Formalizing Termination Proofs under Polynomial Quasi-interpretations

Naohi Eguchi*
Department of Mathematics and Informatics
Chiba University, Japan
neguchi@g.math.s.chiba-u.ac.jp

Usual termination proofs for a functional program require to check all the possible reduction paths. Due to an exponential gap between the height and size of such the reduction tree, no naive formalization of termination proofs yields a connection to the polynomial complexity of the given program. We solve this problem employing the notion of minimal function graph, a set of pairs of a term and its normal form, which is defined as the least fixed point of a monotone operator. We show that termination proofs for programs reducing under lexicographic path orders (LPOs for short) and polynomially quasi-interpretable can be optimally performed in a weak fragment of Peano arithmetic. This yields an alternative proof of the fact that every function computed by an LPO-terminating, polynomially quasi-interpretable program is computable in polynomial space. The formalization is indeed optimal since every polynomial-space computable function can be computed by such a program. The crucial observation is that inductive definitions of minimal function graphs under LPO-terminating programs can be approximated with transfinite induction along LPOs.

1 Introduction

1.1 Motivation

The termination of a program states that any reduction under the program leads to a normal form. Recent developments in termination analysis of first order functional programs, or of *term rewrite systems* more specifically, have drawn interest in computational resource analysis, i.e., not just the termination but also the estimation of time/space-resources required to execute a given program, which includes the polynomial run-space complexity analysis. Usual termination proofs for a program require to check all the possible reduction paths under the program. Due to an exponential gap between the *height* and *size* of such the reduction tree, no naive termination proof yields a connection to the polynomial complexity of the given program. For the sake of optimal termination proofs, it seems necessary to discuss "all the possible reduction paths" by means of an alternative notion smaller in size than reduction trees.

1.2 Backgrounds

Stemming from [21], there are various functional characterizations of polynomial-space computable functions [14, 16, 17, 9], Those characterizations state that every poly-space computable function can be defined by a finite set of equations, i.e., by a functional program. Orienting those equations suitably, such programs reduce under a termination order, the *lexicographic path orders* (LPOs for short). The well-founded-ness of LPOs yields the termination of the reducing programs.

*The author is supported by Grants-in-Aid for JSPS Fellows (Grant No. 25 · 726).

R. Matthes, M. Mio (Eds.): Fixed Points
in Computer Science 2015 (FICS 2015)
EPTCS 191, 2015, pp. 33–47, doi:10.4204/EPTCS.191.5

© Naohi Eguchi
This work is licensed under the
Creative Commons Attribution License.

In the seminal work [5], it was discussed, depending on the choice of a termination order, what mathematical axiom is necessary to formalize termination proofs by the termination order within Peano arithmetic PA that axiomatizes ordered semi-rings with mathematical induction. In case of *multiset path orders* (MPOs for short), termination proofs can be formalized in the fragment of PA with induction restricted to computably enumerable sets. This yields an alternative proof of the fact that every function computed by an MPO-terminating program is primitive recursive, cf. [10]. The formalization is optimal since every primitive recursive function can be computed by an MPO-terminating program. In case of LPOs, termination proofs can be formalized in the fragment with induction restricted to expressions of the form "f is total" for some computable function f. The formalization is optimal in the same sense as in case of MPOs, cf. [22].

In more recent works [3, 4], MPOs and LPOs are combined with *polynomial quasi-interpretations* (PQIs for short). Unlike (strict) polynomial interpretations [2], the existence of a quasi-interpretation does not tell us anything about termination. However, combined with these termination orders, the PQI can be a powerful method in computational resource analysis. Indeed, those functional programs characterizing poly-space computable functions that was motioned above admit PQIs. This means that every poly-space computable function can be computed by an LPO-terminating program that admits a PQI. Moreover, conversely, every function computed by such a program is computable in polynomial space [3, Theorem 1].

1.3 Outline

In Section 2 we fix the syntax of first order functional programs and the semantics in accordance with the syntax. In Section 3 we present the definitions of LPOs and PQIs together with some examples, stating an application to poly-space computable functions (Theorem 1, [3, Theorem 1]). In Section 4 we present the framework of formalization. For an underlying formal system, a second order system U_2^1 of *bounded arithmetic* [6], which can be regarded as a weak fragment of PA, seems suitable since it is known that the system U_2^1 is complete for poly-space computable functions (Theorem 2.2).

