsf{s}(q)\rangle \tag{219}$$
$$\mathsf{true} \wedge \neg\mathsf{eq}(\mathsf{s}(q),r) \;\rightarrow\; \langle b,\mathsf{s}(v),\mathsf{s}(r),x,0,\mathsf{s}(\mathsf{s}(q))\rangle + \langle b,\mathsf{s}(v),r,x,0,q\rangle \preceq \langle b,\mathsf{s}(v),\mathsf{s}(r),x,0,\mathsf{s}(q)\rangle + \langle b,\mathsf{s}(v),r,x,0,\mathsf{s}(q)\rangle \tag{220}$$

7. Premise Elimination in (210), (212), (216), (218), and (220), Rejection in (211), (215), (217), and (219)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \mathsf{lempty} \mapsto 0,\ \mathsf{ladd}(v,w) \mapsto w+1,\ \langle b,m,n,x,i,j\rangle \mapsto (n(m-i)-j)^2$

Soundness Predicates

function $\lambda_{(209)} : \mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

$\lambda_{(209)}(b,0,x,0,j) = \mathsf{true}$
$\lambda_{(209)}(b,\mathsf{s}(v),x,0,j) = \mathsf{true}$
$\lambda_{(209)}(b,0,x,\mathsf{s}(u),j) = \mathsf{false}$
$\lambda_{(209)}(b,\mathsf{s}(v),x,\mathsf{s}(u),j) = \lambda_{(209)}(b,v,x,u,j)$

$\lambda_{(209)}(b,m,x,i,j) = \mathsf{true}$ iff $i \le m$

function $\lambda_{(214)} : \mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

$\lambda_{(214)}(b,v,0,x,0) = \mathsf{false}$
$\lambda_{(214)}(b,v,\mathsf{s}(r),x,0) = \mathsf{true}$
$\lambda_{(214)}(b,v,0,x,\mathsf{s}(q)) = \mathsf{false}$
$\lambda_{(214)}(b,v,\mathsf{s}(r),x,\mathsf{s}(q)) = \lambda_{(214)}(b,v,r,x,q)$

$\lambda_{(214)}(b,v,n,x,j) = \mathsf{true}$ iff $j < n$

function $\lambda_{(213)} : \mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

$\lambda_{(213)}(b,0,n,x,0,j) = \mathsf{true}$
$\lambda_{(213)}(b,\mathsf{s}(v),n,x,0,j) = \lambda_{(214)}(b,v,n,x,j)$
$\lambda_{(213)}(b,0,n,x,\mathsf{s}(u),j) = \mathsf{false}$
$\lambda_{(213)}(b,\mathsf{s}(v),n,x,\mathsf{s}(u),j) = \lambda_{(213)}(b,v,n,x,u,j)$

$\lambda_{(213)}(b,m,n,x,i,j) = \mathsf{true}$ iff $i = m$ or $(i < m$ and $j < n)$

Termination Predicate

```
**function** $\theta_{\mathsf{loop}}$ : $\mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
  $\theta_{\mathsf{loop}}(b,m,n,x,i,j) = \mathsf{if}(\neg\mathsf{eq}(i,m) \wedge \neg\mathsf{eq}(x,\mathsf{lnth}(i,j,b)),$
                           $\mathsf{if}(\mathsf{eq}(\mathsf{s}(j),n),$
                              $\lambda_{(209)}(b,m,x,i,j) \wedge \theta_{\mathsf{loop}}(b,m,n,x,\mathsf{s}(i),0),$
                              $\lambda_{(213)}(b,m,n,x,i,j) \wedge \theta_{\mathsf{loop}}(b,m,n,x,i,\mathsf{s}(j))),$
                           $\mathsf{true})$
```

$\theta_{\mathsf{loop}}(b,m,n,x,i,j) = \mathsf{true}$ iff $i \leq m$ and $j < n$ or $x = b[i,j]$ or $i = m$

```
**function** $\theta_{\mathsf{main}}$ : $\mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
  $\theta_{\mathsf{main}}(b,m,n,x) = \theta_{\mathsf{loop}}(b,m,n,x,0,0)$
```

$\theta_{\mathsf{main}}(b,m,n,x) = \mathsf{true}$ iff $m = 0$ or $n \neq 0$ or $x = b[0,0]$.

Note that the termination predicate θ_{main} describes only a proper subset of the domain of the functional procedure main. For instance, $\theta_{\mathsf{main}}([[1,2,3]],1,0,3)$ is evaluated to false, although evaluation of $\mathsf{main}([[1,2,3]],1,0,3)$ halts returning 0. In general, $\theta_{\mathsf{main}}(b,m,n,x)$ returns false even if evaluation of $\mathsf{main}(b,m,n,x)$ halts if and only if $m \neq 0$, $n = 0$, and $b[0,k] = x$ for some $k > 0$. However, this condition is only satisfied for inputs that do not meet the specification of the imperative procedure llist_search1 where $n > 0$ is demanded. Hence, regarding the specification of llist_search1, its termination can be proved using θ_{main}.

44 4_tuplesort ([Gri81] p. 187)

The following imperative procedure sorts the initial values of the four variables w, x, y, z such that finally $w \leq x \leq y \leq z$.

```
**procedure** 4_tuplesort($w, x, y, z, h$:nat)
  **while** $x < w \vee y < x \vee z < y$ **do** ;
    **if** $x < w$ **then**
      $h := w;\ \ w := x;\ \ x := h$
    **else**
      **if** $y < x$ **then**
        $h := x;\ \ x := y;\ \ y := h$
      **else**
        **if** $z < y$ **then**
          $h := y;\ \ y := z;\ \ z := h$
        **else**
          **skip**
        **fi**
      **fi**
    **fi**
  **od**
```

Translation

```
**function** loop : $\mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$
  $\mathsf{loop}(w,x,y,z,h) = \mathsf{if}(\mathsf{lt}(x,w) \vee \mathsf{lt}(y,x) \vee \mathsf{lt}(z,y),$
                        $\mathsf{if}(\mathsf{lt}(x,w),$
                           $\mathsf{loop}(x,w,y,z,w),$
                           $\mathsf{if}(\mathsf{lt}(y,x),$
                              $\mathsf{loop}(w,y,x,z,x),$
                              $\mathsf{if}(\mathsf{lt}(z,y),$
                                 $\mathsf{loop}(w,x,z,y,y),$
                                 $\mathsf{loop}(w,x,y,z,h))))$
                        $w)$
```

function $\mathsf{main} : \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$
$\mathsf{main}(w, x, y, z, h) = \mathsf{loop}(w, x, y, z, h)$

Termination Hypotheses

$$(\mathsf{lt}(x,w) \vee \mathsf{lt}(y,x) \vee \mathsf{lt}(z,y)) \wedge \mathsf{lt}(x,w) \to \langle x,w,y,z,w\rangle \prec \langle w,x,y,z,h\rangle \quad (221)$$
$$(\mathsf{lt}(x,w) \vee \mathsf{lt}(y,x) \vee \mathsf{lt}(z,y)) \wedge \neg\mathsf{lt}(x,w) \wedge \mathsf{lt}(y,x) \to \langle w,y,x,z,x\rangle \prec \langle w,x,y,z,h\rangle \quad (222)$$
$$(\mathsf{lt}(x,w) \vee \mathsf{lt}(y,x) \vee \mathsf{lt}(z,y)) \wedge \neg\mathsf{lt}(x,w) \wedge \neg\mathsf{lt}(y,x) \wedge \mathsf{lt}(z,y) \to \langle w,x,z,y,y\rangle \prec \langle w,x,y,z,h\rangle \quad (223)$$
$$(\mathsf{lt}(x,w) \vee \mathsf{lt}(y,x) \vee \mathsf{lt}(z,y)) \wedge \neg\mathsf{lt}(x,w) \wedge \neg\mathsf{lt}(y,x) \wedge \neg\mathsf{lt}(z,y) \to \langle w,x,z,y,h\rangle \prec \langle w,x,y,z,h\rangle \quad (224)$$

Transformation

1. In (221), Inductive Evaluation w.r.t. $\mathsf{lt}(x,w)$, Symbolic Evaluation (Here we use the heuristic to select that term for Inductive Evaluation which contains the maximum number of arguments changed in the recursive call.):
$$(\mathsf{true} \vee \mathsf{lt}(y,0) \vee \mathsf{lt}(z,y)) \wedge \mathsf{true} \;\to\; \langle 0,\mathsf{s}(v),y,z,\mathsf{s}(v)\rangle \prec \langle \mathsf{s}(v),0,y,z,h\rangle \quad (225)$$
$$(\mathsf{lt}(u,v) \vee \mathsf{lt}(y,\mathsf{s}(u)) \vee \mathsf{lt}(z,y)) \wedge \mathsf{lt}(u,v) \;\to\; \langle \mathsf{s}(u),\mathsf{s}(v),y,z,\mathsf{s}(v)\rangle + \langle v,u,y,z,h\rangle \preceq \langle \mathsf{s}(v),\mathsf{s}(u),y,z,h\rangle + \langle u,v,y,z,v\rangle \quad (226)$$
2. In (222), Inverse Weakening eliminating $(\mathsf{lt}(x,w) \vee \mathsf{lt}(y,x) \vee \mathsf{lt}(z,y)) \wedge \neg\mathsf{lt}(x,w)$:
$$\mathsf{lt}(y,x) \;\to\; \langle w,y,x,z,x\rangle \prec \langle w,x,y,z,h\rangle \quad (227)$$
3. In (227), Inductive Evaluation w.r.t. $\mathsf{lt}(y,x)$, Symbolic Evaluation:
$$\mathsf{true} \;\to\; \langle w,0,\mathsf{s}(v),z,\mathsf{s}(v)\rangle \prec \langle w,\mathsf{s}(v),0,z,h\rangle \quad (228)$$
$$\mathsf{lt}(u,v) \;\to\; \langle w,\mathsf{s}(u),\mathsf{s}(v),z,\mathsf{s}(v)\rangle + \langle w,v,u,z,h\rangle \preceq \langle w,\mathsf{s}(v),\mathsf{s}(u),z,h\rangle + \langle w,u,v,z,v\rangle \quad (229)$$
4. In (223), Inverse Weakening eliminating $(\mathsf{lt}(x,w) \vee \mathsf{lt}(y,x) \vee \mathsf{lt}(z,y)) \wedge \neg\mathsf{lt}(x,w) \wedge \neg\mathsf{lt}(y,x)$:
$$\mathsf{lt}(z,y) \;\to\; \langle w,x,z,y,y\rangle \prec \langle w,x,y,z,h\rangle \quad (230)$$
5. In (230), Inductive Evaluation w.r.t. $\mathsf{lt}(z,y)$, Symbolic Evaluation:
$$\mathsf{true} \;\to\; \langle w,x,0,\mathsf{s}(v),\mathsf{s}(v)\rangle \prec \langle w,x,\mathsf{s}(v),0,h\rangle \quad (231)$$
$$\mathsf{lt}(u,v) \;\to\; \langle x,w,\mathsf{s}(u),\mathsf{s}(v),\mathsf{s}(v)\rangle + \langle x,w,v,u,h\rangle \preceq \langle w,x,\mathsf{s}(v),\mathsf{s}(u),h\rangle + \langle w,x,u,v,v\rangle \quad (232)$$
6. In (224), Inductive Evaluation w.r.t. to any term $\mathsf{lt}(a,b)$ yields no consequences as the premise of (224) is unsatisfiable.
7. Premise Elimination in (225)–(232)

Polynomial Ordering
$0 \mapsto 0,\; \mathsf{s}(v) \mapsto v+1,\; \langle w,x,y,z,h\rangle \mapsto 3w+2x+y$

Soundness Predicates

function $\lambda_{(221)} : \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(221)}(w,x,y,z,h) = \mathsf{true}$

function $\lambda_{(227)} : \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(227)}(w,x,y,z,h) = \mathsf{true}$

function $\lambda_{(230)} : \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(230)}(w,x,y,z,h) = \mathsf{true}$

function $\lambda_{(224)} : \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(224)}(w,x,y,z,h) = \mathsf{true}$

Termination Predicate

function $\theta_{\mathsf{4\_tuplesort}} : \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\theta_{\mathsf{4\_tuplesort}}(w,x,y,z,h) = \mathsf{true}$

45 sqrtl ([Gri81] p. 196)

Given a natural n, the following imperative procedure computes the maximum natural a such that $a^2 \leq n$.

```
procedure sqrtl($n, a$: nat)
  $a := 0$;
  while $(a+1)^2 \leq n$ do $a := a + 1$ od
```

Translation

function $\mathsf{loop} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$
$\mathsf{loop}(n,a) = \mathsf{if}(\mathsf{le}(\mathsf{sqr}(\mathsf{s}(a)),n),\ \mathsf{loop}(n,\mathsf{s}(a)),\ a)$

function $\mathsf{main} : \mathsf{nat} \to \mathsf{nat}$
$\mathsf{main}(n) = \mathsf{loop}(n,0)$

Termination Hypothesis

$$\mathsf{le}(\mathsf{sqr}(\mathsf{s}(a)),n) \ \to\ \langle n,\mathsf{s}(a)\rangle \prec \langle n,a\rangle \tag{233}$$

Transformation

1. Inductive Evaluation w.r.t. $\mathsf{lt}(a,n)$ (even if our heuristic suggests *no* Inductive Evaluation), Symbolic Evaluation:

$$\mathsf{le}(\mathsf{sqr}(\mathsf{s}(0)),\mathsf{s}(v)) \ \to\ \langle \mathsf{s}(v),\mathsf{s}(0)\rangle \prec \langle \mathsf{s}(v),0\rangle \tag{234}$$

$$\begin{aligned}\mathsf{le}(\mathsf{s}(\mathsf{plus}(\mathsf{sqr}(\mathsf{s}(u)),\mathsf{dbl}(\mathsf{s}(u)))),\mathsf{s}(v)) \ \to\ & \langle \mathsf{s}(v),\mathsf{s}(\mathsf{s}(u))\rangle + \langle v,u\rangle \preceq \\ & \langle \mathsf{s}(v),\mathsf{s}(u)\rangle + \langle v,\mathsf{s}(u)\rangle\end{aligned} \tag{235}$$

   Notice that for the application of the induction hypothesis the formula $\mathsf{le}(\mathsf{sqr}(\mathsf{s}(\mathsf{s}(u))),\mathsf{s}(v)) \to \mathsf{le}(\mathsf{sqr}(\mathsf{s}(u)),v)$ has to be proved by *Induction.*
2. Premise Elimination in (234) and (235)

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle n,a\rangle \mapsto (n-a)^2$

Soundness Predicate

function $\lambda_{(233)} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(233)}(n,a) = \mathsf{true}$

Termination Predicates

function $\theta_{\mathsf{loop}} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\theta_{\mathsf{loop}}(n,a) = \mathsf{true}$

function $\theta_{\mathsf{main}} : \mathsf{nat} \to \mathsf{bool}$
$\theta_{\mathsf{main}}(n) = \mathsf{true}$

46 plateau ([Gri81] p. 204)

A *plateau* of an ordered n-element list b is a sequence of equal values. The following imperative procedure stores in variable p the length of the longest plateau of b. It terminates iff $n > 0$.

```
procedure plateau($n, i, p$: nat, $b$: list)
  $i := 1$;  $p := 1$;
  while $i \neq n$ do
    if $b[i] = b[i-p]$ then
      $i := i + 1$;  $p := p + 1$
    else
      $i := i + 1$
    fi
  od
```

Translation

function loop : nat × nat × nat × list ⟶ nat

$$\begin{aligned}\mathsf{loop}(n,i,p,b) = \mathsf{if}(&\neg\mathsf{eq}(i,n),\\ &\mathsf{if}(\mathsf{eq}(\mathsf{nth}(i,b),\mathsf{nth}(\mathsf{minus}(i,p),b)),\\ &\quad \mathsf{loop}(n,\mathsf{s}(i),\mathsf{s}(p),b),\\ &\quad \mathsf{loop}(n,\mathsf{s}(i),p,b)),\\ &p)\end{aligned}$$

function main : nat × list ⟶ nat

$$\mathsf{main}(n,b) = \mathsf{loop}(n,\mathsf{s}(0),\mathsf{s}(0),b)$$

Termination Hypotheses

$$\neg\mathsf{eq}(i,n) \wedge \mathsf{eq}(\mathsf{nth}(i,b),\mathsf{nth}(\mathsf{minus}(i,p),b)) \;\rightarrow\; \langle n,\mathsf{s}(i),\mathsf{s}(p),b\rangle \prec \langle n,i,p,b\rangle \tag{236}$$

$$\neg\mathsf{eq}(i,n) \wedge \neg\mathsf{eq}(\mathsf{nth}(i,b),\mathsf{nth}(\mathsf{minus}(i,p),b)) \;\rightarrow\; \langle n,\mathsf{s}(i),p,b)\rangle \prec \langle n,i,p,b\rangle \tag{237}$$

Transformation

1. In (236), Inverse Weakening eliminating $\mathsf{eq}(\mathsf{nth}(i,b),\mathsf{nth}(\mathsf{minus}(i,p),b))$:

$$\neg\mathsf{eq}(i,n) \;\rightarrow\; \langle n,\mathsf{s}(i),\mathsf{s}(p),b\rangle \prec \langle n,i,p,b\rangle \tag{238}$$

2. In (238), Inductive Evaluation w.r.t. $\mathsf{eq}(i,n)$, Symbolic Evaluation:

$$\mathsf{true} \;\rightarrow\; \langle \mathsf{s}(v),\mathsf{s}(0),\mathsf{s}(p),b\rangle \prec \langle \mathsf{s}(v),0,p,b\rangle \tag{239}$$

$$\mathsf{true} \;\rightarrow\; \langle 0,\mathsf{s}(\mathsf{s}(u)),\mathsf{s}(p),b\rangle \prec \langle 0,\mathsf{s}(u),p,b\rangle \tag{240}$$

$$\begin{aligned}\neg\mathsf{eq}(u,v) \;\rightarrow\; &\langle \mathsf{s}(v),\mathsf{s}(\mathsf{s}(u)),\mathsf{s}(p),b\rangle + \langle v,u,p,b\rangle \preceq\\ &\langle \mathsf{s}(v),\mathsf{s}(u),p,b\rangle + \langle v,\mathsf{s}(u),\mathsf{s}(p),b\rangle\end{aligned} \tag{241}$$

3. In (237), Inverse Weakening eliminating $\neg\mathsf{eq}(\mathsf{nth}(i,b),\mathsf{nth}(\mathsf{minus}(i,p),b))$:

$$\neg\mathsf{eq}(i,n) \;\rightarrow\; \langle n,\mathsf{s}(i),p,b)\rangle \prec \langle n,i,p,b\rangle \tag{242}$$

4. In (242), Inductive Evaluation w.r.t. $\mathsf{eq}(i,n)$, Symbolic Evaluation:

$$\mathsf{true} \;\rightarrow\; \langle \mathsf{s}(v),\mathsf{s}(0),p,b)\rangle \prec \langle \mathsf{s}(v),0,p,b\rangle \tag{243}$$

$$\mathsf{true} \;\rightarrow\; \langle 0,\mathsf{s}(\mathsf{s}(u)),p,b)\rangle \prec \langle 0,\mathsf{s}(u),p,b\rangle \tag{244}$$

$$\begin{aligned}\neg\mathsf{eq}(u,v) \;\rightarrow\; &\langle \mathsf{s}(v),\mathsf{s}(\mathsf{s}(u)),p,b)\rangle + \langle v,u,p,b\rangle \preceq\\ &\langle \mathsf{s}(v),\mathsf{s}(u),p,b\rangle + \langle v,\mathsf{s}(u),p,b)\rangle\end{aligned} \tag{245}$$

5. Premise Elimination in (239), (241), (243), and (245), Rejection in (240) and (244)

Polynomial Ordering

$0 \mapsto 0$, $\mathsf{s}(v) \mapsto v+1$, $\langle n,i,p,b\rangle \mapsto (n-i)^2$

Soundness Predicates

function $\lambda_{(238)}$: nat $\times$ nat $\times$ nat $\times$ list $\to$ bool

$\lambda_{(238)}(0, 0, p, b) = \mathsf{true}$
$\lambda_{(238)}(\mathsf{s}(v), 0, p, b) = \mathsf{true}$
$\lambda_{(238)}(0, \mathsf{s}(u), p, b) = \mathsf{false}$
$\lambda_{(238)}(\mathsf{s}(v), \mathsf{s}(u), p, b) = \lambda_{(238)}(v, u, p, b)$

$\lambda_{(238)}(n, i, p, b) = \mathsf{true}$ iff $i \leq n$

function $\lambda_{(242)}$: nat $\times$ nat $\times$ nat $\times$ list $\to$ bool

$\lambda_{(242)}(0, 0, p, b) = \mathsf{true}$
$\lambda_{(242)}(\mathsf{s}(v), 0, p, b) = \mathsf{true}$
$\lambda_{(242)}(0, \mathsf{s}(u), p, b) = \mathsf{false}$
$\lambda_{(242)}(\mathsf{s}(v), \mathsf{s}(u), p, b) = \lambda_{(242)}(v, u, p, b)$

$\lambda_{(242)}(n, i, p, b) = \mathsf{true}$ iff $i \leq n$

Termination Predicates

function θ_{loop} : nat $\times$ nat $\times$ nat $\times$ list $\to$ bool

```
θ_loop(n, i, p, b) = if( ¬eq(i, n),
                         if( eq(nth(i, b), nth(minus(i, p), b)),
                             λ_(238)(n, i, p, b) ∧ θ_loop(n, s(i), s(p), b),
                             λ_(242)(n, i, p, b) ∧ θ_loop(n, s(i), p, b)),
                         true)
```

$\theta_{\mathsf{loop}}(n, i, p, b) = \mathsf{true}$ iff $i \leq n$.

function θ_{main} : nat $\times$ list $\to$ bool

$\theta_{\mathsf{main}}(n, b) = \theta_{\mathsf{loop}}(n, \mathsf{s}(0), \mathsf{s}(0), b)$

$\theta_{\mathsf{main}}(n, b) = \mathsf{true}$ iff $n > 0$.