In [5], the termination of a program reducing under an LPO $<_{\mathsf{lpo}}$ is deduced by showing that, given a term t, a tree containing all the possible reduction chains starting with t is well founded under $<_{\mathsf{lpo}}$. The same construction of such reduction trees does not work in U_2^1 essentially because the exponentiation $m \mapsto 2^m$ is not available. We lift the problem employing the notion of *minimal function graph* [12, 11, 15], a set of pairs of a term and its normal form. Given a term t, instead of constructing a reduction tree rooted at t, we construct a (subset of a) minimal function graph that stores the pair of t and a normal form of t. Typically, a minimal function graph is inductively defined, or in other words defined as the least fixed point of a monotone operator. Let us recall that the set of natural numbers is the least fixed point of the operator $m \in \Gamma(X) \Longleftrightarrow m = 0 \vee \exists n \in X$ s.t. $m = n+1$. As seen from this example, many instances of inductive definitions are induced by operators of the form $t \in \Gamma(X) \Longleftrightarrow \exists s_1, \ldots, s_k \in X \cdots$. Crucially, a minimal function graph under a program reducing under an LPO $<_{\mathsf{lpo}}$ can be defined as the least fixed point of such an operator but also $t \in \Gamma(X) \Longleftrightarrow \exists s_1, \ldots, s_k \in X \wedge s_1, \ldots, s_k <_{\mathsf{lpo}} t \cdots$ holds. Thanks to the additional condition $s_1, \ldots, s_k <_{\mathsf{lpo}} t$, the minimal function graphs under the program can be defined by $<_{\mathsf{lpo}}$-transfinite induction as well as inductive definitions. In Section 5 this idea is discussed in more details.

In the main section, Section 6, the full details about the formalization are given. Most of the effort is devoted to deduce in U_2^1 an appropriate form of transfinite induction along LPOs (Lemma 5). Based on the idea above, we then construct a minimal function graph G for a given program $\mathbf{R}$ reducing under an LPO $<_{\mathsf{lpo}}$ by $<_{\mathsf{lpo}}$-transfinite induction (Theorem 3). Since G stores all the pairs of a term and its

R-normal form, this means the termination of the program **R**.

In Section 7 it is shown that the formalization presented in Section 6 yields that every function computed by an LPO-terminating program that admits a PQI is poly-space computable (Corollary 3). This shows that the formalization is optimal since such programs can only compute poly-space computable functions as mentioned in Section 1.2.

2 Syntax and semantics of first order functional programs

Throughout the paper, a *program* denotes a *term rewrite system*. We sometimes use unusual notations or formulations for the sake of simplification. More precise, widely accepted formulations can be found, e.g., in [20].

Definition 1 (Constuctor-, basic-, terms, rewrite rules, sizes of terms)**.** Let $\mathbf{C}$ and $\mathbf{D}$ be disjoint finite signatures, respectively of *constructors* and *defined* symbols, and $\mathbf{V}$ a countably infinite set of *variables*. We assume that $\mathbf{C}$ contains at least one constant. The sets $\mathbf{T}(\mathbf{C}\cup\mathbf{D},\mathbf{V})$ of *terms*, $\mathbf{T}(\mathbf{C},\mathbf{V})$ of *constructor* terms, $\mathbf{B}(\mathbf{C}\cup\mathbf{D},\mathbf{V})$ of *basic* terms and $\mathbf{R}(\mathbf{C}\cup\mathbf{D},\mathbf{V})$ of *rewrite rules* are distinguished as follows.

$$\begin{array}{lrcll}
\text{(Terms)} & t & ::= & x \mid c(t_1,\ldots,t_l) \mid f(t_1,\ldots,t_l) & \in \mathbf{T}(\mathbf{C}\cup\mathbf{D},\mathbf{V});\\
\text{(Constructor terms)} & s & ::= & x \mid c(s_1,\ldots,s_k) & \in \mathbf{T}(\mathbf{C},\mathbf{V});\\
\text{(Basic terms)} & u & ::= & f(s_1,\ldots,s_k) & \in \mathbf{B}(\mathbf{C}\cup\mathbf{D},\mathbf{V});\\
\text{(Rewrite rules)} & \rho & ::= & u \to t & \in \mathbf{R}(\mathbf{C}\cup\mathbf{D},\mathbf{V}),
\end{array}$$

where $x\in\mathbf{V}$, $c\in\mathbf{C}$, $f\in\mathbf{D}$, $t,t_1,\ldots,t_l\in\mathbf{T}(\mathbf{C}\cup\mathbf{D},\mathbf{V})$, $s_1,\ldots,s_k\in\mathbf{T}(\mathbf{C},\mathbf{V})$ and $u\in\mathbf{B}(\mathbf{C}\cup\mathbf{D},\mathbf{V})$. For such a class $\mathbf{S}(\mathbf{F},\mathbf{V})$ of terms, $\mathbf{S}(\mathbf{F})$ denotes the subset of closed terms. The *size* $\|t\|$ of a term t is defined as $\|x\|=1$ for a variable x and $\|f(t_1,\ldots,t_k)\|=1+\sum_{j=1}^{k}\|t_j\|$.