47 common_elem ([Gri81] p. 210)

Given three sorted lists f, g, h of naturals, the following functional procedure determines the three indices i, j, k of a natural occurring in all lists iff such a natural exists. Otherwise it also terminates satisfying $f[i] = g[j] = h[k] = 0$.

```
procedure common_elem($f, g, h$ : list, $i, j, k$ : nat)
  $i := 0$;  $j := 0$;  $k := 0$;
  while $f[i] < g[j] \vee g[j] < h[k] \vee h[k] < f[i]$ do
     if $f[i] < g[j]$ then
        $i := i + 1$
     else
        if $g[j] < h[k]$ then
           $j := j + 1$
        else
           $k := k + 1$
        fi
     fi
  od
```

Translation

function loop : list × list × list × nat × nat × nat → nat

```
loop(f, g, h, i, j, k) = if( lt(nth(i, f), nth(j, g)) ∨ lt(nth(j, g), nth(k, h)) ∨ lt(nth(k, h), nth(i, f)),
                            if( lt(nth(i, f), nth(j, g)),
                               loop(f, g, h, s(i), j, k),
                               if( lt(nth(j, g), nth(k, h)),
                                  loop(f, g, h, i, s(j), k),
                                  loop(f, g, h, i, j, s(k)))),
                            i)
```

function main : list × list × list → nat

$\mathsf{main}(f,g,h) = \mathsf{loop}(f,g,h,0,0,0)$

Termination Hypothesis

$$(\mathsf{lt}(\mathsf{nth}(i,f),\mathsf{nth}(j,g)) \lor \mathsf{lt}(\mathsf{nth}(j,g),\mathsf{nth}(k,h)) \lor \mathsf{lt}(\mathsf{nth}(k,h),\mathsf{nth}(i,f))) \land \mathsf{lt}(\mathsf{nth}(i,f),\mathsf{nth}(j,g)) \;\rightarrow\; \langle f,g,h,\mathsf{s}(i),j,k\rangle \prec \langle f,g,h,i,j,k\rangle \tag{246}$$

$$(\mathsf{lt}(\mathsf{nth}(i,f),\mathsf{nth}(j,g)) \lor \mathsf{lt}(\mathsf{nth}(j,g),\mathsf{nth}(k,h)) \lor \mathsf{lt}(\mathsf{nth}(k,h),\mathsf{nth}(i,f))) \land \neg\mathsf{lt}(\mathsf{nth}(i,f),\mathsf{nth}(j,g)) \land \mathsf{lt}(\mathsf{nth}(j,g),\mathsf{nth}(k,h)) \;\rightarrow\; \langle f,g,h,i,\mathsf{s}(j),k\rangle \prec \langle f,g,h,i,j,k\rangle \tag{247}$$

$$(\mathsf{lt}(\mathsf{nth}(i,f),\mathsf{nth}(j,g)) \lor \mathsf{lt}(\mathsf{nth}(j,g),\mathsf{nth}(k,h)) \lor \mathsf{lt}(\mathsf{nth}(k,h),\mathsf{nth}(i,f))) \land\neg\mathsf{lt}(\mathsf{nth}(i,f),\mathsf{nth}(j,g)) \land \neg\mathsf{lt}(\mathsf{nth}(j,g),\mathsf{nth}(k,h)) \;\rightarrow\; \langle f,g,h,i,j,\mathsf{s}(k)\rangle \prec \langle f,g,h,i,j,k\rangle \tag{248}$$

Transformation

1. In (246), Inductive Evaluation w.r.t. $\mathsf{nth}(i,f)$, Symbolic Evaluation:

$$(\mathsf{lt}(0,\mathsf{nth}(j,g)) \lor \mathsf{lt}(\mathsf{nth}(j,g),\mathsf{nth}(k,h)) \lor \mathsf{false}) \land \mathsf{lt}(0,\mathsf{nth}(j,g)) \;\rightarrow\; \langle \mathsf{empty},g,h,\mathsf{s}(i),j,k\rangle \prec \langle \mathsf{empty},g,h,i,j,k\rangle \tag{249}$$

$$(\mathsf{lt}(v,\mathsf{nth}(j,g)) \lor \mathsf{lt}(\mathsf{nth}(j,g),\mathsf{nth}(k,h)) \lor \mathsf{lt}(\mathsf{nth}(k,h),v)) \land \mathsf{lt}(v,\mathsf{nth}(j,g)) \;\rightarrow\; \langle \mathsf{add}(v,w),g,h,\mathsf{s}(0),j,k\rangle \prec \langle \mathsf{add}(v,w),g,h,0,j,k\rangle \tag{250}$$

$$\mathsf{lt}(\mathsf{nth}(u,w),\mathsf{nth}(j,g)) \lor \mathsf{lt}(\mathsf{nth}(j,g),\mathsf{nth}(k,h)) \lor \mathsf{lt}(\mathsf{nth}(k,h),\mathsf{nth}(u,w)) \land\mathsf{lt}(\mathsf{nth}(u,w),\mathsf{nth}(j,g)) \;\rightarrow\; \langle \mathsf{add}(v,w),g,h,\mathsf{s}(\mathsf{s}(u)),j,k\rangle + \langle w,g,h,u,j,k\rangle \preceq \langle \mathsf{add}(v,w),g,h,\mathsf{s}(u),j,k\rangle + \langle w,g,h,\mathsf{s}(u),j,k\rangle \tag{251}$$

2. In (249), Inductive Evaluation w.r.t. $\mathsf{nth}(j,g)$, Symbolic Evaluation:

$$(\mathsf{lt}(0,r) \lor \mathsf{lt}(r,\mathsf{nth}(k,h)) \lor \mathsf{false}) \land \mathsf{lt}(0,r) \;\rightarrow\; \langle \mathsf{empty},\mathsf{add}(r,b),h,\mathsf{s}(i),0,k\rangle \prec \langle \mathsf{empty},\mathsf{add}(r,b),h,i,0,k\rangle \tag{252}$$

$$(\mathsf{lt}(0,\mathsf{nth}(y,b)) \lor \mathsf{lt}(\mathsf{nth}(y,b),\mathsf{nth}(k,h)) \lor \mathsf{lt}(\mathsf{nth}(k,h),0)) \land \mathsf{lt}(0,\mathsf{nth}(y,b)) \;\rightarrow\; \langle \mathsf{empty},\mathsf{add}(r,b),h,\mathsf{s}(i),\mathsf{s}(y),k\rangle + \langle \mathsf{empty},b,h,i,y,k\rangle \preceq \langle \mathsf{empty},\mathsf{add}(r,b),h,i,\mathsf{s}(y),k\rangle + \langle \mathsf{empty},b,h,\mathsf{s}(i),y,k\rangle \tag{253}$$

   In (252), Inductive Evaluation w.r.t. $\mathsf{nth}(k,h)$ is suggested by our heuristic, but does not result in a "better" termination predicate.

3. In (250), Inductive Evaluation w.r.t. $\mathsf{nth}(j,g)$, Symbolic Evaluation:

$$(\mathsf{lt}(v,r) \lor \mathsf{lt}(r,\mathsf{nth}(k,h)) \lor \mathsf{lt}(\mathsf{nth}(k,h),v)) \land \mathsf{lt}(v,r) \;\rightarrow\; \langle \mathsf{add}(v,w),\mathsf{add}(r,b),h,\mathsf{s}(0),0,k\rangle \prec \langle \mathsf{add}(v,w),\mathsf{add}(r,b),h,0,0,k\rangle \tag{254}$$

$$(\mathsf{lt}(v,\mathsf{nth}(y,b)) \lor \mathsf{lt}(\mathsf{nth}(y,b),\mathsf{nth}(k,h)) \lor \mathsf{lt}(\mathsf{nth}(k,h),v)) \land \mathsf{lt}(v,\mathsf{nth}(y,b)) \;\rightarrow\; \langle \mathsf{add}(v,w),\mathsf{add}(r,b),h,\mathsf{s}(0),\mathsf{s}(y),k\rangle + \langle \mathsf{add}(v,w),b,h,0,y,k\rangle \preceq \langle \mathsf{add}(v,w),\mathsf{add}(r,b),h,0,\mathsf{s}(y),k\rangle + \langle \mathsf{add}(v,w),b,h,\mathsf{s}(0),y,k\rangle \tag{255}$$

4. In (254), Inductive Evaluation w.r.t. $\mathsf{nth}(k,h)$, Symbolic Evaluation:

$$(\mathsf{lt}(v,r) \vee \mathsf{false} \vee \mathsf{lt}(0,v)) \wedge \mathsf{lt}(v,r) \rightarrow \langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{empty}, \mathsf{s}(0), 0, k\rangle \prec \langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{empty}, 0, 0, k\rangle \quad (256)$$

$$(\mathsf{lt}(v,r) \vee \mathsf{lt}(r,t) \vee \mathsf{lt}(t,v)) \wedge \mathsf{lt}(v,r) \rightarrow \langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), \mathsf{s}(0), 0, 0\rangle \prec \langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), 0, 0, 0\rangle \quad (257)$$

$$\begin{aligned}(\mathsf{lt}(v,r) \vee \mathsf{lt}(r,\mathsf{nth}(z,c)) \vee \mathsf{lt}(\mathsf{nth}(z,c),v)) \wedge \mathsf{lt}(v,r) \\ \rightarrow \langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), \mathsf{s}(0), 0, \mathsf{s}(z)\rangle + \langle \mathsf{add}(v,w), \mathsf{add}(r,b), c, 0, 0, z\rangle \preceq \\ \langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), 0, 0, \mathsf{s}(z)\rangle + \langle \mathsf{add}(v,w), \mathsf{add}(r,b), c, \mathsf{s}(0), 0, z\rangle \quad (258)\end{aligned}$$

5. Premise Elimination in (251), (253), (255), (257), and (258), Rejection in (252) and (256):

$$\begin{aligned}\langle \mathsf{add}(v,w), g, h, \mathsf{s}(\mathsf{s}(u)), j, k\rangle + \langle w, g, h, u, j, k\rangle \preceq \\ \langle \mathsf{add}(v,w), g, h, \mathsf{s}(u), j, k\rangle + \langle w, g, h, \mathsf{s}(u), j, k\rangle\end{aligned}$$

$$\begin{aligned}\langle \mathsf{empty}, \mathsf{add}(r,b), h, \mathsf{s}(i), \mathsf{s}(y), k\rangle + \langle \mathsf{empty}, b, h, i, y, k\rangle \preceq \\ \langle \mathsf{empty}, \mathsf{add}(r,b), h, i, \mathsf{s}(y), k\rangle + \langle \mathsf{empty}, b, h, \mathsf{s}(i), y, k\rangle\end{aligned}$$

$$\begin{aligned}\langle \mathsf{add}(v,w), \mathsf{add}(r,b), h, \mathsf{s}(0), \mathsf{s}(y), k\rangle + \langle \mathsf{add}(v,w), b, h, 0, y, k\rangle \preceq \\ \langle \mathsf{add}(v,w), \mathsf{add}(r,b), h, 0, \mathsf{s}(y), k\rangle + \langle \mathsf{add}(v,w), b, h, \mathsf{s}(0), y, k\rangle\end{aligned}$$

$$\begin{aligned}\langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), \mathsf{s}(0), 0, 0\rangle \prec \\ \langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), 0, 0, 0\rangle\end{aligned}$$

$$\begin{aligned}\langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), \mathsf{s}(0), 0, \mathsf{s}(z)\rangle + \langle \mathsf{add}(v,w), \mathsf{add}(r,b), c, 0, 0, z\rangle \preceq \\ \langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), 0, 0, \mathsf{s}(z)\rangle + \langle \mathsf{add}(v,w), \mathsf{add}(r,b), c, \mathsf{s}(0), 0, z\rangle\end{aligned}$$

6. By analogous transformation of the second and the third termination hypotheses, (247) and (248), we obtain the strict inequalities

$$\langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), 0, \mathsf{s}(0), 0\rangle \prec \langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), 0, 0, 0\rangle$$
$$\langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), 0, 0, \mathsf{s}(0)\rangle \prec \langle \mathsf{add}(v,w), \mathsf{add}(r,b), \mathsf{add}(t,c), 0, 0, 0\rangle$$

and further non-strict inequalities.

Polynomial Ordering

$0 \mapsto 0$, $\mathsf{s}(v) \mapsto v{+}1$, $\mathsf{empty} \mapsto 0$, $\mathsf{add}(v,w) \mapsto w{+}1$, $\langle f,g,h,i,j,k\rangle \mapsto (f-i)^2+(g-j)^2+(h-k)^2$

Soundness Predicates

function $\lambda_{(249)} : \mathsf{list} \times \mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

$$\begin{aligned}\lambda_{(249)}(\mathsf{empty}, h, i, j, k) &= \mathsf{true}\\ \lambda_{(249)}(\mathsf{add}(r,b), h, i, 0, k) &= \mathsf{false}\\ \lambda_{(249)}(\mathsf{add}(r,b), h, i, \mathsf{s}(y), k) &= \lambda_{(249)}(b, h, i, y, k)\end{aligned}$$

$\lambda_{(249)}(g,h,i,j,k) = \mathsf{true}$ iff $\mathsf{len}(g) \le j$

function $\lambda_{(254)} : \mathsf{list} \times \mathsf{list} \times \mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

$$\begin{aligned}\lambda_{(254)}(\mathsf{empty}, w, b, k, v, r) &= \mathsf{false}\\ \lambda_{(254)}(\mathsf{add}(t,c), w, b, 0, v, r) &= \mathsf{true}\\ \lambda_{(254)}(\mathsf{add}(t,c), w, b, \mathsf{s}(z), v, r) &= \lambda_{(254)}(c, w, b, z, v, r)\end{aligned}$$

$\lambda_{(254)}(h,w,b,k,v,r) = \mathsf{true}$ iff $k < \mathsf{len}(h)$

function $\lambda_{(250)} : \mathsf{list} \times \mathsf{list} \times \mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

$$\begin{aligned}\lambda_{(250)}(\mathsf{empty}, h, w, j, k, v) &= \mathsf{true}\\ \lambda_{(250)}(\mathsf{add}(r,b), h, w, 0, k, v) &= \lambda_{(254)}(h, b, w, b, k, v, r)\\ \lambda_{(250)}(\mathsf{add}(r,b), h, w, \mathsf{s}(y), k, v) &= \lambda_{(250)}(b, h, w, y, k, v)\end{aligned}$$

$\lambda_{(250)}(g, h, w, j, k, v) = \mathsf{true}$ iff $\mathsf{len}(g) \leq j \vee (j < \mathsf{len}(g) \wedge k < \mathsf{len}(h))$

```
function $\lambda_{(246)}$ : list × list × list × nat × nat × nat → bool
    $\lambda_{(246)}(\mathsf{empty}, g, h, i, j, k) = \lambda_{(249)}(g, h, i, j, k)$
    $\lambda_{(246)}(\mathsf{add}(v, w), g, h, 0, j, k) = \lambda_{(250)}(g, h, w, j, k, v)$
    $\lambda_{(246)}(\mathsf{add}(v, w), g, h, \mathsf{s}(u), j, k) = \lambda_{(246)}(w, g, h, u, j, k)$
```

$\lambda_{(246)}(f, g, h, i, j, k) = \mathsf{true}$ iff $(\mathsf{len}(f) \leq i \wedge \mathsf{len}(g) \leq j) \vee (i < \mathsf{len}(f) \wedge \mathsf{len}(g) \leq j) \vee (i < \mathsf{len}(f) \wedge j < \mathsf{len}(g) \wedge k < \mathsf{len}(h))$

Analogously we obtain functional procedures for $\lambda_{(247)}$ and $\lambda_{(248)}$ such that

$\lambda_{(247)}(f, g, h, i, j, k) = \mathsf{true}$ iff $(\mathsf{len}(g) \leq j \wedge \mathsf{len}(h) \leq k) \vee (j < \mathsf{len}(g) \wedge \mathsf{len}(h) \leq k) \vee (i < \mathsf{len}(f) \wedge j < \mathsf{len}(g) \wedge k < \mathsf{len}(h))$ and

$\lambda_{(248)}(f, g, h, i, j, k) = \mathsf{true}$ iff $(\mathsf{len}(h) \leq k \wedge \mathsf{len}(f) \leq i) \vee (k < \mathsf{len}(h) \wedge \mathsf{len}(f) \leq i) \vee (i < \mathsf{len}(f) \wedge j < \mathsf{len}(g) \wedge k < \mathsf{len}(h))$.

Termination Predicates

```
function $\theta_{\mathsf{loop}}$ : list × list × list × nat × nat × nat → bool
    $\theta_{\mathsf{loop}}(f, g, h, i, j, k) = \mathsf{if}(\mathsf{lt}(\mathsf{nth}(i, f), \mathsf{nth}(j, g)) \vee \mathsf{lt}(\mathsf{nth}(j, g), \mathsf{nth}(k, h)) \vee \mathsf{lt}(\mathsf{nth}(k, h), \mathsf{nth}(i, f)),$
                                   $\mathsf{if}(\mathsf{lt}(\mathsf{nth}(i, f), \mathsf{nth}(j, g)),$
                                      $\lambda_{(246)}(f, g, h, i, j, k) \wedge \theta_{\mathsf{loop}}(f, g, h, \mathsf{s}(i), j, k),$
                                      $\mathsf{if}(\mathsf{lt}(\mathsf{nth}(j, g), \mathsf{nth}(k, h)),$
                                         $\lambda_{(247)}(f, g, h, i, j, k) \wedge \theta_{\mathsf{loop}}(f, g, h, i, \mathsf{s}(j), k),$
                                         $\lambda_{(248)}(f, g, h, i, j, k) \wedge \theta_{\mathsf{loop}}(f, g, h, i, j, \mathsf{s}(k)))),$
                                   $\mathsf{true})$
```

$\theta_{\mathsf{loop}}(f, g, h, i, j, k) = \mathsf{true}$.

```
function $\theta_{\mathsf{main}}$ : list × list × list → bool
    $\theta_{\mathsf{main}}(f, g, h) = \theta_{\mathsf{loop}}(f, g, h, 0, 0, 0)$
```

$\theta_{\mathsf{main}}(f, g, h) = \mathsf{true}$.

48 swap_equals ([Gri81] p. 213)

Given a list b with two non-overlapping *sections* $b[i : i + n - 1]$ and $b[j : j + n - 1]$, both of length $n \geq 0$, the following imperative procedure swaps the two sections.

```
procedure swap_equals($b$ : list, $i, j, k, n, h$ : nat)
  $k := 0$;
  while $k \neq n$ do
     $h := b[i + k]$;
     $b[i + k] := b[j + k]$;
     $b[j + k] := h$;
     $k := k + 1$;
  od
```

Translation

```
function loop : list × nat × nat × nat × nat × nat → list
  loop(b, i, j, k, n, h) = if( ¬eq(k, n),
                                loop( set( plus(j, k),
                                           set(plus(i, k), b, nth(plus(j, k), b)),
                                           nth(plus(i, k), b)),
                                      i,
                                      j,
                                      s(k),
                                      n,
                                      nth(plus(i, k), b)),
                                b)
```

```
function main : list × nat × nat × nat × nat → list
  main(b, i, j, n, h) = loop(b, i, j, 0, n, h)
```

Termination Hypothesis

$$\neg\mathsf{eq}(k,n) \rightarrow$$
$$\langle \mathsf{set}(\mathsf{plus}(j,k), \mathsf{set}(\mathsf{plus}(i,k), b, \mathsf{nth}(\mathsf{plus}(j,k), b)), \mathsf{nth}(\mathsf{plus}(i,k), b)), i, j, \mathsf{s}(k), n, \mathsf{nth}(\mathsf{plus}(i,k), b)\rangle \prec$$
$$\langle b, i, j, k, n, h\rangle$$

Transformation

1. Generalization $\{\mathsf{set}(\mathsf{plus}(j,k), \mathsf{set}(\mathsf{plus}(i,k), b, \mathsf{nth}(\mathsf{plus}(j,k), b)), \mathsf{nth}(\mathsf{plus}(i,k), b))/y,$ $\mathsf{nth}(\mathsf{plus}(i,k), b)/z\}$ (since $b, h \notin \{k, n\}$):

$$\neg\mathsf{eq}(k,n) \;\rightarrow\; \langle y, i, j, \mathsf{s}(k), n, z\rangle \prec \langle b, i, j, k, n, h\rangle \tag{259}$$

2. Inductive Evaluation w.r.t. $\mathsf{eq}(k,n)$, Symbolic Evaluation:

$$\mathsf{true} \;\rightarrow\; \langle y, i, j, \mathsf{s}(0), \mathsf{s}(v), z\rangle \prec \langle b, i, j, 0, \mathsf{s}(v), h\rangle \tag{260}$$
$$\mathsf{true} \;\rightarrow\; \langle y, i, j, \mathsf{s}(\mathsf{s}(u)), 0, z\rangle \prec \langle b, i, j, \mathsf{s}(u), 0, h\rangle \tag{261}$$
$$\begin{aligned}\neg\mathsf{eq}(u,v) \;\rightarrow\; & \langle y, i, j, \mathsf{s}(\mathsf{s}(u)), \mathsf{s}(v), z\rangle + \langle b, i, j, u, v, h\rangle \preceq \\ & \langle b, i, j, \mathsf{s}(u), \mathsf{s}(v), h\rangle + \langle y, i, j, \mathsf{s}(u), v, z\rangle\end{aligned} \tag{262}$$

3. Premise Elimination in (260) and (262), Rejection in (261)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \langle b, i, j, k, n, h\rangle \mapsto (n - k)^2$

Soundness Predicate

function $\lambda_{(259)}$: list × nat × nat × nat × nat × nat → bool

$$\begin{aligned}
\lambda_{(259)}(b, i, j, 0, 0, h, y, z) &= \mathsf{true}\\
\lambda_{(259)}(b, i, j, 0, \mathsf{s}(v), h, y, z) &= \mathsf{true}\\
\lambda_{(259)}(b, i, j, \mathsf{s}(u), 0, h, y, z) &= \mathsf{false}\\
\lambda_{(259)}(b, i, j, \mathsf{s}(u), \mathsf{s}(v), h, y, z) &= \lambda_{(259)}(b, i, j, u, v, h, y, z)
\end{aligned}$$

$\lambda_{(259)}(b, i, j, k, n, h, y, z) = \mathsf{true}$ iff $k \leq n$

Termination Predicates

function θ_{loop} : list × nat × nat × nat × nat × nat → bool

```
$\theta_{\mathsf{loop}}(b,i,j,k,n,h) = \mathsf{if}(\neg\mathsf{eq}(k,n),$
    $\lambda_{(259)}(b,i,j,k,n,h,\mathsf{set}(\ldots),\mathsf{nth}(\ldots)) \wedge$
      $\theta_{\mathsf{loop}}(\mathsf{set}(\mathsf{plus}(j,k),$
          $\mathsf{set}(\mathsf{plus}(i,k),b,\mathsf{nth}(\mathsf{plus}(j,k),b)),$
          $\mathsf{nth}(\mathsf{plus}(i,k),b)),$
        $i,$
        $j,$
        $\mathsf{s}(k),$
        $n,$
        $\mathsf{nth}(\mathsf{plus}(i,k),b)),$
    $\mathsf{true})$
```

$\theta_{\mathsf{loop}}(b,i,j,k,n,h) = \mathsf{true}$ iff $k \le n$.

function θ_{main} : nat × list → bool

$\theta_{\mathsf{main}}(b,i,j,n,h) = \theta_{\mathsf{loop}}(b,i,j,0,n,h)$

$\theta_{\mathsf{main}}(n,b) = \mathsf{true}$.

49 horner ([Gri81] p. 242)

Given a natural x and an n-element list a, the following imperative procedure computes in y the value of $a[0]*x^0+\ldots+a[n-1]x^{n-1}$ using Horner's rule. It terminates iff $n \neq 0$.

```
procedure horner($a$: list, $n, i, x, y, z$: nat)
  $i := 1$;  $y := a[0]$;  $z := x$;
  while $i \neq n$ do
    $y := y + a[i] * z$;  $z := z * x$;  $i := i + 1$
  od
```

Translation

function loop : list × nat × nat × nat × nat × nat → nat

```
$\mathsf{loop}(a,n,i,x,y,z) = \mathsf{if}(\neg\mathsf{eq}(i,n),$
    $\mathsf{loop}(a,n,\mathsf{s}(i),x,\mathsf{plus}(y,\mathsf{times}(\mathsf{nth}(i,a),z)),\mathsf{times}(z,x)),$
    $y)$
```

function main : list × nat × nat → nat

$\mathsf{main}(a,n,x) = \mathsf{loop}(a,n,\mathsf{s}(0),x,\mathsf{nth}(0,a),x)$

Termination Hypothesis

$$\neg\mathsf{eq}(i,n) \;\rightarrow\; \langle a,n,\mathsf{s}(i),x,\mathsf{plus}(y,\mathsf{times}(\mathsf{nth}(i,a),z)),\mathsf{times}(z,x)\rangle \prec \langle a,n,i,x,y,z\rangle$$

Transformation

1. Generalization $\{\mathsf{plus}(y,\mathsf{times}(\mathsf{nth}(i,a),z))/q, \mathsf{times}(z,x)/r\}$ (since $y,z \notin \{i,n\}$):

$$\neg\mathsf{eq}(i,n) \;\rightarrow\; \langle a,n,\mathsf{s}(i),x,q,r\rangle \prec \langle a,n,i,x,y,z\rangle \tag{263}$$

2. Inductive Evaluation w.r.t. $\mathsf{eq}(i,n)$, Symbolic Evaluation:

$$\mathsf{true} \;\rightarrow\; \langle a,\mathsf{s}(v),\mathsf{s}(0),x,q,r\rangle \prec \langle a,\mathsf{s}(v),0,x,y,z\rangle \tag{264}$$

$$\mathsf{true} \;\rightarrow\; \langle a,0,\mathsf{s}(\mathsf{s}(u)),x,q,r\rangle \prec \langle a,0,\mathsf{s}(u),x,y,z\rangle \tag{265}$$

$$\begin{aligned}\neg\mathsf{eq}(u,v) \;\rightarrow\; &\langle a,\mathsf{s}(v),\mathsf{s}(\mathsf{s}(u)),x,q,r\rangle + \langle a,v,u,x,y,z\rangle \preceq\\ &\langle a,\mathsf{s}(v),\mathsf{s}(u),x,y,z\rangle + \langle a,v,\mathsf{s}(u),x,q,r\rangle\end{aligned} \tag{266}$$

3. Premise Elimination in (264) and (266), Rejection in (265)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle a,n,i,x,y,z\rangle \mapsto (n-i)^2$

Soundness Predicate

```
function $\lambda_{(263)}$ : $\mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
  $\lambda_{(263)}(a, 0, 0, x, y, z, q, r) = \mathsf{true}$
  $\lambda_{(263)}(a, \mathsf{s}(v), 0, x, y, z, q, r) = \mathsf{true}$
  $\lambda_{(263)}(a, 0, \mathsf{s}(u), x, y, z, q, r) = \mathsf{false}$
  $\lambda_{(263)}(a, \mathsf{s}(v), \mathsf{s}(u), x, y, z, q, r) = \lambda_{(263)}(a, v, u, x, y, z, q, r)$
```

$\lambda_{(263)}(a,n,i,x,y,z,q,r) = \mathsf{true}$ iff $i \le n$

Termination Predicates

```
function $\theta_{\mathsf{loop}}$ : $\mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
  $\theta_{\mathsf{loop}}(a, n, i, x, y, z) = \mathsf{if}(\neg\mathsf{eq}(i, n),$
                              $\lambda_{(263)}(a, n, i, x, y, z, \mathsf{plus}(y, \mathsf{times}(\mathsf{nth}(i, a), z)), \mathsf{times}(z, x))$
                                $\wedge\ \theta_{\mathsf{loop}}(a, n, \mathsf{s}(i), x, \mathsf{plus}(y, \mathsf{times}(\mathsf{nth}(i, a), z)), \mathsf{times}(z, x)),$
                              $\mathsf{true})$
```

$\theta_{\mathsf{loop}}(a,n,i,x,y,z) = \mathsf{true}$ iff $i \le n$.

```
function $\theta_{\mathsf{main}}$ : $\mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
  $\theta_{\mathsf{main}}(a, n, x) = \theta_{\mathsf{loop}}(a, n, \mathsf{s}(0), x, \mathsf{nth}(0, a), x)$
```

$\theta_{\mathsf{main}}(a,n,x) = \mathsf{true}$ iff $n \neq 0$.

50 perm_to_code ([Gri81] p. 272)

Let the list x contain a permutation of the naturals $0, 1, .., n-1$. Let *code*(i) be the number of values among the first i elements of x that are less than $x[i]$. For example, given the array $x = [2, 0, 3, 1]$ we have *code*$(0) =$ *code*$(1) = 0$, *code*$(2) = 2$, and *code*$(3) = 1$. The imperative procedure perm_to_code changes x such that it finally contains the corresponding *code* values.

```
procedure perm_to_code($x$ : list, $j, k, n$ : nat)
  $k := n$;
  while $k \neq 0$ do                                    % loop2
    $k := k - 1$;  $j := 0$;
    while $j \neq k$ do                                  % loop1
      if $x[k] < x[j]$ then $x[j] := x[j] - 1$ else skip fi;
      $j := j + 1$
    od;
  od
```

Translation

```
function loop1 : $\mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{list}$
  $\mathsf{loop1}(x, j, k) = \mathsf{if}(\neg\mathsf{eq}(j, k),$
                    $\mathsf{if}(\mathsf{lt}(\mathsf{nth}(k, x), \mathsf{nth}(j, x)),$
                      $\mathsf{loop1}(\mathsf{set}(j, x, \mathsf{p}(\mathsf{nth}(j, x))), \mathsf{s}(j), k),$
                      $\mathsf{loop1}(x, \mathsf{s}(j), k)),$
                    $x)$

function loop2 : $\mathsf{list} \times \mathsf{nat} \to \mathsf{list}$
  $\mathsf{loop2}(x, 0) = x$
  $\mathsf{loop2}(x, \mathsf{s}(u)) = \mathsf{loop2}(\mathsf{loop1}(x, 0, u), u)$
```

function $\mathsf{main} : \mathsf{list} \times \mathsf{nat} \to \mathsf{list}$
$\mathsf{main}(x, n) = \mathsf{loop2}(x, n)$

loop1:

Termination Hypotheses

$$\neg\mathsf{eq}(j,k) \wedge \mathsf{lt}(\mathsf{nth}(k,x), \mathsf{nth}(j,x)) \to \langle \mathsf{set}(j, x, \mathsf{p}(\mathsf{nth}(j,x))), \mathsf{s}(j), k\rangle \prec \langle x, j, k\rangle \quad (267)$$
$$\neg\mathsf{eq}(j,k) \wedge \neg\mathsf{lt}(\mathsf{nth}(k,x), \mathsf{nth}(j,x)) \to \langle x, \mathsf{s}(j), k\rangle \prec \langle x, j, k\rangle \quad (268)$$

Transformation

1. In (267), Generalization $\{\mathsf{set}(j, x, \mathsf{p}(\mathsf{nth}(j,x)))/y\}$ (since $x \notin \{j, k\}$) and Inverse Weakening eliminating $\mathsf{lt}(\mathsf{nth}(k,x), \mathsf{nth}(j,x))$:

$$\neg\mathsf{eq}(j,k) \to \langle y, \mathsf{s}(j), k\rangle \prec \langle x, j, k\rangle \quad (269)$$

2. In (269), Inductive Evaluation w.r.t. $\mathsf{eq}(j,k)$, Symbolic Evaluation:

$$\mathsf{true} \to \langle y, \mathsf{s}(0), \mathsf{s}(v)\rangle \prec \langle x, 0, \mathsf{s}(v)\rangle \quad (270)$$
$$\mathsf{true} \to \langle y, \mathsf{s}(\mathsf{s}(u)), 0\rangle \prec \langle x, \mathsf{s}(u), 0\rangle \quad (271)$$
$$\neg\mathsf{eq}(u,v) \to \langle y, \mathsf{s}(\mathsf{s}(u)), \mathsf{s}(v)\rangle + \langle x, u, v\rangle \preceq \langle x, \mathsf{s}(u), \mathsf{s}(v)\rangle + \langle y, \mathsf{s}(u), v\rangle \quad (272)$$

3. In (268), Inverse Weakening eliminating $\neg\mathsf{lt}(\mathsf{nth}(k,x), \mathsf{nth}(j,x))$:

$$\neg\mathsf{eq}(j,k) \to \langle x, \mathsf{s}(j), k\rangle \prec \langle x, j, k\rangle \quad (273)$$

4. In (273), Inductive Evaluation w.r.t. $\mathsf{eq}(j,k)$, Symbolic Evaluation:

$$\mathsf{true} \to \langle x, \mathsf{s}(0), \mathsf{s}(v)\rangle \prec \langle x, 0, \mathsf{s}(v)\rangle \quad (274)$$
$$\mathsf{true} \to \langle x, \mathsf{s}(\mathsf{s}(u)), 0\rangle \prec \langle x, \mathsf{s}(u), 0\rangle \quad (275)$$
$$\neg\mathsf{eq}(u,v) \to \langle x, \mathsf{s}(\mathsf{s}(u)), \mathsf{s}(v)\rangle + \langle x, u, v\rangle \preceq \langle x, \mathsf{s}(u), \mathsf{s}(v)\rangle + \langle x, \mathsf{s}(u), v\rangle \quad (276)$$

5. Premise Elimination in (270), (272), (274), and (276), Rejection in (271) and (275)

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle x, j, k\rangle \mapsto (k-j)^2$

Soundness Predicates

function $\lambda_{(269)} : \mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$
$\lambda_{(269)}(x, 0, 0, y) = \mathsf{true}$
$\lambda_{(269)}(x, 0, \mathsf{s}(v), y) = \mathsf{true}$
$\lambda_{(269)}(x, \mathsf{s}(u), 0, y) = \mathsf{false}$
$\lambda_{(269)}(x, \mathsf{s}(u), \mathsf{s}(v), y) = \lambda_{(269)}(x, u, v, y)$

$\lambda_{(269)}(x, j, k, y) = \mathsf{true}$ iff $j \le k$

function $\lambda_{(273)} : \mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$
$\lambda_{(273)}(x, 0, 0) = \mathsf{true}$
$\lambda_{(273)}(x, 0, \mathsf{s}(v)) = \mathsf{true}$
$\lambda_{(273)}(x, \mathsf{s}(u), 0) = \mathsf{false}$
$\lambda_{(273)}(x, \mathsf{s}(u), \mathsf{s}(v)) = \lambda_{(273)}(x, u, v)$

$\lambda_{(273)}(x, j, k) = \mathsf{true}$ iff $j \le k$

Termination Predicate

```
function $\theta_{\mathsf{loop1}}$ : list × nat × nat → bool
  $\theta_{\mathsf{loop1}}(x, j, k) = \mathsf{if}(\neg\mathsf{eq}(j, k),$
                    $\mathsf{if}(\mathsf{lt}(\mathsf{nth}(k, x), \mathsf{nth}(j, x)),$
                        $\lambda_{(269)}(x, j, k, \mathsf{set}(j, x, \mathsf{p}(\mathsf{nth}(j, x)))) \wedge$
                            $\theta_{\mathsf{loop1}}(\mathsf{set}(j, x, \mathsf{p}(\mathsf{nth}(j, x))), \mathsf{s}(j), k),$
                        $\lambda_{(273)}(x, j, k) \wedge \theta_{\mathsf{loop1}}(x, \mathsf{s}(j), k)),$
                    $\mathsf{true})$
```

$\theta_{\mathsf{loop1}}(x, j, k) = \mathsf{true}$ iff $j \leq k$.

loop2:

Termination Hypothesis

$$\langle \mathsf{loop1}(x, 0, u), u\rangle \prec \langle x, \mathsf{s}(u)\rangle \tag{277}$$

Transformation

1. Generalization $\{\mathsf{loop1}(x, 0, u)/y\}$ (since $x \notin \{u\}$):

$$\langle y, u\rangle \prec \langle x, \mathsf{s}(u)\rangle \tag{278}$$

2. Premise Elimination in (278)

Polynomial Ordering
$0 \mapsto 0, \ \mathsf{s}(u) \mapsto u + 1, \ \langle x, u\rangle \mapsto u$

Soundness Predicate

```
function $\lambda_{(278)}$ : nat × nat × nat → bool
  $\lambda_{(278)}(y, x, u) = \mathsf{true}$
```

Termination Predicate

```
function $\theta_{\mathsf{loop2}}$ : list × nat → bool
  $\theta_{\mathsf{loop2}}(x, 0)$      $= \mathsf{true}$
  $\theta_{\mathsf{loop2}}(x, \mathsf{s}(u)) = \theta_{\mathsf{loop1}}(x, 0, u) \wedge \theta_{\mathsf{loop2}}(\mathsf{loop1}(x, 0, u), u)$
```

$\theta_{\mathsf{loop2}}(x, u) = \mathsf{true}$.

main:

Termination Predicate

```
function $\theta_{\mathsf{main}}$ : list × nat → bool
  $\theta_{\mathsf{main}}(x, n) = \mathsf{true}$
```

51 code_to_perm ([Gri81] p. 274)

Let the list x contain the *code* values of a permutation of $0, 1, .., n-1$, cf. Example 50. Then the imperative procedure code_to_perm rebuilds in x the associated permutation.

```
procedure code_to_perm($x$ : list, $j, k, n$ : nat)
  $k := 0$;
  while $k \neq n$ do                                   % loop2
     $j := k$;
     while $j \neq 0$ do                                % loop1
        $j := j - 1$;
        if $x[j] < x[k]$ then skip else $x[j] := x[j] + 1$ fi
     od;
     $k := k + 1$
  od
```

Translation

function $\mathsf{loop1}$: $\mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{list}$

$$\begin{aligned}\mathsf{loop1}(x, 0, k) &= x \\ \mathsf{loop1}(x, \mathsf{s}(u), k) &= \mathsf{if}(\,\mathsf{lt}(\mathsf{nth}(u, x), \mathsf{nth}(k, x)), \\ &\qquad \mathsf{loop1}(x, u, k), \\ &\qquad \mathsf{loop1}(\mathsf{set}(u, x, \mathsf{s}(\mathsf{nth}(u, x))), u, k))\end{aligned}$$

function $\mathsf{loop2}$: $\mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{list}$

$$\mathsf{loop2}(x, k, n) = \mathsf{if}(\neg\mathsf{eq}(k, n),\ \mathsf{loop2}(\mathsf{loop1}(x, k, k), \mathsf{s}(k), n),\ x)$$

function main : $\mathsf{list} \times \mathsf{nat} \to \mathsf{list}$

$$\mathsf{main}(x, n) = \mathsf{loop2}(x, 0, n)$$

loop1:

Termination Hypothesis

$$\mathsf{lt}(\mathsf{nth}(u, x), \mathsf{nth}(k, x)) \;\to\; \langle x, u, k\rangle \prec \langle x, \mathsf{s}(u), k\rangle \tag{279}$$

$$\neg\mathsf{lt}(\mathsf{nth}(u, x), \mathsf{nth}(k, x)) \;\to\; \langle \mathsf{set}(u, x, \mathsf{s}(\mathsf{nth}(u, x))), u, k\rangle \prec \langle x, \mathsf{s}(u), k\rangle \tag{280}$$

Transformation

1. In (280), Generalization $\{\mathsf{set}(u, x, \mathsf{s}(\mathsf{nth}(u, x)))/y\}$ (since $x \notin \{u\}$):

$$\neg\mathsf{lt}(\mathsf{nth}(u, x), \mathsf{nth}(k, x)) \;\to\; \langle y, u, k\rangle \prec \langle x, \mathsf{s}(u), k\rangle \tag{281}$$

2. Premise Elimination in (280) and (281)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \langle x, j, k\rangle \mapsto j$

Soundness Predicates

function $\lambda_{(279)}$: $\mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$\lambda_{(279)}(x, u, k) = \mathsf{true}$

function $\lambda_{(281)}$: $\mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{list} \to \mathsf{bool}$

$\lambda_{(281)}(x, u, k, y) = \mathsf{true}$

Termination Predicate

function θ_{loop1} : $\mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{list}$

$\theta_{\mathsf{loop1}}(x, j, k) = \mathsf{true}$

loop2:

Termination Hypothesis

$$\neg\mathsf{eq}(k, n) \;\to\; \langle \mathsf{loop1}(x, k, k), \mathsf{s}(k), n\rangle \prec \langle x, k, n\rangle$$

Transformation

1. Generalization $\{\mathsf{loop1}(x,k,k)/y\}$ (since $x \notin \{k,n\}$):

$$\neg\mathsf{eq}(k,n) \;\rightarrow\; \langle y, \mathsf{s}(k), n\rangle \prec \langle x,k,n\rangle \tag{282}$$

2. Inductive Evaluation w.r.t. $\mathsf{eq}(k,n)$, Symbolic Evaluation:

$$\mathsf{true} \;\rightarrow\; \langle y, \mathsf{s}(0), \mathsf{s}(v)\rangle \prec \langle x, 0, \mathsf{s}(v)\rangle \tag{283}$$
$$\mathsf{true} \;\rightarrow\; \langle y, \mathsf{s}(\mathsf{s}(u)), 0\rangle \prec \langle x, \mathsf{s}(u), 0\rangle \tag{284}$$
$$\neg\mathsf{eq}(u,v) \;\rightarrow\; \langle y, \mathsf{s}(\mathsf{s}(u)), \mathsf{s}(v)\rangle + \langle x,u,v\rangle \preceq \langle x, \mathsf{s}(u), \mathsf{s}(v)\rangle + \langle y, \mathsf{s}(u), v\rangle \tag{285}$$

3. Premise Elimination in (283) and (285), Rejection in (284)

Polynomial Ordering
$0 \mapsto 0,\;\; \mathsf{s}(v) \mapsto v+1,\;\; \langle x,k,n\rangle \mapsto (n-k)^2$

Soundness Predicate

```
function $\lambda_{(282)}$ : list × nat × nat × list → bool
  $\lambda_{(282)}(x, 0, 0, y)$          $= \mathsf{true}$
  $\lambda_{(282)}(x, 0, \mathsf{s}(v), y)$      $= \mathsf{true}$
  $\lambda_{(282)}(x, \mathsf{s}(u), 0, y)$      $= \mathsf{false}$
  $\lambda_{(282)}(x, \mathsf{s}(u), \mathsf{s}(v), y)$ $= \lambda_{(282)}(x,u,v,y)$
```

$\lambda_{(282)}(x,k,n,y) = \mathsf{true}$ iff $k \le n$.

Termination Predicate

```
function $\theta_{\mathsf{loop2}}$ : list × nat × nat → bool
  $\theta_{\mathsf{loop2}}(x,k,n) = \mathsf{if}(\neg\mathsf{eq}(k,n),$
                      $\lambda_{(282)}(x,k,n,\mathsf{loop1}(x,k,k)) \wedge \theta_{\mathsf{loop2}}(\mathsf{loop1}(x,k,k), \mathsf{s}(k), n),$
                      $\mathsf{true})$
```

$\theta_{\mathsf{loop2}}(x,k,n) = \mathsf{true}$ iff $k \le n$.

main:

Termination Predicate

```
function $\theta_{\mathsf{main}}$ : list × nat → bool
  $\theta_{\mathsf{main}}(x,n) = \theta_{\mathsf{loop2}}(x,0,n)$
```

$\theta_{\mathsf{main}}(x,n) = \mathsf{true}$.

52 list_min ([Gri81] p. 343)

The imperative procedure list_min stores in x the minimum value of the n-element list b. It terminates iff $n \neq 0$.

```
procedure list_min($n, i, x$ : nat, $b$ : list)
  $i := 1;\;\; x := b[0];$
  while $i \neq n$ do
     if $b[i] \le x$ then
        $x := b[i];\;\; i := i+1$
     else
        $i := i+1$
     fi;
  od
```

Translation

```
function loop : nat × nat × nat × list → nat
  loop(n, i, x, b) = if( ¬eq(i, n),
                         if( le(nth(i, b), x),
                             loop(n, s(i), nth(i, b), b),
                             loop(n, s(i), x, b)),
                         x)

function main : nat × list → nat
  main(n, b) = loop(n, s(0), nth(0, b), b)
```

Termination Hypotheses

$$\neg\mathsf{eq}(i,n) \wedge \ \mathsf{le}(\mathsf{nth}(i,b),x) \ \rightarrow \ \langle n, \mathsf{s}(i), \mathsf{nth}(i,b), b\rangle \prec \langle n,i,x,b\rangle \tag{286}$$

$$\neg\mathsf{eq}(i,n) \wedge \neg\mathsf{le}(\mathsf{nth}(i,b),x) \ \rightarrow \ \langle n, \mathsf{s}(i), x, b\rangle \prec \langle n,i,x,b\rangle \tag{287}$$

Transformation

1. In (286), Generalization $\{\mathsf{nth}(i,b)/y\}$ (since $x \notin \{i,n\}$) and Inverse Weakening eliminating $\mathsf{le}(\mathsf{nth}(i,b),x)$:

$$\neg\mathsf{eq}(i,n) \ \rightarrow \ \langle n,\mathsf{s}(i),y,b\rangle \prec \langle n,i,x,b\rangle \tag{288}$$

2. In (288), Inductive Evaluation w.r.t. $\mathsf{eq}(i,n)$, Symbolic Evaluation:

$$\mathsf{true} \ \rightarrow \ \langle \mathsf{s}(v),\mathsf{s}(0),y,b\rangle \prec \langle \mathsf{s}(v),0,x,b\rangle \tag{289}$$

$$\mathsf{true} \ \rightarrow \ \langle 0,\mathsf{s}(\mathsf{s}(u)),y,b\rangle \prec \langle 0,\mathsf{s}(u),x,b\rangle \tag{290}$$

$$\neg\mathsf{eq}(u,v) \ \rightarrow \ \langle \mathsf{s}(v),\mathsf{s}(\mathsf{s}(u)),y,b\rangle + \langle v,u,x,b\rangle \preceq \langle \mathsf{s}(v),\mathsf{s}(u),x,b\rangle + \langle v,\mathsf{s}(u),y,b\rangle \tag{291}$$

3. In (287), Inverse Weakening eliminating $\neg\mathsf{le}(\mathsf{nth}(i,b),x)$:

$$\neg\mathsf{eq}(i,n) \ \rightarrow \ \langle n,\mathsf{s}(i),x,b\rangle \prec \langle n,i,x,b\rangle \tag{292}$$

4. In (292), Inductive Evaluation w.r.t. $\mathsf{eq}(i,n)$, Symbolic Evaluation:

$$\mathsf{true} \ \rightarrow \ \langle \mathsf{s}(v),\mathsf{s}(0),x,b\rangle \prec \langle \mathsf{s}(v),0,x,b\rangle \tag{293}$$

$$\mathsf{true} \ \rightarrow \ \langle 0,\mathsf{s}(\mathsf{s}(u)),x,b\rangle \prec \langle 0,\mathsf{s}(u),x,b\rangle \tag{294}$$

$$\neg\mathsf{eq}(u,v) \ \rightarrow \ \langle \mathsf{s}(v),\mathsf{s}(\mathsf{s}(u)),x,b\rangle + \langle v,u,x,b\rangle \preceq \langle \mathsf{s}(v),\mathsf{s}(u),x,b\rangle + \langle v,\mathsf{s}(u),x,b\rangle \tag{295}$$

5. Premise Elimination in (289), (291), (293) and, (295), Rejection in (290) and (294)

Polynomial Ordering

$0 \mapsto 0, \ \mathsf{s}(v) \mapsto v+1, \ \langle n,i,x,b\rangle \mapsto (n-i)^2$

Soundness Predicates

function $\lambda_{(288)}$: nat × nat × nat × list × nat → bool

$\lambda_{(288)}(0,0,x,b,y) = \mathsf{true}$
$\lambda_{(288)}(\mathsf{s}(v),0,x,b,y) = \mathsf{true}$
$\lambda_{(288)}(0,\mathsf{s}(u),x,b,y) = \mathsf{false}$
$\lambda_{(288)}(\mathsf{s}(v),\mathsf{s}(u),x,b,y) = \lambda_{(288)}(v,u,x,b,y)$

$\lambda_{(288)}(n,i,x,b,y) = \mathsf{true}$ iff $i \leq n$

function $\lambda_{(292)}$: nat × nat × nat × list → bool

$\lambda_{(292)}(0,0,x,b) = \mathsf{true}$
$\lambda_{(292)}(\mathsf{s}(v),0,x,b) = \mathsf{true}$
$\lambda_{(292)}(0,\mathsf{s}(u),x,b) = \mathsf{false}$
$\lambda_{(292)}(\mathsf{s}(v),\mathsf{s}(u),x,b) = \lambda_{(292)}(v,u,x,b)$

$\lambda_{(292)}(n,i,x,b) = \mathsf{true}$ iff $i \le n$

Termination Predicates

```
function $\theta_{\mathsf{loop}}$ : nat × nat × nat × list → bool
  $\theta_{\mathsf{loop}}(n,i,x,b)$ = if( ¬eq(i, n),
                            if( le(nth(i, b), x),
                               $\lambda_{(288)}(n,i,x,b,\mathsf{nth}(i,b)) \mathbin{\hat\wedge} \theta_{\mathsf{loop}}(n,\mathsf{s}(i),\mathsf{nth}(i,b),b)$,
                               $\lambda_{(292)}(n,i,x,b) \mathbin{\hat\wedge} \theta_{\mathsf{loop}}(n,\mathsf{s}(i),x,b)$),
                            true)
```

$\theta_{\mathsf{loop}}(n,i,x,b) = \mathsf{true}$ iff $i \le n$.

```
function $\theta_{\mathsf{main}}$ : nat × list → bool
  $\theta_{\mathsf{main}}(n,b) = \theta_{\mathsf{loop}}(n,\mathsf{s}(0),\mathsf{nth}(0,b),b)$
```

$\theta_{\mathsf{main}}(n,b) = \mathsf{true}$ iff $n \ne 0$.

53 llist_search2 ([Gri81] p. 344)

Given a natural x and a list b containing lists of length n, the following imperative procedure determines the least indices i and j such that $b[i,j] = x$. The procedure terminates iff $x = 0$ or $b[i,j] = x$ for some i,j satisfying $0 \le i < \mathsf{len}(b)$ and $0 \le j < n$.

```
procedure llist_search2(b: llist, n, x, i, j: nat)
  $i := 0$;  $j := 0$;
  while $x \ne b[i,j]$ do
    if $j = n - 1$ then
      $j := 0$;  $i := i + 1$;
    else
      $j := j + 1$
    fi
  od
```

Translation

```
function loop : llist × nat × nat × nat × nat → nat
  loop(b, n, x, i, j) = if( ¬eq(x, lnth(i, j, b)),
                            if( eq(j, p(n)),
                               loop(b, n, x, s(i), 0),
                               loop(b, n, x, i, s(j)))
                            i)

function main : llist × nat × nat → nat
  main(b, n, x) = loop(b, n, x, 0, 0)
```

Termination Hypotheses

$$\neg\mathsf{eq}(x,\mathsf{lnth}(i,j,b)) \wedge \mathsf{eq}(j,\mathsf{p}(n)) \rightarrow \langle b,n,x,\mathsf{s}(i),0\rangle \prec \langle b,n,x,i,j\rangle \tag{296}$$

$$\neg\mathsf{eq}(x,\mathsf{lnth}(i,j,b)) \wedge \neg\mathsf{eq}(j,\mathsf{p}(n)) \rightarrow \langle b,n,x,i,\mathsf{s}(j)\rangle \prec \langle b,n,x,i,j\rangle \tag{297}$$

Transformation

1. In (296), Inductive Evaluation w.r.t. $\mathsf{lnth}(i,j,b)$, Symbolic Evaluation:

$$\neg\mathsf{eq}(x,0) \wedge \mathsf{eq}(j,\mathsf{p}(n)) \rightarrow \langle \mathsf{lempty},n,x,\mathsf{s}(i),0\rangle \prec \langle \mathsf{lempty},n,x,i,j\rangle \tag{298}$$

$$\neg\mathsf{eq}(x,\mathsf{nth}(j,v)) \wedge \mathsf{eq}(j,\mathsf{p}(n)) \rightarrow \langle \mathsf{ladd}(v,w),n,x,\mathsf{s}(0),0\rangle \prec \langle \mathsf{ladd}(v,w),n,x,0,j\rangle \tag{299}$$

$$\neg\mathsf{eq}(x,\mathsf{lnth}(u,j,w)) \wedge \mathsf{eq}(j,\mathsf{p}(n)) \rightarrow \langle \mathsf{ladd}(v,w),n,x,\mathsf{s}(\mathsf{s}(u)),0\rangle + \langle w,n,x,u,j\rangle \preceq \langle \mathsf{ladd}(v,w),n,x,\mathsf{s}(u),j\rangle + \langle w,n,x,\mathsf{s}(u),0\rangle \tag{300}$$

2. In (299), Inductive Evaluation w.r.t. $\mathsf{p}(n)$, Symbolic Evaluation:

$$\neg\mathsf{eq}(x,\mathsf{nth}(j,v)) \wedge \mathsf{eq}(j,0) \;\rightarrow\; \langle \mathsf{ladd}(v,w),0,x,\mathsf{s}(0),0\rangle \prec \langle \mathsf{ladd}(v,w),0,x,0,j\rangle \qquad (301)$$
$$\neg\mathsf{eq}(x,\mathsf{nth}(j,v)) \wedge \mathsf{eq}(j,z) \;\rightarrow\; \langle \mathsf{ladd}(v,w),\mathsf{s}(z),x,\mathsf{s}(0),0\rangle \prec \langle \mathsf{ladd}(v,w),\mathsf{s}(z),x,0,j\rangle \qquad (302)$$

3. In (301), Replace w.r.t. $\mathsf{eq}(j,0)$:

$$\neg\mathsf{eq}(x,\mathsf{nth}(j,v)) \;\rightarrow\; \langle \mathsf{ladd}(v,w),0,x,\mathsf{s}(0),0\rangle \prec \langle \mathsf{ladd}(v,w),0,x,0,0\rangle \qquad (303)$$

4. In (302), Replace w.r.t. $\mathsf{eq}(j,z)$:

$$\neg\mathsf{eq}(x,\mathsf{nth}(j,v)) \;\rightarrow\; \langle \mathsf{ladd}(v,w),\mathsf{s}(j),x,\mathsf{s}(0),0\rangle \prec \langle \mathsf{ladd}(v,w),\mathsf{s}(j),x,0,j\rangle \qquad (304)$$

5. In (297), Inductive Evaluation w.r.t. $\mathsf{lnth}(i,j,b)$, Symbolic Evaluation:

$$\neg\mathsf{eq}(x,0) \wedge \neg\mathsf{eq}(j,\mathsf{p}(n)) \;\rightarrow\; \langle \mathsf{lempty},n,x,i,\mathsf{s}(j)\rangle \prec \langle \mathsf{lempty},n,x,i,j\rangle \qquad (305)$$
$$\neg\mathsf{eq}(x,\mathsf{nth}(j,v)) \wedge \neg\mathsf{eq}(j,\mathsf{p}(n)) \;\rightarrow\; \langle \mathsf{ladd}(v,w),n,x,0,\mathsf{s}(j)\rangle \prec \langle \mathsf{ladd}(v,w),n,x,0,j\rangle \qquad (306)$$
$$\neg\mathsf{eq}(x,\mathsf{lnth}(u,j,w)) \wedge \neg\mathsf{eq}(j,\mathsf{p}(n)) \;\rightarrow\; \langle \mathsf{ladd}(v,w),n,x,\mathsf{s}(u),\mathsf{s}(j)\rangle + \langle w,n,x,u,j\rangle \preceq \langle \mathsf{ladd}(v,w),n,x,\mathsf{s}(u),j\rangle + \langle w,n,x,u,\mathsf{s}(j)\rangle \qquad (307)$$

6. In (306), Inductive Evaluation w.r.t. $\mathsf{p}(n)$, Symbolic Evaluation:

$$\neg\mathsf{eq}(x,\mathsf{nth}(j,v)) \wedge \neg\mathsf{eq}(j,0) \;\rightarrow\; \langle \mathsf{ladd}(v,w),0,x,0,\mathsf{s}(j)\rangle \prec \langle \mathsf{ladd}(v,w),0,x,0,j\rangle \qquad (308)$$
$$\neg\mathsf{eq}(x,\mathsf{nth}(j,v)) \wedge \neg\mathsf{eq}(j,z) \;\rightarrow\; \langle \mathsf{ladd}(v,w),\mathsf{s}(z),x,0,\mathsf{s}(j)\rangle \prec \langle \mathsf{ladd}(v,w),\mathsf{s}(z),x,0,j\rangle \qquad (309)$$

7. In (309), Inverse Weakening eliminating $\neg\mathsf{eq}(x,\mathsf{nth}(j,v))$:

$$\neg\mathsf{eq}(j,z) \;\rightarrow\; \langle \mathsf{ladd}(v,w),\mathsf{s}(z),x,0,\mathsf{s}(j)\rangle \prec \langle \mathsf{ladd}(v,w),\mathsf{s}(z),x,0,j\rangle \qquad (310)$$

8. In (310), Inductive Evaluation w.r.t. $\mathsf{eq}(j,z)$, Symbolic Evaluation:

$$\mathsf{true} \;\rightarrow\; \langle \mathsf{ladd}(v,w),\mathsf{s}(\mathsf{s}(r)),x,0,\mathsf{s}(0)\rangle \prec \langle \mathsf{ladd}(v,w),\mathsf{s}(\mathsf{s}(r)),x,0,0\rangle \qquad (311)$$
$$\mathsf{true} \;\rightarrow\; \langle \mathsf{ladd}(v,w),\mathsf{s}(0),x,0,\mathsf{s}(\mathsf{s}(q))\rangle \prec \langle \mathsf{ladd}(v,w),\mathsf{s}(0),x,0,\mathsf{s}(q)\rangle \qquad (312)$$
$$\neg\mathsf{eq}(q,r) \;\rightarrow\; \langle \mathsf{ladd}(v,w),\mathsf{s}(\mathsf{s}(r)),x,0,\mathsf{s}(\mathsf{s}(q))\rangle + \langle \mathsf{ladd}(v,w),\mathsf{s}(r),x,0,q\rangle \preceq \langle \mathsf{ladd}(v,w),\mathsf{s}(\mathsf{s}(r)),x,0,\mathsf{s}(q)\rangle + \langle \mathsf{ladd}(v,w),\mathsf{s}(r),x,0,\mathsf{s}(q)\rangle \qquad (313)$$

9. Premise Elimination in (300), (304), (307), (311), (313),
   Rejection in (298), (303), (305), (308), and (312)

Polynomial Ordering

$0 \mapsto 0$, $\mathsf{s}(v) \mapsto v+1$, $\mathsf{empty} \mapsto 0$, $\mathsf{add}(v,w) \mapsto w+1$, $\mathsf{lempty} \mapsto 0$,
$\mathsf{ladd}(v,w) \mapsto v+w+1$, $\langle b,n,x,i,j\rangle \mapsto (n(b-i)-j)^2$

Soundness Predicates

function $\lambda_{(299)} : \mathsf{list} \times \mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
$\lambda_{(299)}(v,w,0,x,j) = \mathsf{false}$
$\lambda_{(299)}(v,w,\mathsf{s}(z),x,j) = \mathsf{true}$

$\lambda_{(299)}(v,w,n,x,j) = \mathsf{true}$ iff $n \neq 0$

function $\lambda_{(296)} : \mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
$\lambda_{(296)}(\mathsf{lempty},n,x,i,j) = \mathsf{false}$
$\lambda_{(296)}(\mathsf{ladd}(v,w),n,x,0,j) = \lambda_{(299)}(v,w,n,x,j)$
$\lambda_{(296)}(\mathsf{ladd}(v,w),n,x,\mathsf{s}(u),j) = \lambda_{(296)}(w,n,x,u,j)$

$\lambda_{(296)}(b,n,x,i,j) = \mathsf{true}$ iff $i < \mathsf{len}(b) \wedge n \neq 0$

function $\lambda_{(310)}$: $\mathsf{list} \times \mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$$\begin{aligned}
&\lambda_{(310)}(v, w, 0, x, 0) &&= \mathsf{true}\\
&\lambda_{(310)}(v, w, \mathsf{s}(r), x, 0) &&= \mathsf{true}\\
&\lambda_{(310)}(v, w, 0, x, \mathsf{s}(q)) &&= \mathsf{false}\\
&\lambda_{(310)}(v, w, \mathsf{s}(r), x, \mathsf{s}(q)) &&= \lambda_{(310)}(l, w, r, x, q)
\end{aligned}$$

$\lambda_{(310)}(v, w, z, x, j) = \mathsf{true}$ iff $j \leq z$

function $\lambda_{(308)}$: $\mathsf{list} \times \mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$$\begin{aligned}
&\lambda_{(308)}(v, w, 0, x, j) &&= \mathsf{false}\\
&\lambda_{(308)}(v, w, \mathsf{s}(z), x, j) &&= \lambda_{(310)}(v, w, z, x, j)
\end{aligned}$$

$\lambda_{(308)}(v, w, n, x, j) = \mathsf{true}$ iff $j < n$

function $\lambda_{(297)}$: $\mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$$\begin{aligned}
&\lambda_{(297)}(\mathsf{lempty}, n, x, i, j) &&= \mathsf{false}\\
&\lambda_{(297)}(\mathsf{ladd}(v, w), n, x, 0, j) &&= \lambda_{(308)}(v, w, n, x, j)\\
&\lambda_{(297)}(\mathsf{ladd}(v, w), n, x, \mathsf{s}(u), j) &&= \lambda_{(297)}(w, n, x, u, j)
\end{aligned}$$

$\lambda_{(297)}(b, n, x, i, j) = \mathsf{true}$ iff $i < \mathsf{len}(b) \wedge j < n$

Termination Predicates

function θ_{loop} : $\mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

```
$\theta_{\mathsf{loop}}(b, n, x, i, j) = \mathsf{if}(\neg\mathsf{eq}(x, \mathsf{lnth}(i, j, b)),$
                          $\mathsf{if}(\mathsf{eq}(j, \mathsf{p}(n)),$
                             $\lambda_{(296)}(b, n, x, i, j) \wedge \theta_{\mathsf{loop}}(b, n, x, \mathsf{s}(i), 0),$
                             $\lambda_{(297)}(b, n, x, i, j) \wedge \theta_{\mathsf{loop}}(b, n, x, i, \mathsf{s}(j))),$
                          $\mathsf{true})$
```

$\theta_{\mathsf{loop}}(b, n, x, i, j) = \mathsf{true}$ iff $x = b[i, j]$ or
$x = 0$ or
$i < \mathsf{len}(b)$ and $b[i, l] = x$ for some l satisfying $j \leq l < n$ or
$b[k, l] = x$ for some k, l satisfying $i < k < \mathsf{len}(b)$ and $0 \leq l < n$

function θ_{main} : $\mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$\theta_{\mathsf{main}}(b, n, x) = \theta_{\mathsf{loop}}(b, n, x, 0, 0)$

$\theta_{\mathsf{main}}(b, n, x) = \mathsf{true}$ iff $x = 0$ or $b[i, j] = x$ for some i, j satisfying $0 \leq i < \mathsf{len}(b)$ and $0 \leq j < n$.

54 partition ([Gri81] p. 345)

Given naturals m and n, $m < n$, and a list b of a length greater than or equal to n, the following imperative procedure permutes the values $v_m, .., v_{n-1}$ given as $b[m], .., b[n-1]$ with respect to the first value v_m such that finally $v_m = b[r]$ for some r, $m \leq r \leq n-1$, and $b[i] \leq v_m$ for all i, $m \leq i < r$, and $v_m < b[i]$ for all i, $r < i \leq n-1$.

```
procedure partition($b$: list, $m, n, x, q, r, h$: nat)
  $x := b[m]$;  $q := m + 1$;  $r := n - 1$;
  while $q \leq r$ do
    if $b[q] \leq x$ then
      $q := q + 1$
    else
      if $x < b[r]$ then
        $r := r - 1$
      else
        $h := b[r]$;  $b[r] := b[q]$;  $b[q] := h$;
        $q := q + 1$;  $r := r - 1$
      fi
    fi
  od
  $h := b[m]$;  $b[m] := b[r]$;  $b[r] := h$
```

Translation

function loop : list × nat × nat × nat × nat × nat × nat → list

$$\begin{aligned}\mathsf{loop}(b,m,n,x,q,r,h) = \mathsf{if}(&\mathsf{le}(q,r),\\ &\mathsf{if}(\mathsf{le}(\mathsf{nth}(q,b),x),\\ &\quad \mathsf{loop}(b,m,n,x,\mathsf{s}(q),r,h),\\ &\quad \mathsf{if}(\mathsf{lt}(x,\mathsf{nth}(r,b))\\ &\qquad \mathsf{loop}(b,m,n,x,q,\mathsf{p}(r),h),\\ &\qquad \mathsf{loop}(\mathsf{set}(q,\mathsf{set}(r,b,\mathsf{nth}(q,b)),\mathsf{nth}(r,b)),\\ &\qquad\qquad m,n,x,\mathsf{s}(q),\mathsf{p}(r),\mathsf{nth}(r,b)))),\\ &b)\end{aligned}$$

function main : list × nat × nat × nat → list

$$\begin{aligned}\mathsf{main}(b,m,n,h) = \mathsf{set}(&r,\\ &\mathsf{set}(m,\\ &\quad \mathsf{loop}(b,m,n,\mathsf{nth}(m,b),\mathsf{s}(m),\mathsf{p}(n),h),\\ &\quad \mathsf{nth}(r,\mathsf{loop}(b,m,n,\mathsf{nth}(m,b),\mathsf{s}(m),\mathsf{p}(n),h))),\\ &\mathsf{nth}(m,\mathsf{loop}(b,m,n,\mathsf{nth}(m,b),\mathsf{s}(m),\mathsf{p}(n),h)))\end{aligned}$$

Termination Hypotheses

In this example we have to use a polynomial ordering where tuples of data objects may also be mapped to negative numbers.

$$\mathsf{le}(q,r) \wedge \ \mathsf{le}(\mathsf{nth}(q,b),x) \rightarrow \langle b,m,n,x,\mathsf{s}(q),r,h\rangle \prec \langle b,m,n,x,q,r,h\rangle \tag{314}$$

$$\mathsf{le}(q,r) \wedge \ \mathsf{le}(\mathsf{nth}(q,b),x) \rightarrow \mathsf{zero} \preceq \langle b,m,n,x,q,r,h\rangle \tag{315}$$

$$\mathsf{le}(q,r) \wedge \neg\mathsf{le}(\mathsf{nth}(q,b),x) \wedge \ \mathsf{lt}(x,\mathsf{nth}(r,b)) \rightarrow \langle b,m,n,x,q,\mathsf{p}(r),h\rangle \prec \langle b,m,n,x,q,r,h\rangle \tag{316}$$

$$\mathsf{le}(q,r) \wedge \neg\mathsf{le}(\mathsf{nth}(q,b),x) \wedge \ \mathsf{lt}(x,\mathsf{nth}(r,b)) \rightarrow \mathsf{zero} \preceq \langle b,m,n,x,q,r,h\rangle \tag{317}$$

$$\begin{aligned}&\mathsf{le}(q,r) \wedge \neg\mathsf{le}(\mathsf{nth}(q,b),x) \wedge \neg\mathsf{lt}(x,\mathsf{nth}(r,b)) \rightarrow\\ &\quad \langle \mathsf{set}(q,\mathsf{set}(r,b,\mathsf{nth}(q,b)),\mathsf{nth}(r,b)),m,n,x,\mathsf{s}(q),\mathsf{p}(r),\mathsf{nth}(r,b)\rangle \prec \langle b,m,n,x,q,r,h\rangle\end{aligned} \tag{318}$$

$$\mathsf{le}(q,r) \wedge \neg\mathsf{le}(\mathsf{nth}(q,b),x) \wedge \neg\mathsf{lt}(x,\mathsf{nth}(r,b)) \rightarrow \mathsf{zero} \preceq \langle b,m,n,x,q,r,h\rangle \tag{319}$$

Transformation

1. In (314), Inductive Evaluation is suggested, but not necessary.
2. In (315), Inverse Weakening eliminating $\mathsf{le}(\mathsf{nth}(q,b),x)$:

$$\mathsf{le}(q,r) \ \rightarrow \ \mathsf{zero} \preceq \langle b,m,n,x,q,r,h\rangle \tag{320}$$

3. In (320), Inductive Evaluation w.r.t. $\mathsf{le}(q,r)$, Symbolic Evaluation:

$$\mathsf{true} \ \rightarrow \ \mathsf{zero} \preceq \langle b,m,n,x,0,r,h\rangle \tag{321}$$

$$\mathsf{le}(u,v) \ \rightarrow \ \langle b,m,n,x,u,v,h\rangle \preceq \langle b,m,n,x,\mathsf{s}(u),\mathsf{s}(v),h\rangle \tag{322}$$

4. In (316), Estimation of p by an upper bound:

$$\langle b, m, n, x, q, \overline{\mathsf{p}}(r), h\rangle \preceq \langle b, m, n, x, q, r, h\rangle \tag{323}$$

$$u \prec v \rightarrow \langle b, m, n, x, q, u, h\rangle \prec \langle b, m, n, x, q, v, h\rangle \tag{324}$$

$$\mathcal{E}_{\mathsf{p}\preceq\overline{\mathsf{p}}}$$

$$\mathsf{le}(q,r) \wedge \neg\mathsf{le}(\mathsf{nth}(q,b),x) \wedge \mathsf{lt}(x,\mathsf{nth}(r,b)) \rightarrow \delta_{\mathsf{p}\prec\overline{\mathsf{p}}}(r) \vee \langle b, m, n, x, q, \overline{\mathsf{p}}(r), h\rangle \prec \langle b, m, n, x, q, r, h\rangle \tag{325}$$

5. Computing the estimation inequalities $\mathcal{E}_{\mathsf{p}\preceq\overline{\mathsf{p}}}$ and the strictness predicate $\delta_{\mathsf{p}\prec\overline{\mathsf{p}}}$:

$$0 \preceq \overline{\mathsf{p}}(0) \tag{326}$$

$$v \preceq \overline{\mathsf{p}}(\mathsf{s}(v)) \tag{327}$$

```
function $\delta_{\mathsf{p}\prec\overline{\mathsf{p}}}$ : nat → bool
  $\delta_{\mathsf{p}\prec\overline{\mathsf{p}}}(0) \quad = 0 \prec \overline{\mathsf{p}}(0)$
  $\delta_{\mathsf{p}\prec\overline{\mathsf{p}}}(\mathsf{s}(v)) = v \prec \overline{\mathsf{p}}(\mathsf{s}(v))$
```

6. In (325), Elimination of the strictness predicate call omitting the strict inequality and replacing (325) by the second inequality of $\delta_{\mathsf{p}\prec\overline{\mathsf{p}}}$:

$$v \prec \overline{\mathsf{p}}(\mathsf{s}(v)) \tag{328}$$

7. In (317), Inverse Weakening eliminating $\neg\mathsf{le}(\mathsf{nth}(q,b),x) \wedge \mathsf{lt}(x,\mathsf{nth}(r,b))$:

$$\mathsf{le}(q,r) \ \rightarrow\ \mathsf{zero} \preceq \langle b, m, n, x, q, r, h\rangle \tag{329}$$

8. In (329), Inductive Evaluation w.r.t. $\mathsf{le}(q,r)$, Symbolic Evaluation:

$$\mathsf{true} \ \rightarrow\ \mathsf{zero} \preceq \langle b, m, n, x, 0, r, h\rangle \tag{330}$$

$$\mathsf{le}(u,v) \ \rightarrow\ \langle b, m, n, x, u, v, h\rangle \preceq \langle b, m, n, x, \mathsf{s}(u), \mathsf{s}(v), h\rangle \tag{331}$$

9. In (318), Generalization $\{\mathsf{set}(q,\mathsf{set}(r,b,\mathsf{nth}(q,b)),\mathsf{nth}(r,b))/y,\ \mathsf{nth}(r,b)/z\}$ (since $b,h \notin \{q,r\}$):

$$\mathsf{le}(q,r) \wedge \neg\mathsf{le}(\mathsf{nth}(q,b),x) \wedge \neg\mathsf{lt}(x,\mathsf{nth}(r,b)) \ \rightarrow\ \langle y, m, n, x, \mathsf{s}(q), \mathsf{p}(r), z\rangle \prec \langle b, m, n, x, q, r, h\rangle \tag{332}$$

10. In (332), Estimation of p by an upper bound:

$$\langle y, m, n, x, \mathsf{s}(q), \overline{\mathsf{p}}(r), z\rangle \preceq \langle b, m, n, x, q, r, h\rangle \tag{333}$$

$$u \prec v \rightarrow \langle y, m, n, x, \mathsf{s}(q), u, z\rangle \prec \langle y, m, n, x, \mathsf{s}(q), v, z\rangle \tag{334}$$

$$\mathcal{E}_{\mathsf{p}\preceq\overline{\mathsf{p}}}$$

$$\mathsf{le}(q,r) \wedge \neg\mathsf{le}(\mathsf{nth}(q,b),x) \wedge \neg\mathsf{lt}(x,\mathsf{nth}(r,b)) \rightarrow \delta_{\mathsf{p}\prec\overline{\mathsf{p}}}(r) \vee \langle y, m, n, x, \mathsf{s}(q), \overline{\mathsf{p}}(r), z\rangle \prec \langle b, m, n, x, q, r, h\rangle \tag{335}$$

11. In (335), Elimination of the strictness predicate call omitting the strict inequality and replacing (335) by the second inequality of $\delta_{\mathsf{p}\prec\overline{\mathsf{p}}}$, viz. (328)
12. In (319), Inverse Weakening eliminating $\neg\mathsf{le}(\mathsf{nth}(q,b),x) \wedge \neg\mathsf{lt}(x,\mathsf{nth}(r,b))$:

$$\mathsf{le}(q,r) \ \rightarrow\ \mathsf{zero} \preceq \langle b, m, n, x, q, r, h\rangle \tag{336}$$

13. In (336), Inductive Evaluation w.r.t. $\mathsf{le}(q,r)$, Symbolic Evaluation:

$$\mathsf{true} \ \rightarrow\ \mathsf{zero} \preceq \langle b, m, n, x, 0, r, h\rangle \tag{337}$$

$$\mathsf{le}(u,v) \ \rightarrow\ \langle b, m, n, x, u, v, h\rangle \preceq \langle b, m, n, x, \mathsf{s}(u), \mathsf{s}(v), h\rangle \tag{338}$$

14. Premise Elimination in (314), (321), (322), (323), (324), (326), (327), (328), (330), (331), (333), (334), (337), and (338)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle b, m, n, x, q, r, h\rangle \mapsto r - q,\ \overline{\mathsf{p}}(x) \mapsto x$

Soundness Predicates

```
function $\lambda_{(314)}$ : list × nat × nat × nat × nat × nat × nat → bool
  $\lambda_{(314)}(b, m, n, x, q, r, h) = \mathsf{true}$
```

```
function $\lambda_{(320)}$ : list × nat × nat × nat × nat × nat × nat → bool
  $\lambda_{(320)}(b, m, n, x, q, r, h) = \mathsf{true}$
```

In the transformation of the second termination hypothesis, the elimination of the strictness predicate $\delta_{\mathsf{p}\prec\overline{\mathsf{p}}}$ from (325) is not sound. Hence we have also to construct a soundness predicate for that kind of unsound transformation. According to [Bra97] we construct a *soundness predicate* $\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}$ such that $\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(q) = \mathsf{true}$ implies $((328) \to (325))[r/q]$ for each natural q. In general, if a strictness predicate $\delta_{g\prec\overline{g}}$ is eliminated by selecting its i_1-th, i_2-th,.. i_k-th strict inequality, the associated soundness predicate procedure $\gamma^{i_1,i_2,..i_k}_{g\prec\overline{g}}$ is obtained by replacing in the procedure $\delta_{g\prec\overline{g}}$ each selected inequality by true and all remaining inequalities by false.
Thus we obtain the following functional procedure for the soundness predicate $\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}$:

```
function $\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}$ : nat → bool
  $\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(0) \quad = \mathsf{false}$
  $\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(\mathsf{s}(v)) = \mathsf{true}$
```

$\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(x) = \mathsf{true}$ iff $x \neq 0$.

```
function $\lambda_{(329)}$ : list × nat × nat × nat × nat × nat × nat → bool
  $\lambda_{(329)}(b, m, n, x, q, r, h) = \mathsf{true}$
```

The elimination of the strictness predicate call in (335) is sound.

```
function $\lambda_{(336)}$ : list × nat × nat × nat × nat × nat × nat → bool
  $\lambda_{(332)}(b, m, n, x, q, r, h) = \mathsf{true}$
```

Termination Predicates

```
function $\theta_{\mathsf{loop}}$ : list × nat × nat × nat × nat × nat × nat → bool
  $\theta_{\mathsf{loop}}(b, m, n, x, q, r, h) = \mathsf{if}(\,\mathsf{le}(q, r),$
                                  $\mathsf{if}(\,\mathsf{le}(\mathsf{nth}(q, b), x),$
                                     $\theta_{\mathsf{loop}}(b, m, n, x, \mathsf{s}(q), r, h),$
                                     $\mathsf{if}(\,\mathsf{lt}(x, \mathsf{nth}(r, b))$
                                        $\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(r) \wedge \theta_{\mathsf{loop}}(b, m, n, x, q, \mathsf{p}(r), h),$
                                        $\theta_{\mathsf{loop}}(\,\mathsf{set}(q, \mathsf{set}(r, b, \mathsf{nth}(q, b)), \mathsf{nth}(r, b)),$
                                              $m, n, x, \mathsf{s}(q), \mathsf{p}(r), \mathsf{nth}(r, b)))),$
                                  $\mathsf{true})$
```

$\theta_{\mathsf{loop}}(b, m, n, x, q, r, h) = \mathsf{true}$ iff $q \neq 0$ or $x \not< b[i]$ for some i, $0 < i \leq r$.

```
function main : list × nat × nat × nat → bool
  $\mathsf{main}(b, m, n, h) = \theta_{\mathsf{loop}}(b, m, n, \mathsf{nth}(m, b), \mathsf{s}(m), \mathsf{p}(n), h)$
```

$\theta_{\mathsf{main}}(b, m, n, h) = \mathsf{true}$.

55 llist_search3 ([Gri81] p. 347)

Let b be a list containing ordered lists of length n. Consider that also each "column" in b is ordered, i.e. $b[i, j] \leq b[i+1, j]$. Given a natural x, the following imperative procedure determines indices i and j such that $b[i, j] = x$.

```
procedure llist_search3($b$: llist, $n, x, i, j$: nat)
  $i := 0$;  $j := n - 1$;
  while $x \neq b[i,j]$ do
    if $b[i,j] < x$ then $i := i + 1$; else $j := j - 1$ fi
  od
```

The procedure terminates iff $b = \mathsf{empty}$ and $x = 0$ or $b[i,j] = x$ for some i, j satisfying $0 \leq i < \mathsf{len}(b)$ and $0 \leq j < n$ and if there exists a sequence of related elements that connect $b[0, n-1]$ and $b[i,j]$ and that are visited by the loop one by one, i.e. if there exist indices $k_1, .., k_r, l_1, .., l_r$ such that $k_1 = 0$, $l_1 = n - 1$, $k_r = i$, $l_r = j$ and for all $p \in \{1, .., r-1\}$ either $x < b[k_p, l_p]$ and $k_{p+1} = k_p$ and $l_{p+1} = l_p - 1$ or $x > b[k_p, l_p]$ and $k_{p+1} = k_p + 1$ and $l_{p+1} = l_p$.

Translation

```
function loop : llist × nat × nat × nat × nat → nat
  loop(b, n, x, i, j) = if(¬eq(x, lnth(i, j, b)),
                           if( lt(lnth(i, j, b), x),
                              loop(b, n, x, s(i), j),
                              loop(b, n, x, i, p(j)))
                           i)

function main : llist × nat × nat → nat
  main(b, n, x) = loop(b, n, x, 0, p(n))
```

Termination Hypotheses

In this example we have to use a polynomial ordering where tuples of data objects may also be mapped to negative numbers.

$$\neg\mathsf{eq}(x, \mathsf{lnth}(i,j,b)) \wedge \mathsf{lt}(\mathsf{lnth}(i,j,b), x) \rightarrow \langle b, n, x, \mathsf{s}(i), j\rangle \prec \langle b, n, x, i, j\rangle \tag{339}$$

$$\neg\mathsf{eq}(x, \mathsf{lnth}(i,j,b)) \wedge \mathsf{lt}(\mathsf{lnth}(i,j,b), x) \rightarrow \mathsf{zero} \preceq \langle b, n, x, i, j\rangle \tag{340}$$

$$\neg\mathsf{eq}(x, \mathsf{lnth}(i,j,b)) \wedge \neg\mathsf{lt}(\mathsf{lnth}(i,j,b), x) \rightarrow \langle b, n, x, i, \mathsf{p}(j)\rangle \prec \langle b, n, x, i, j\rangle \tag{341}$$

$$\neg\mathsf{eq}(x, \mathsf{lnth}(i,j,b)) \wedge \neg\mathsf{lt}(\mathsf{lnth}(i,j,b), x) \rightarrow \mathsf{zero} \preceq \langle b, n, x, i, j\rangle \tag{342}$$

Transformation

1. In (339), Inductive Evaluation is suggested, but not necessary.
2. In (340), Inductive Evaluation w.r.t. $\mathsf{lnth}(i,j,b)$, Symbolic Evaluation:

$$\neg\mathsf{eq}(x, 0) \wedge \mathsf{lt}(0, x) \rightarrow \mathsf{zero} \prec \langle \mathsf{empty}, n, x, i, j\rangle \tag{343}$$

$$\neg\mathsf{eq}(x, \mathsf{nth}(j,b)) \wedge \mathsf{lt}(\mathsf{nth}(j,b), x) \rightarrow \mathsf{zero} \prec \langle \mathsf{add}(v,w), n, x, 0, j\rangle \tag{344}$$

$$\neg\mathsf{eq}(x, \mathsf{lnth}(u,j,w)) \wedge \mathsf{lt}(\mathsf{lnth}(u,j,w), x) \rightarrow \langle w, n, x, u, j\rangle \preceq \langle \mathsf{add}(v,w), n, x, \mathsf{s}(u), j\rangle \tag{345}$$

   In (344), Inductive Evaluation w.r.t. $\mathsf{nth}(j,b)$ is suggested by our heuristic, but does not result in a "better" termination predicate.

3. In (341), Estimation of p by an upper bound:

$$\langle b, n, x, i, \overline{\mathsf{p}}(j)\rangle \preceq \langle b, n, x, i, j\rangle \tag{346}$$

$$u \prec v \rightarrow \langle b, n, x, i, u\rangle \prec \langle b, n, x, i, v\rangle \tag{347}$$

$\mathcal{E}_{\mathsf{p}\preceq\overline{\mathsf{p}}}$

$$\neg\mathsf{eq}(x, \mathsf{lnth}(i,j,b)) \wedge \neg\mathsf{lt}(\mathsf{lnth}(i,j,b), x) \rightarrow \delta_{\mathsf{p}\prec\overline{\mathsf{p}}}(j) \vee \langle b, n, x, i, \overline{\mathsf{p}}(j)\rangle \prec \langle b, n, x, i, j\rangle \tag{348}$$

4. In (348), Elimination of the strictness predicate call omitting the strict inequality and replacing (348) by the inequality of the second case of $\delta_{\mathsf{p}\prec\overline{\mathsf{p}}}$:

$$v \prec \overline{\mathsf{p}}(\mathsf{s}(v)) \tag{349}$$

5. In (342), Inductive Evaluation w.r.t. $\mathsf{lnth}(i,j,b)$, Symbolic Evaluation:

$$\neg\mathsf{eq}(x,\mathsf{nth}(j,b)) \wedge \neg\mathsf{lt}(\mathsf{nth}(j,b),x) \;\rightarrow\; \mathsf{zero} \prec \langle \mathsf{add}(v,w),n,x,0,j\rangle \tag{350}$$

$$\neg\mathsf{eq}(x,\mathsf{lnth}(u,j,w)) \wedge \neg\mathsf{lt}(\mathsf{lnth}(u,j,w),x) \;\rightarrow\; \langle w,n,x,u,j\rangle \preceq \langle \mathsf{add}(v,w),n,x,\mathsf{s}(u),j\rangle \tag{351}$$

In (350), Inductive Evaluation w.r.t. $\mathsf{nth}(j,b)$ is suggested by our heuristic, but does not result in a "better" termination predicate.

6. Premise Elimination in (339), (344), (345), (346), (347), (349), (350), and (351), Rejection in (343)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \mathsf{lempty} \mapsto 0,\ \mathsf{ladd}(v,w) \mapsto w+1,\ \langle b,n,x,i,j\rangle \mapsto b-i+j$

Soundness Predicates

function $\lambda_{(339)}$: $\mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

$\lambda_{(339)}(b,n,x,i,j) = \mathsf{true}$

function $\lambda_{(340)}$: $\mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

$\lambda_{(340)}(\mathsf{empty},n,x,i,j) = \mathsf{false}$

$\lambda_{(340)}(\mathsf{add}(v,w),n,x,0,j) = \mathsf{true}$

$\lambda_{(340)}(\mathsf{add}(v,w),n,x,\mathsf{s}(u),j) = \lambda_{(340)}(w,n,x,u,j)$

$\lambda_{(340)}(b,n,x,i,j) = \mathsf{true}$ iff $i < \mathsf{len}(b)$

function $\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}$: $\mathsf{nat} \rightarrow \mathsf{bool}$

$\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(0) = \mathsf{false}$

$\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(\mathsf{s}(v)) = \mathsf{true}$

$\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(x) = \mathsf{true}$ iff $x \neq 0$.

function $\lambda_{(342)}$: $\mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

$\lambda_{(342)}(b,n,x,i,j) = \mathsf{true}$

Termination Predicates

function θ_{loop} : $\mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

```
$\theta_{\mathsf{loop}}(b,n,x,i,j) = \mathsf{if}(\neg\mathsf{eq}(x,\mathsf{lnth}(i,j,b)),$
        $\mathsf{if}(\mathsf{lt}(\mathsf{lnth}(i,j,b),x),$
            $\lambda_{(340)}(b,n,x,i,j) \wedge \theta_{\mathsf{loop}}(b,n,x,\mathsf{s}(i),j),$
            $\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(j) \wedge \theta_{\mathsf{loop}}(b,n,x,i,\mathsf{p}(j)))$
        $\mathsf{true})$
```

$\theta_{\mathsf{loop}}(b,n,x,i,j) = \mathsf{true}$ iff $b[i,j] = x$ or
$b[u,v] = x$ for some u,v satisfying $i \le u < \mathsf{len}(b)$ and $0 \le v \le j$ and
there exist indices $k_1,..,k_r,l_1,..,l_r$ such that
$k_1 = 0$, $l_1 = j$, $k_r = u$, $l_r = v$ and for all $p \in \{1,..,r-1\}$
either $x < b[k_p,l_p]$ and $k_{p+1} = k_p$ and $l_{p+1} = l_p - 1$
or $x > b[k_p,l_p]$ and $k_{p+1} = k_p + 1$ and $l_{p+1} = l_p$.

function θ_{main} : $\mathsf{llist} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$

$\theta_{\mathsf{main}}(b,n,x) = \theta_{\mathsf{loop}}(b,n,x,0,\mathsf{p}(n))$

$\theta_{\mathsf{main}}(b,n,x) = \mathsf{true}$ iff $b = \mathsf{empty}$ and $x = 0$ or
$b[i,j] = x$ for some i,j satisfying $0 \le i < \mathsf{len}(b)$ and $0 \le j < n$ and
there exist indices $k_1,..,k_r,l_1,..,l_r$ such that
$k_1 = 0$, $l_1 = n-1$, $k_r = i$, $l_r = j$ and for all $p \in \{1,..,r-1\}$
either $x < b[k_p,l_p]$ and $k_{p+1} = k_p$ and $l_{p+1} = l_p - 1$
or $x > b[k_p,l_p]$ and $k_{p+1} = k_p + 1$ and $l_{p+1} = l_p$.

56 sqr_sums (**[Gri81] p. 349**)

Given a natural r, the following imperative procedure generates all different ways in which r can be written as the sum of two squares. In particular, two lists a, b are filled with values and the number i of different ways is computed such that finally $a[j]^2 + b[j]^2 = r^2$ for all j, $0 \leq j \leq i-1$.

```
procedure sqr_sums($r$: nat, $a, b$: list, $i, x, y$: nat)
  $i := 0$;  $x := 0$;
  while $2 * x^2 < r$ do $x := x + 1$ od;              % loop1
  $y := x$;
  while $x^2 \leq r$ do                                  % loop3
    while $r < x^2 + y^2$ do $y := y - 1$ od;            % loop2
    if $r = x^2 + y^2$ then
      $a[i] := x$;  $b[i] := y$;
      $i := i + 1$;  $x := x + 1$
    else
      $x := x + 1$
    fi
  od
```

Translation

function loop1 : nat × nat → nat

$\mathsf{loop1}(r, x) = \mathsf{if}(\mathsf{lt}(\mathsf{dbl}(\mathsf{sqr}(x)), r),\ \mathsf{loop1}(r, \mathsf{s}(x)),\ x)$

function loop2 : nat × nat × nat → nat

$\mathsf{loop2}(r, x, y) = \mathsf{if}(\mathsf{lt}(r, \mathsf{plus}(\mathsf{sqr}(x), \mathsf{sqr}(y))),\ \mathsf{loop}(r, x, \mathsf{p}(y)),\ y)$

function loop3 : nat × list × list × nat × nat × nat → nat

$$\begin{aligned}\mathsf{loop3}(r, a, b, i, x, y) = \mathsf{if}(&\mathsf{le}(\mathsf{sqr}(x), r),\\ &\mathsf{if}(\mathsf{eq}(r, \mathsf{plus}(\mathsf{sqr}(x), \mathsf{sqr}(\mathsf{loop2}(r, x, y)))),\\ &\quad \mathsf{loop3}(r,\ \mathsf{set}(i, a, x),\ \mathsf{set}(i, b, \mathsf{loop2}(r, x, y)),\ \mathsf{s}(i),\ \mathsf{s}(x),\ y),\\ &\quad \mathsf{loop3}(r, a, b, i, \mathsf{s}(x), y))\\ &i)\end{aligned}$$

function main : nat × list × list → nat

$\mathsf{main}(r, a, b) = \mathsf{loop3}(r, a, b, 0, \mathsf{loop1}(r, x), \mathsf{loop1}(r, x))$

loop1:

Termination Hypothesis

$$\mathsf{lt}(\mathsf{dbl}(\mathsf{sqr}(x)), r) \ \rightarrow\ \langle r, \mathsf{s}(x)\rangle \prec \langle r, x\rangle \tag{352}$$

Transformation

1. Inductive Evaluation w.r.t. $\mathsf{lt}(x, r)$ (even if our heuristic suggests Inductive Evaluation w.r.t. $\mathsf{sqr}(x)$), Symbolic Evaluation:

$$\mathsf{true} \ \rightarrow\ \langle \mathsf{s}(v), \mathsf{s}(0)\rangle \prec \langle \mathsf{s}(v), 0\rangle \tag{353}$$

$$\begin{aligned}\mathsf{lt}(\mathsf{s}(\mathsf{dbl}(\mathsf{plus}(\mathsf{sqr}(u), \mathsf{dbl}(u)))), v) \ \rightarrow\ &\langle \mathsf{s}(v), \mathsf{s}(\mathsf{s}(u))\rangle + \langle v, u\rangle \preceq\\ &\langle \mathsf{s}(v), \mathsf{s}(u)\rangle + \langle v, \mathsf{s}(u)\rangle\end{aligned} \tag{354}$$

   Notice that for the application of the induction hypothesis the formula $\mathsf{lt}(\mathsf{dbl}(\mathsf{sqr}(\mathsf{s}(u))), \mathsf{s}(v)) \rightarrow \mathsf{lt}(\mathsf{dbl}(\mathsf{sqr}(u)), v)$ has to be proved by *Induction*.
   In (354), Inductive Evaluation w.r.t. $\mathsf{sqr}(u)$ or $\mathsf{dbl}(u)$ is suggested by our heuristic, but does not result in a "better" termination predicate.

2. Premise Elimination in (353) and (354)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle r, x\rangle \mapsto (r-x)^2$

Soundness Predicate

function $\lambda_{(352)}$: $\mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$\lambda_{(352)}(r, x) = \mathsf{true}$

Termination Predicate

function θ_{loop1} : $\mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$\theta_{\mathsf{loop1}}(r, x) = \mathsf{true}$

loop2:

Termination Hypothesis

$$\mathsf{lt}(r, \mathsf{plus}(\mathsf{sqr}(x), \mathsf{sqr}(y))) \ \to\ \langle r, x, \mathsf{p}(y)\rangle \prec \langle r, x, y\rangle \tag{355}$$

Transformation

1. Estimation of p by an upper bound:

$$\langle r, x, \overline{\mathsf{p}}(y)\rangle \preceq \langle r, x, y\rangle \tag{356}$$

$$u \prec v \to \langle r, x, u\rangle \prec \langle r, x, v\rangle \tag{357}$$

$$\mathcal{E}_{\mathsf{p}\preceq\overline{\mathsf{p}}}$$

$$\mathsf{lt}(r, \mathsf{plus}(\mathsf{sqr}(x), \mathsf{sqr}(y))) \to \delta_{\mathsf{p}\prec\overline{\mathsf{p}}}(y) \lor \langle r, x, \overline{\mathsf{p}}(y)\rangle \prec \langle r, x, y\rangle \tag{358}$$

2. In (358), Elimination of the strictness predicate call omitting the strict inequality and replacing (358) by the inequality of the second case of $\delta_{\mathsf{p}\prec\overline{\mathsf{p}}}$:

$$v \prec \overline{\mathsf{p}}(\mathsf{s}(v)) \tag{359}$$

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle r, x, y\rangle \mapsto y,\ \overline{\mathsf{p}}(x) \mapsto x$

Soundness Predicate

function $\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}$: $\mathsf{nat} \to \mathsf{bool}$

$\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(0) \quad = \mathsf{false}$

$\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(\mathsf{s}(v)) = \mathsf{true}$

$\gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(x) = \mathsf{true}$ iff $x \neq 0$.

Termination Predicate

function θ_{loop2} : $\mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$

$\theta_{\mathsf{loop2}}(r, x, y) = \mathsf{if}(\ \mathsf{lt}(r, \mathsf{plus}(\mathsf{sqr}(x), \mathsf{sqr}(y))),$
$\quad \gamma^2_{\mathsf{p}\prec\overline{\mathsf{p}}}(y) \land \theta_{\mathsf{loop}}(r, x, \mathsf{p}(y)),$
$\quad \mathsf{true})$

$\theta_{\mathsf{loop2}}(r, x, y) = \mathsf{true}$ iff $x^2 \leq r$.

loop3:

Termination Hypotheses

$$\mathsf{le}(\mathsf{sqr}(x), r) \land \ \mathsf{eq}(r, \mathsf{plus}(\mathsf{sqr}(x), \mathsf{sqr}(\mathsf{loop2}(r, x, y)))) \to$$
$$\langle r,\ \mathsf{set}(i, a, x),\ \mathsf{set}(i, b, \mathsf{loop2}(r, x, y)),\ \mathsf{s}(i),\ \mathsf{s}(x),\ y\rangle \prec \langle r, a, b, i, x, y\rangle \tag{360}$$

$$\mathsf{le}(\mathsf{sqr}(x), r) \land \neg\mathsf{eq}(r, \mathsf{plus}(\mathsf{sqr}(x), \mathsf{sqr}(\mathsf{loop2}(r, x, y)))) \to$$
$$\langle r, a, b, i, \mathsf{s}(x), y\rangle \prec \langle r, a, b, i, x, y\rangle \tag{361}$$

Transformation

1. In (360), Generalization $\{\mathsf{set}(i,a,x)/w,\ \mathsf{set}(i,b,\mathsf{loop2}(r,x,y))/z\}$ (since $a,b \notin \{r,x\}$):

$$\mathsf{le}(\mathsf{sqr}(x),r) \wedge \ \mathsf{eq}(r,\mathsf{plus}(\mathsf{sqr}(x),\mathsf{sqr}(\mathsf{loop2}(r,x,y)))) \rightarrow \langle r,w,z,\mathsf{s}(i),\mathsf{s}(x),y\rangle \prec \langle r,a,b,i,x,y\rangle \qquad (362)$$

2. In (362), Inverse Weakening eliminating $\mathsf{eq}(r,\mathsf{plus}(\mathsf{sqr}(x),\mathsf{sqr}(\mathsf{loop2}(r,x,y))))$:

$$\mathsf{le}(\mathsf{sqr}(x),r) \ \rightarrow \ \langle r,w,z,\mathsf{s}(i),\mathsf{s}(x),y\rangle \prec \langle r,a,b,i,x,y\rangle \qquad (363)$$

3. In (363), Inductive Evaluation w.r.t. $\mathsf{le}(x,r)$ (even if our heuristic suggests Inductive Evaluation w.r.t. $\mathsf{sqr}(x)$), Symbolic Evaluation :

$$\mathsf{true} \ \rightarrow \ \langle r,w,z,\mathsf{s}(i),\mathsf{s}(0),y\rangle \prec \langle r,a,b,i,0,y\rangle \qquad (364)$$

$$\mathsf{le}(\mathsf{plus}(\mathsf{sqr}(u),\mathsf{dbl}(u)),v) \ \rightarrow \ \langle \mathsf{s}(v),w,z,\mathsf{s}(i),\mathsf{s}(\mathsf{s}(u)),y\rangle + \langle v,a,b,i,u,y\rangle \preceq \langle \mathsf{s}(v),a,b,i,\mathsf{s}(u),y\rangle + \langle v,w,z,\mathsf{s}(i),\mathsf{s}(u),y\rangle \qquad (365)$$

4. In (361), Inverse Weakening eliminating $\neg\mathsf{eq}(r,\mathsf{plus}(\mathsf{sqr}(x),\mathsf{sqr}(\mathsf{loop2}(r,x,y))))$:

$$\mathsf{le}(\mathsf{sqr}(x),r) \ \rightarrow \ \langle r,a,b,i,\mathsf{s}(x),y)\rangle \prec \langle r,a,b,i,x,y\rangle \qquad (366)$$

5. In (366), Inductive Evaluation w.r.t. $\mathsf{le}(x,r)$ (even if our heuristic suggests Inductive Evaluation w.r.t. $\mathsf{sqr}(x)$), Symbolic Evaluation :

$$\mathsf{true} \ \rightarrow \ \langle r,a,b,i,\mathsf{s}(0),y)\rangle \prec \langle r,a,b,i,0,y\rangle \qquad (367)$$

$$\mathsf{le}(\mathsf{plus}(\mathsf{sqr}(u),\mathsf{dbl}(u)),v) \ \rightarrow \ \langle \mathsf{s}(v),a,b,i,\mathsf{s}(\mathsf{s}(u)),y)\rangle + \langle v,a,b,i,u,y\rangle \preceq \langle \mathsf{s}(v),a,b,i,\mathsf{s}(u),y\rangle + \langle v,a,b,i,\mathsf{s}(u),y)\rangle \qquad (368)$$

6. Premise Elimination in (364), (365), (367), and (368)

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle r,a,b,i,x,y\rangle \mapsto (r-x+1)^2$

Soundness Predicates

function $\lambda_{(363)}$: $\mathsf{nat} \times \mathsf{list} \times \mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{list} \times \mathsf{list} \rightarrow \mathsf{bool}$
$\lambda_{(363)}(r,a,b,i,x,y,w,z) = \mathsf{true}$

function $\lambda_{(366)}$: $\mathsf{nat} \times \mathsf{list} \times \mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
$\lambda_{(366)}(r,a,b,i,x,y) = \mathsf{true}$

Termination Predicate

```
function $\theta_{\mathsf{loop3}}$ : $\mathsf{nat} \times \mathsf{list} \times \mathsf{list} \times \mathsf{nat} \times \mathsf{nat} \times \mathsf{nat} \rightarrow \mathsf{bool}$
  $\theta_{\mathsf{loop3}}(r,a,b,i,x,y) = \mathsf{if}(\mathsf{le}(\mathsf{sqr}(x),r),$
                             $\mathsf{if}(\theta_{\mathsf{loop2}}(r,x,y),$
                                $\mathsf{if}(\mathsf{eq}(r,\mathsf{plus}(\mathsf{sqr}(x),\mathsf{sqr}(\mathsf{loop2}(r,x,y)))),$
                                   $\theta_{\mathsf{loop3}}(r,\ \mathsf{set}(i,a,x),\ \mathsf{set}(i,b,\mathsf{loop2}(r,x,y)),\ \mathsf{s}(i),\ \mathsf{s}(x),\ y),$
                                   $\theta_{\mathsf{loop3}}(r,a,b,i,\mathsf{s}(x),y))$
                                $\mathsf{false})$
                             $\mathsf{true})$

$\theta_{\mathsf{loop3}}(r,a,b,i,x,y) = \mathsf{true}.$
```

main:

Termination Predicate

function θ_{main} : nat × list × list → bool

$\theta_{\mathsf{main}}(r, a, b) = \theta_{\mathsf{loop3}}(r, a, b, 0, \mathsf{loop1}(r, x), \mathsf{loop1}(r, x))$

$\theta_{\mathsf{main}}(r, a, b) = \mathsf{true}$.

57 sqrt2 ([Gri81] p. 352)

Given a natural n, the following imperative procedure computes the maximum natural q such that $q^2 \leq n$.

```
procedure sqrt2($n, p, q, r$ : nat)
  $p := 1$;  $q := 0$;  $r := n$;
  while $p \leq n$ do $p := 4 * p$ od;                % loop1
  while $p \neq 1$ do                                 % loop2
    $p := p/4$;
    $q := q/2$;
    if $2 * q + p \leq r$ then
      $r := r - (2 * q + p)$;
      $q := q + p$
    else
      skip
    fi
  od
```

Translation

```
function loop1 : nat × nat → nat
  loop1($n, p$) = if(le($p, n$), loop1($n$, dbl(dbl($p$))), $p$)
function loop2 : nat × nat × nat × nat → nat
  loop2($p, q, r$) = if($\neg$eq($p$, s(0)),
                       if( le(plus(dbl(half($q$)), half(half($p$))), $r$),
                          loop2( half(half($p$)),
                                 plus(half($q$), half(half($p$))),
                                 minus($r$, plus(dbl(half($q$)), half(half($p$))))),
                          loop2(half(half($p$)), half($q$), $r$)),
                       $q$)
function main : nat → nat
  main($n$) = loop2(loop1(s(0), $n$), 0, $n$)
```

loop1:

Termination Hypotheses

For termination analysis of loop1, we use a polynomial ordering where tuples of data objects are mapped to negative numbers.

$$\mathsf{le}(p, n) \rightarrow \langle n, \mathsf{dbl}(\mathsf{dbl}(p))\rangle \prec \langle n, p\rangle \tag{369}$$

$$\mathsf{le}(p, n) \rightarrow \mathsf{zero} \preceq \langle n, p\rangle \tag{370}$$

Transformation

1. Estimation of dbl by a lower bound:

$$\langle n, \underline{\mathsf{dbl}}(\underline{\mathsf{dbl}}(p))\rangle \preceq \langle n, p\rangle \tag{371}$$

$$u \prec v \rightarrow \langle n, v\rangle \prec \langle n, u\rangle \tag{372}$$

$$u \prec v \rightarrow \langle n, \underline{\mathsf{dbl}}(v)\rangle \prec \langle n, \underline{\mathsf{dbl}}(u)\rangle \tag{373}$$

$$\mathcal{E}_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}$$
$$\mathsf{le}(p,n) \rightarrow \delta_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}(p) \vee \delta_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}(\mathsf{dbl}(p)) \vee \langle n, \underline{\mathsf{dbl}}(\underline{\mathsf{dbl}}(p))\rangle \prec \langle n,p\rangle \tag{374}$$

2. Computing the estimation inequalities $\mathcal{E}_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}$ and the strictness predicate $\delta_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}$:

$$\underline{\mathsf{dbl}}(0) \preceq 0 \tag{375}$$
$$\underline{\mathsf{dbl}}(\mathsf{s}(v)) \preceq \mathsf{s}(\mathsf{s}(\underline{\mathsf{dbl}}(v))) \tag{376}$$
$$u \prec v \rightarrow \mathsf{s}(\mathsf{s}(u)) \prec \mathsf{s}(\mathsf{s}(v)) \tag{377}$$

```
function $\delta_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}$ : nat → bool
  $\delta_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}(0) = \underline{\mathsf{dbl}}(0) \prec 0$
  $\delta_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}(\mathsf{s}(v)) = \underline{\mathsf{dbl}}(\mathsf{s}(v)) \prec \mathsf{s}(\mathsf{s}(\underline{\mathsf{dbl}}(v))) \vee \delta_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}(v)$
```

3. In (374), Elimination of the strictness predicate call omitting the strict inequality and replacing (374) by the second inequality of $\delta_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}$:

$$\underline{\mathsf{dbl}}(\mathsf{s}(v)) \prec \mathsf{s}(\mathsf{s}(\underline{\mathsf{dbl}}(v))) \tag{378}$$

4. In (370), Inductive Evaluation w.r.t. $\mathsf{le}(p,n)$:

$$\mathsf{true} \rightarrow \mathsf{zero} \preceq \langle n, 0\rangle \tag{379}$$
$$\mathsf{le}(u,v) \rightarrow \langle v,u\rangle \preceq \langle \mathsf{s}(v), \mathsf{s}(u)\rangle \tag{380}$$

5. Premise Elimination in (371), (372), (373), (375), (376), (377), (378), (379), and (380)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle n,p\rangle \mapsto n-p,\ \underline{\mathsf{dbl}}(x) \mapsto x$

Soundness Predicate

```
function $\gamma^2_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}$ : nat → bool
  $\gamma^2_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}(0) = \mathsf{false}$
  $\gamma^2_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}(\mathsf{s}(v)) = \mathsf{true}$
```

$\gamma^2_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}(x) = \mathsf{true}$ iff $x \neq 0$

```
function $\lambda_{(370)}$ : nat × nat → nat
  $\lambda_{(370)}(n,p) = \mathsf{true}$
```

Termination Predicate

```
function $\theta_{\mathsf{loop1}}$ : nat × nat → bool
  $\theta_{\mathsf{loop1}}(n,p) = \mathsf{if}(\mathsf{le}(p,n),$
                  $(\gamma^2_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}(p) \vee \gamma^2_{\underline{\mathsf{dbl}}\prec\mathsf{dbl}}(\mathsf{dbl}(p))) \wedge \theta_{\mathsf{loop1}}(n, \mathsf{dbl}(\mathsf{dbl}(p))),$
                  $\mathsf{true})$
```

$\theta_{\mathsf{loop1}}(n,p) = \mathsf{true}$ iff $p \neq 0$.

loop2:

Termination Hypotheses

$$\neg\mathsf{eq}(p,\mathsf{s}(0)) \wedge \mathsf{le}(\mathsf{plus}(\mathsf{dbl}(\mathsf{half}(q)), \mathsf{half}(\mathsf{half}(p))), r) \rightarrow \langle \mathsf{half}(\mathsf{half}(p)), \mathsf{plus}(\mathsf{half}(q), \mathsf{half}(\mathsf{half}(p))), \mathsf{minus}(r, \mathsf{plus}(\mathsf{dbl}(\mathsf{half}(q)), \mathsf{half}(\mathsf{half}(p))))\rangle \prec \langle p,q,r\rangle \tag{381}$$

$$\neg\mathsf{eq}(p,\mathsf{s}(0)) \wedge \neg\mathsf{le}(\mathsf{plus}(\mathsf{dbl}(\mathsf{half}(q)), \mathsf{half}(\mathsf{half}(p))), r) \rightarrow \langle \mathsf{half}(\mathsf{half}(p)), \mathsf{half}(q), r\rangle \prec \langle p,q,r\rangle \tag{382}$$

Transformation

1. In (381), Generalization $\{\mathsf{plus}(\mathsf{half}(q), \mathsf{half}(\mathsf{half}(p)))/x, \mathsf{minus}(r, \mathsf{plus}(\mathsf{dbl}(\mathsf{half}(q)), \mathsf{half}(\mathsf{half}(p))))/y\}$ and Estimation of half by an upper bound:

$$\langle \overline{\mathsf{half}}(\overline{\mathsf{half}}(p)), x, y\rangle \preceq \langle p, q, r\rangle \tag{383}$$
$$u \prec v \rightarrow \langle u, x, y\rangle \prec \langle v, q, r\rangle \tag{384}$$
$$u \prec v \rightarrow \langle \overline{\mathsf{half}}(u), x, y\rangle \prec \langle \overline{\mathsf{half}}(v), q, r\rangle \tag{385}$$
$$\mathcal{E}_{\mathsf{half} \preceq \overline{\mathsf{half}}}$$
$$\neg\mathsf{eq}(p, \mathsf{s}(0)) \wedge \mathsf{le}(\mathsf{plus}(\mathsf{dbl}(\mathsf{half}(q)), \mathsf{half}(\mathsf{half}(p))), r) \rightarrow$$
$$\delta_{\mathsf{half} \prec \overline{\mathsf{half}}}(\mathsf{half}(p)) \vee \delta_{\mathsf{half} \prec \overline{\mathsf{half}}}(p) \vee \langle \overline{\mathsf{half}}(\overline{\mathsf{half}}(p)), x, y\rangle \prec \langle p, q, r\rangle \tag{386}$$

2. Computing the estimation inequalities $\mathcal{E}_{\mathsf{half} \preceq \overline{\mathsf{half}}}$ and the strictness predicate $\delta_{\mathsf{half} \prec \overline{\mathsf{half}}}$:

$$0 \preceq \overline{\mathsf{half}}(0) \tag{387}$$
$$0 \preceq \overline{\mathsf{half}}(\mathsf{s}(0)) \tag{388}$$
$$\mathsf{s}(\overline{\mathsf{half}}(v)) \preceq \overline{\mathsf{half}}(\mathsf{s}(\mathsf{s}(v))) \tag{389}$$
$$u \prec v \rightarrow \mathsf{s}(u) \prec \mathsf{s}(v) \tag{390}$$

```
function $\delta_{\mathsf{half} \prec \overline{\mathsf{half}}}$ : nat → bool
  $\delta_{\mathsf{half} \prec \overline{\mathsf{half}}}(0)$          $= 0 \prec \overline{\mathsf{half}}(0)$
  $\delta_{\mathsf{half} \prec \overline{\mathsf{half}}}(\mathsf{s}(0))$       $= 0 \prec \overline{\mathsf{half}}(\mathsf{s}(0))$
  $\delta_{\mathsf{half} \prec \overline{\mathsf{half}}}(\mathsf{s}(\mathsf{s}(v)))$ $= \mathsf{s}(\overline{\mathsf{half}}(v)) \prec \overline{\mathsf{half}}(\mathsf{s}(\mathsf{s}(v))) \vee \delta_{\mathsf{half} \prec \overline{\mathsf{half}}}(v)$
```

3. In (386), Elimination of the strictness predicate call omitting the strict inequality and replacing (386) by the second and the third inequality of $\delta_{\mathsf{half} \prec \overline{\mathsf{half}}}$:

$$0 \prec \overline{\mathsf{half}}(\mathsf{s}(0)) \tag{391}$$
$$\mathsf{s}(\overline{\mathsf{half}}(v)) \prec \overline{\mathsf{half}}(\mathsf{s}(\mathsf{s}(v))) \tag{392}$$

4. In (382), Generalization $\{\mathsf{half}(q)/x\}$ and Estimation of half by an upper bound:

$$\langle \overline{\mathsf{half}}(\overline{\mathsf{half}}(p)), x, r\rangle \preceq \langle p, q, r\rangle \tag{393}$$
$$u \prec v \rightarrow \langle u, x, r\rangle \prec \langle v, q, r\rangle \tag{394}$$
$$u \prec v \rightarrow \langle \overline{\mathsf{half}}(u), x, r\rangle \prec \langle \overline{\mathsf{half}}(v), q, r\rangle \tag{395}$$
$$\mathcal{E}_{\mathsf{half} \preceq \overline{\mathsf{half}}}$$
$$\neg\mathsf{eq}(p, \mathsf{s}(0)) \wedge \neg\mathsf{le}(\mathsf{plus}(\mathsf{dbl}(\mathsf{half}(q)), \mathsf{half}(\mathsf{half}(p))), r) \rightarrow$$
$$\delta_{\mathsf{half} \prec \overline{\mathsf{half}}}(\mathsf{half}(p)) \vee \delta_{\mathsf{half} \prec \overline{\mathsf{half}}}(p) \vee \langle \overline{\mathsf{half}}(\overline{\mathsf{half}}(p)), x, r\rangle \prec \langle p, q, r\rangle \tag{396}$$

5. In (396), Elimination of the strictness predicate call omitting the strict inequality and replacing (386) by the second and the third inequality of $\delta_{\mathsf{half} \prec \overline{\mathsf{half}}}$, viz. (391) and (392).

Polynomial Ordering
$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v + 1,\ \langle p, q, r\rangle \mapsto p,\ \overline{\mathsf{half}}(x) \mapsto x$

Soundness Predicate

```
function $\gamma^{2,3}_{\mathsf{half} \prec \overline{\mathsf{half}}}$ : nat → bool
  $\gamma^{2,3}_{\mathsf{half} \prec \overline{\mathsf{half}}}(0)$          = false
  $\gamma^{2,3}_{\mathsf{half} \prec \overline{\mathsf{half}}}(\mathsf{s}(0))$       = true
  $\gamma^{2,3}_{\mathsf{half} \prec \overline{\mathsf{half}}}(\mathsf{s}(\mathsf{s}(v)))$ = true
```

$\gamma^{2,3}_{\mathsf{half} \prec \overline{\mathsf{half}}}(x) = \mathsf{true}$ iff $x \neq 0$

Termination Predicate

function θ_{loop2} : nat × nat × nat × nat → bool

$$\begin{aligned}\theta_{\mathsf{loop2}}(p,q,r) = \mathsf{if}(&\neg\mathsf{eq}(p,\mathsf{s}(0)),\\ &\mathsf{if}(\mathsf{le}(\mathsf{plus}(\mathsf{dbl}(\mathsf{half}(q)),\mathsf{half}(\mathsf{half}(p))),r),\\ &\quad(\gamma^{2,3}_{\mathsf{half}\prec\overline{\mathsf{half}}}(p) \vee \gamma^{2,3}_{\mathsf{half}\prec\overline{\mathsf{half}}}(\mathsf{half}(p))) \mathbin{\dot\wedge}\\ &\qquad\theta_{\mathsf{loop2}}(\mathsf{half}(\mathsf{half}(p)),\\ &\qquad\qquad\mathsf{plus}(\mathsf{half}(q),\mathsf{half}(\mathsf{half}(p))),\\ &\qquad\qquad\mathsf{minus}(r,\mathsf{plus}(\mathsf{dbl}(\mathsf{half}(q)),\mathsf{half}(\mathsf{half}(p))))),\\ &\quad(\gamma^{2,3}_{\mathsf{half}\prec\overline{\mathsf{half}}}(p) \vee \gamma^{2,3}_{\mathsf{half}\prec\overline{\mathsf{half}}}(\mathsf{half}(p))) \mathbin{\dot\wedge} \theta_{\mathsf{loop2}}(\mathsf{half}(\mathsf{half}(p)),\mathsf{half}(q),r)),\\ &\mathsf{true})\end{aligned}$$

$\theta_{\mathsf{loop2}}(p,q,r) = \mathsf{true}$ iff $p \neq 0$ and $p \neq 2$ and $p \neq 3$.

main:

Termination Predicate

function θ_{main} : nat → bool

$$\theta_{\mathsf{main}}(n) = \theta_{\mathsf{loop1}}(\mathsf{s}(0),n) \mathbin{\dot\wedge} \theta_{\mathsf{loop2}}(\mathsf{loop1}(\mathsf{s}(0),n),0,n)$$

$\theta_{\mathsf{main}}(n) = \mathsf{true}$.

58 singles **([Gri81] p. 354)**

Given the sorted lists f (of length n) and g (of length m) containing no duplicates, the imperative procedure singles counts in c the number of all naturals x that occur in at least one list but not in both and that are less than or equal to the minimum of $f[n-1]$ and $g[m-1]$.

```
procedure singles($h, k, c, n, m$: nat, $f, g$: list)
  $h := 0$;  $k := 0$;  $c := 0$;
  while $h \neq n \wedge k \neq m$ do
    if $f[h] < g[k]$ then
      $h := h + 1$;  $c := c + 1$
    else
      if $g[k] < f[h]$ then
        $k : k + 1$;  $c := c + 1$
      else
        $h := h + 1$;  $k : k + 1$
      fi
    fi
  od
```

Translation

function loop : nat × nat × nat × nat × nat × list × list → nat

$$\begin{aligned}\mathsf{loop}(h,k,c,n,m,f,g) = \mathsf{if}(&\neg\mathsf{eq}(h,n) \wedge \neg\mathsf{eq}(k,m),\\ &\mathsf{if}(\mathsf{lt}(\mathsf{nth}(h,f),\mathsf{nth}(k,g)),\\ &\quad\mathsf{loop}(\mathsf{s}(h),k,\mathsf{s}(c),n,m,f,g),\\ &\quad\mathsf{if}(\mathsf{lt}(\mathsf{nth}(k,g),\mathsf{nth}(h,f)),\\ &\qquad\mathsf{loop}(h,\mathsf{s}(k),\mathsf{s}(c),n,m,f,g),\\ &\qquad\mathsf{loop}(\mathsf{s}(h),\mathsf{s}(k),c,n,m,f,g)))\\ &c)\end{aligned}$$

function main : nat × nat × list × list → nat

$$\mathsf{main}(n,m,f,g) = \mathsf{loop}(0,0,0,n,m,f,g)$$

Termination Hypotheses

$$\neg\mathsf{eq}(h,n)\wedge\neg\mathsf{eq}(k,m)\wedge\ \mathsf{lt}(\mathsf{nth}(h,f),\mathsf{nth}(k,g))\rightarrow$$
$$\langle\mathsf{s}(h),k,\mathsf{s}(c),n,m,f,g\rangle\prec\langle h,k,c,n,m,f,g\rangle \tag{397}$$
$$\neg\mathsf{eq}(h,n)\wedge\neg\mathsf{eq}(k,m)\wedge\neg\mathsf{lt}(\mathsf{nth}(h,f),\mathsf{nth}(k,g))\wedge\ \mathsf{lt}(\mathsf{nth}(k,g),\mathsf{nth}(h,f))\rightarrow$$
$$\langle h,\mathsf{s}(k),\mathsf{s}(c),n,m,f,g\rangle\prec\langle h,k,c,n,m,f,g\rangle \tag{398}$$
$$\neg\mathsf{eq}(h,n)\wedge\neg\mathsf{eq}(k,m)\wedge\neg\mathsf{lt}(\mathsf{nth}(h,f),\mathsf{nth}(k,g))\wedge\neg\mathsf{lt}(\mathsf{nth}(k,g),\mathsf{nth}(h,f))\rightarrow$$
$$\langle\mathsf{s}(h),\mathsf{s}(k),c,n,m,f,g\rangle\prec\langle h,k,c,n,m,f,g\rangle \tag{399}$$

Transformation

1. In (397), Inverse Weakening eliminating $\mathsf{lt}(\mathsf{nth}(h,f),\mathsf{nth}(k,g))$:

$$\neg\mathsf{eq}(h,n)\wedge\neg\mathsf{eq}(k,m)\ \rightarrow\ \langle\mathsf{s}(h),k,\mathsf{s}(c),n,m,f,g\rangle\prec\langle h,k,c,n,m,f,g\rangle \tag{400}$$

2. In (400), Inductive Evaluation w.r.t. $\mathsf{eq}(h,n)$, Symbolic Evaluation:

$$\mathsf{true}\wedge\neg\mathsf{eq}(k,m)\ \rightarrow\ \langle\mathsf{s}(0),k,\mathsf{s}(c),\mathsf{s}(v),m,f,g\rangle\prec\langle 0,k,c,\mathsf{s}(v),m,f,g\rangle \tag{401}$$
$$\mathsf{true}\wedge\neg\mathsf{eq}(k,m)\ \rightarrow\ \langle\mathsf{s}(\mathsf{s}(u)),k,\mathsf{s}(c),0,m,f,g\rangle\prec\langle\mathsf{s}(u),k,c,0,m,f,g\rangle \tag{402}$$
$$\neg\mathsf{eq}(u,v)\wedge\neg\mathsf{eq}(k,m)\ \rightarrow\ \langle\mathsf{s}(\mathsf{s}(u)),k,\mathsf{s}(c),\mathsf{s}(v),m,f,g\rangle+\langle u,k,c,v,m,f,g\rangle\preceq$$
$$\langle\mathsf{s}(u),k,c,\mathsf{s}(v),m,f,g\rangle+\langle\mathsf{s}(u),k,\mathsf{s}(c),v,m,f,g\rangle \tag{403}$$

3. In (398), Inverse Weakening eliminating $\neg\mathsf{lt}(\mathsf{nth}(h,f),\mathsf{nth}(k,g))\wedge\mathsf{lt}(\mathsf{nth}(k,g),\mathsf{nth}(h,f))$:

$$\neg\mathsf{eq}(h,n)\wedge\neg\mathsf{eq}(k,m)\ \rightarrow\ \langle h,\mathsf{s}(k),\mathsf{s}(c),n,m,f,g\rangle\prec\langle h,k,c,n,m,f,g\rangle \tag{404}$$

4. In (404), Inductive Evaluation w.r.t. $\mathsf{eq}(k,m)$, Symbolic Evaluation:

$$\neg\mathsf{eq}(h,n)\wedge\mathsf{true}\ \rightarrow\ \langle h,\mathsf{s}(0),\mathsf{s}(c),n,\mathsf{s}(v),f,g\rangle\prec\langle h,0,c,n,\mathsf{s}(v),f,g\rangle \tag{405}$$
$$\neg\mathsf{eq}(h,n)\wedge\mathsf{true}\ \rightarrow\ \langle h,\mathsf{s}(\mathsf{s}(u)),\mathsf{s}(c),n,0,f,g\rangle\prec\langle h,\mathsf{s}(u),c,n,0,f,g\rangle \tag{406}$$
$$\neg\mathsf{eq}(h,n)\wedge\neg\mathsf{eq}(u,v)\ \rightarrow\ \langle h,\mathsf{s}(\mathsf{s}(u)),\mathsf{s}(c),n,\mathsf{s}(v),f,g\rangle+\langle h,u,c,n,v,f,g\rangle\preceq$$
$$\langle h,\mathsf{s}(u),c,n,\mathsf{s}(v),f,g\rangle+\langle h,\mathsf{s}(u),\mathsf{s}(c),n,v,f,g\rangle \tag{407}$$

5. In (399), Inverse Weakening eliminating $\neg\mathsf{lt}(\mathsf{nth}(h,f),\mathsf{nth}(k,g))\wedge\neg\mathsf{lt}(\mathsf{nth}(k,g),\mathsf{nth}(h,f))$:

$$\neg\mathsf{eq}(h,n)\wedge\neg\mathsf{eq}(k,m)\ \rightarrow\ \langle\mathsf{s}(h),\mathsf{s}(k),c,n,m,f,g\rangle\prec\langle h,k,c,n,m,f,g\rangle \tag{408}$$

6. In (408), Inductive Evaluation w.r.t. $\mathsf{eq}(h,n)$, Symbolic Evaluation:

$$\mathsf{true}\wedge\neg\mathsf{eq}(k,m)\ \rightarrow\ \langle\mathsf{s}(0),\mathsf{s}(k),c,\mathsf{s}(v),m,f,g\rangle\prec\langle 0,k,c,\mathsf{s}(v),m,f,g\rangle \tag{409}$$
$$\mathsf{true}\wedge\neg\mathsf{eq}(k,m)\ \rightarrow\ \langle\mathsf{s}(\mathsf{s}(u)),\mathsf{s}(k),c,0,m,f,g\rangle\prec\langle\mathsf{s}(u),k,c,0,m,f,g\rangle \tag{410}$$
$$\neg\mathsf{eq}(u,v)\wedge\neg\mathsf{eq}(k,m)\ \rightarrow\ \langle\mathsf{s}(\mathsf{s}(u)),\mathsf{s}(k),c,\mathsf{s}(v),m,f,g\rangle+\langle u,k,c,v,m,f,g\rangle\preceq$$
$$\langle\mathsf{s}(u),k,c,\mathsf{s}(v),m,f,g\rangle+\langle\mathsf{s}(u),\mathsf{s}(k),c,v,m,f,g\rangle \tag{411}$$

7. In (409), Inductive Evaluation w.r.t. $\mathsf{eq}(k,m)$, Symbolic Evaluation:

$$\mathsf{true}\wedge\mathsf{true}\ \rightarrow\ \langle\mathsf{s}(0),\mathsf{s}(0),c,\mathsf{s}(v),\mathsf{s}(y),f,g\rangle\prec\langle 0,0,c,\mathsf{s}(v),\mathsf{s}(y),f,g\rangle \tag{412}$$
$$\mathsf{true}\wedge\mathsf{true}\ \rightarrow\ \langle\mathsf{s}(0),\mathsf{s}(\mathsf{s}(x)),c,\mathsf{s}(v),0,f,g\rangle\prec\langle 0,\mathsf{s}(x),c,\mathsf{s}(v),0,f,g\rangle \tag{413}$$
$$\mathsf{true}\wedge\neg\mathsf{eq}(x,y)\ \rightarrow\ \langle\mathsf{s}(0),\mathsf{s}(\mathsf{s}(x)),c,\mathsf{s}(v),\mathsf{s}(y),f,g\rangle+\langle 0,x,c,\mathsf{s}(v),y,f,g\rangle\preceq$$
$$\langle 0,\mathsf{s}(x),c,\mathsf{s}(v),\mathsf{s}(y),f,g\rangle+\langle\mathsf{s}(0),\mathsf{s}(x),c,\mathsf{s}(v),y,f,g\rangle \tag{414}$$

8. Premise Elimination in (401), (403), (405), (407), (411), (412), and (414), Rejection in (402), (406), (410), and (413)

Polynomial Ordering

$0 \mapsto 0,\ \mathsf{s}(v) \mapsto v+1,\ \langle h,k,c,n,m,f,g\rangle \mapsto (n-h)^2+(m-k)^2$

Soundness Predicates

$$\begin{aligned}
&\textbf{function}\ \lambda_{(400)} : \mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{list}\times\mathsf{list}\longrightarrow\mathsf{bool}\\
&\quad \lambda_{(400)}(0,k,c,0,m,f,g) = \mathsf{true}\\
&\quad \lambda_{(400)}(0,k,c,\mathsf{s}(v),m,f,g) = \mathsf{true}\\
&\quad \lambda_{(400)}(\mathsf{s}(u),k,c,0,m,f,g) = \mathsf{false}\\
&\quad \lambda_{(400)}(\mathsf{s}(u),k,c,\mathsf{s}(v),m,f,g) = \lambda_{(400)}(u,k,c,v,m,f,g)
\end{aligned}$$

$\lambda_{(400)}(h,k,c,n,m,f,g) = \mathsf{true}$ iff $h \le n$

$$\begin{aligned}
&\textbf{function}\ \lambda_{(404)} : \mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{list}\times\mathsf{list}\longrightarrow\mathsf{bool}\\
&\quad \lambda_{(404)}(h,0,c,n,0,f,g) = \mathsf{true}\\
&\quad \lambda_{(404)}(h,0,c,n,\mathsf{s}(v),f,g) = \mathsf{true}\\
&\quad \lambda_{(404)}(h,\mathsf{s}(u),c,n,0,f,g) = \mathsf{false}\\
&\quad \lambda_{(404)}(h,\mathsf{s}(u),c,n,\mathsf{s}(v),f,g) = \lambda_{(404)}(h,u,c,n,v,f,g)
\end{aligned}$$

$\lambda_{(404)}(h,k,c,n,m,f,g) = \mathsf{true}$ iff $k \le m$

$$\begin{aligned}
&\textbf{function}\ \lambda_{(409)} : \mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{list}\times\mathsf{list}\longrightarrow\mathsf{bool}\\
&\quad \lambda_{(409)}(0,c,v,0,f,g) = \mathsf{true}\\
&\quad \lambda_{(409)}(0,c,v,\mathsf{s}(y),f,g) = \mathsf{true}\\
&\quad \lambda_{(409)}(\mathsf{s}(x),c,v,0,f,g) = \mathsf{false}\\
&\quad \lambda_{(409)}(\mathsf{s}(x),c,v,\mathsf{s}(y),f,g) = \lambda_{(409)}(x,c,v,y,f,g)
\end{aligned}$$

$\lambda_{(409)}(k,c,v,m,f,g) = \mathsf{true}$ iff $k \le m$

$$\begin{aligned}
&\textbf{function}\ \lambda_{(408)} : \mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{list}\times\mathsf{list}\longrightarrow\mathsf{bool}\\
&\quad \lambda_{(408)}(0,k,c,0,m,f,g) = \mathsf{true}\\
&\quad \lambda_{(408)}(0,k,c,\mathsf{s}(v),m,f,g) = \lambda_{(409)}(k,c,v,m,f,g)\\
&\quad \lambda_{(408)}(\mathsf{s}(u),k,c,0,m,f,g) = \mathsf{false}\\
&\quad \lambda_{(408)}(\mathsf{s}(u),k,c,\mathsf{s}(v),m,f,g) = \lambda_{(408)}(u,k,c,v,m,f,g)
\end{aligned}$$

$\lambda_{(408)}(h,k,c,n,m,f,g) = \mathsf{true}$ iff $h = n$ or $h < n \wedge k \le m$

Termination Predicate

```
function $\theta_{\mathsf{loop}} : \mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{nat}\times\mathsf{list}\times\mathsf{list}\longrightarrow\mathsf{bool}$
  $\theta_{\mathsf{loop}}(h,k,c,n,m,f,g) = \mathsf{if}(\neg\mathsf{eq}(h,n)\wedge\neg\mathsf{eq}(k,m),$
                                   $\mathsf{if}(\mathsf{lt}(\mathsf{nth}(h,f),\mathsf{nth}(k,g)),$
                                      $\lambda_{(400)}(h,k,c,n,m,f,g) \wedge \theta_{\mathsf{loop}}(\mathsf{s}(h),k,\mathsf{s}(c),n,m,f,g),$
                                      $\mathsf{if}(\mathsf{lt}(\mathsf{nth}(k,g),\mathsf{nth}(h,f)),$
                                         $\lambda_{(404)}(h,k,c,n,m,f,g) \wedge \theta_{\mathsf{loop}}(h,\mathsf{s}(k),\mathsf{s}(c),n,m,f,g),$
                                         $\lambda_{(408)}(h,k,c,n,m,f,g) \wedge \theta_{\mathsf{loop}}(\mathsf{s}(h),\mathsf{s}(k),c,n,m,f,g)))$
                                   $\mathsf{true})$
```

$\theta_{\mathsf{loop}}(h,k,c,n,m,f,g) = \mathsf{true}$ iff $h \le n$ and $k \le m$ or
$h > n$ and $k \le m$ and $g[r] < f[h]$ for all r, $k \le r < m$ or
$k > m$ and $h \le n$ and $f[r] < g[k]$ for all r, $h \le r < n$.

$$\begin{aligned}
&\textbf{function}\ \theta_{\mathsf{main}} : \mathsf{nat}\times\mathsf{nat}\times\mathsf{list}\times\mathsf{list}\longrightarrow\mathsf{bool}\\
&\quad \theta_{\mathsf{main}}(n,m,f,g) = \theta_{\mathsf{loop}}(0,0,0,n,m,f,g)
\end{aligned}$$

$\theta_{\mathsf{main}}(n,m,f,g) = \mathsf{true}$.

References

[AG97] T. Arts and J. Giesl. Proving innermost normalisation automatically. In *Proc. RTA-97*, Sitges, Spain, LNCS 1232, 1997.

[BM79] R. S. Boyer and J S. Moore. *A computational logic.* Academic Press, 1979.

[BR95] A. Bouhoula and M. Rusinowitch. Implicit induction in conditional theories. *Journal of Automated Reasoning*, 14:189–235, 1995.

[BG96] J. Brauburger and J. Giesl. Termination analysis for partial functions. In *Proc. 3rd Int. Static Analysis Symp.*, Aachen, Germany, LNCS 1145, 1996. Extended version appeared as Report IBN-96-33, TU Darmstadt, 1996. Available from `http://www.inferenzsysteme.informatik.tu-darmstadt.de/~reports/notes/ibn-96-33.ps`

[Bra97] J. Brauburger. Automatic termination analysis for partial functions using polynomial orderings. In *Proc. 4th SAS*, Paris, France, LNCS 1302, 1997.

[Bun$^+$89] A. Bundy, F. van Harmelen, J. Hesketh, A. Smaill, and A. Stevens. A rational reconstruction and extension of recursion analysis. In *Proc. IJCAI '89*, Detroit, USA, 1989.

[Der87] N. Dershowitz. Termination of rewriting. *J. Symb. Comp.*, 3:69-115, 1987.

[Gie95a] J. Giesl. Generating polynomial orderings for termination proofs. In *Proc. RTA-95*, Kaiserslautern, Germany, LNCS 914, 1995.

[Gie95b] J. Giesl. *Automatisierung von Terminierungsbeweisen für rekursiv definierte Algorithmen.* PhD thesis, Infix-Verlag, St. Augustin, Germany, 1995.

[Gie95c] J. Giesl. Termination analysis for functional programs using term orderings. In *Proc. 2nd Int. Static Analysis Symp.*, Glasgow, UK, LNCS 983, 1995.

[Gie97] J. Giesl. Termination of nested and mutually recursive algorithms. *Journal of Automated Reasoning*, 19:1-29, 1997.

[GWB98] J. Giesl, C. Walther, and J. Brauburger. Termination analysis for functional programs. In W. Bibel and P. Schmitt, editors, *Automated Deduction*, vol. 3. Kluwer, 1998. To appear.

[Gri81] D. Gries. *The Science of Programming.* Springer-Verlag, New York, 1981.

[Hen80] P. Henderson. *Functional Programming.* Prentice-Hall, London, 1980.

[HS96] D. Hutter and C. Sengler. INKA: The next generation. In *Proc. CADE-13*, New Brunswick, USA, LNAI 1104, 1996.

[KZ95] D. Kapur and H. Zhang. An overview of Rewrite Rule Laboratory (RRL). *J. Computer Math. Appl.*, 29:91-114, 1995.

[Lan79] D.S. Lankford. On proving term rewriting systems are noetherian. Memo MTP-3, Math. Dept., Louisiana Tech. Univ., Ruston, USA, 1979.

[MW78] Z. Manna and R. Waldinger. Is 'sometimes' sometimes better than 'always'. *Comm. of the ACM*, 21(2):159-172, 1978.

[NN96] F. Nielson and H. R. Nielson. Operational semantics of termination types. *Nordic Journal of Computing*, 3(2):144–187, 1996.

[PS97] S. E. Panitz and M. Schmidt-Schauß. TEA: Automatically proving termination of programs in a non-strict higher-order functional language. In *Proc. 4th Int. Static Analysis Symp.*, Paris, France, LNCS 1302, 1997.

[Plü90] L. Plümer. *Termination proofs for logic programs.* LNAI 446, 1990.

[Pro96] M. Protzen. Patching faulty conjectures. In *Proc. CADE-13*, LNAI 1104, New Brunswick, USA, 1996.

[SD94] D. De Schreye and S. Decorte. Termination of logic programs: The never-ending story. *Journal of Logic Programming*, 19/20:199–260, 1994.

[Sen96] C. Sengler. Termination of algorithms over non-freely generated data types. In *Proc. CADE-13*, New Brunswick, USA, LNAI 1104, 1996.

[Ste94] J. Steinbach. Generating polynomial orderings. *IPL*, 49:85-93, 1994.

[Ste95] J. Steinbach. Simplification orderings: History of results. *Fundam. Informaticae*, 24:47-87, 1995.

[UvG88] J. D. Ullman and A. van Gelder. Efficient tests for top-down termination of logical rules. *Journal of the ACM*, 35(2):345-373, 1988.

[Wal94a] C. Walther. Mathematical induction. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 2. Oxford University Press, 1994.

[Wal94b] C. Walther. On proving the termination of algorithms by machine. *Artificial Intelligence*, 71(1):101–157, 1994.

[ZKK88] H. Zhang, D. Kapur, and M. S. Krishnamoorthy. A mechanizable induction principle for equational specifications. In *Proc. CADE-9*, Argonne, USA, LNCS 310, 1988.