Definition 2 (Substitutions, quasi-reducible programs, rewrite relations)**.** A *program* $\mathbf{R}$ is a finite subset of $\mathbf{R}(\mathbf{C}\cup\mathbf{D},\mathbf{V})$ consisting of rewrite rules of the form $l\to r$ such that the variables occurring in r occur in l as well. A mapping $\theta:\mathbf{V}\to\mathbf{S}(\mathbf{F},\mathbf{V})$ from variables to a set $\mathbf{S}(\mathbf{F},\mathbf{V})$ of terms is called a *substitution*. For a term $t\in\mathbf{S}(\mathbf{F},\mathbf{V})$, $t\theta$ denotes the result of replacing every variable x with $\theta(x)$. A program $\mathbf{R}$ is *quasi-reducible* if, for any closed basic term $t\in\mathbf{B}(\mathbf{C}\cup\mathbf{F})$, there exist a rule $l\to r\in\mathbf{R}$ and a substitution $\theta:\mathbf{V}\to\mathbf{T}(\mathbf{C})$ such that $t=l\theta$. We restrict reductions to those under *call-by-value* evaluation, or *innermost* reductions more precisely. For three terms t,u,v, we write $t[u/v]$ to denote the result of replacing an occurrence of v with u. It will not be indicated which occurrence of v is replaced if no confusion likely arises. We write $t\xrightarrow[\mathbf{R}]{\mathrm{i}}s$ if $s=t[r\theta/l\theta]$ holds for some rule $l\to r\in\mathbf{R}$ and constructor substitution $\theta:\mathbf{V}\to\mathbf{T}(\mathbf{C})$. We write $\xrightarrow[\mathbf{R}]{\mathrm{i}}{}^{*}$ to denote the reflexive and transitive closure of $\xrightarrow[\mathbf{R}]{\mathrm{i}}$ and $t\xrightarrow[\mathbf{R}]{\mathrm{i}}{}^{!}s$ if $t\xrightarrow[\mathbf{R}]{\mathrm{i}}{}^{*}s$ and s is a normal form. By definition, for any quasi-reducible program $\mathbf{R}$, if $t\xrightarrow[\mathbf{R}]{\mathrm{i}}{}^{!}s$ and t is closed, then $s\in\mathbf{T}(\mathbf{C})$ holds.

A program $\mathbf{R}$ *computes* a function if any closed basic term has a unique normal form in $\mathbf{T}(\mathbf{C})$. In this case, for every k-ary function symbol $f\in\mathbf{D}$, a function $[\![f]\!]:\mathbf{T}(\mathbf{C})^k\to\mathbf{T}(\mathbf{C})$ is defined by $[\![f]\!](s_1,\ldots,s_k)=s\iff f(s_1,\ldots,s_k)\xrightarrow[\mathbf{R}]{\mathrm{i}}{}^{!}s$.

3 Lexicographic path orders and quasi-interpretations

Lexicographic path orders are *recursive path orders* with lexicographic status only, whose variant was introduced in [13]. Recursive path orders with multiset status only were introduced in [8] and a modern formulation with both multiset and lexicographic status can be found in [20, page 211]. Let $<_{\mathbf{F}}$ be a

(strict) *precedence*, a well-founded partial order on a signature $\mathbf{F} = \mathbf{C} \cup \mathbf{D}$. We always assume that every constructor is $<_{\mathbf{F}}$-minimal. The *lexicographic path order* (LPO for short) $<_{\mathsf{lpo}}$ induced by $<_{\mathbf{F}}$ is defined recursively by the following three rules.

1. $\dfrac{s \leqslant_{\mathsf{lpo}} t_i}{s <_{\mathsf{lpo}} g(t_1,\ldots,t_l)}$ $(i \in \{1,\ldots,l\})$
2. $\dfrac{s_1 <_{\mathsf{lpo}} g(t_1,\ldots,t_l) \quad \cdots \quad s_k <_{\mathsf{lpo}} g(t_1,\ldots,t_l)}{f(s_1,\ldots,s_k) <_{\mathsf{lpo}} g(t_1,\ldots,t_l)}$ $(f <_{\mathbf{F}} g \in \mathbf{D})$
3. $\dfrac{s_1 = t_1 \ \cdots \ s_{i-1} = t_{i-1} \quad s_i <_{\mathsf{lpo}} t_i \quad s_{i+1} <_{\mathsf{lpo}} t \ \cdots \ s_k <_{\mathsf{lpo}} t}{f(s_1,\ldots,s_k) <_{\mathsf{lpo}} f(t_1,\ldots,t_k) = t}$ $(f \in \mathbf{D})$

We say that a program $\mathbf{R}$ *reduces under* $<_{\mathsf{lpo}}$ if $r <_{\mathsf{lpo}} l$ holds for each rule $l \to r \in \mathbf{R}$ and that $\mathbf{R}$ is *LPO-terminating* if there exists an LPO under which $\mathbf{R}$ reduces. We write $s <^{\langle i \rangle}_{\mathsf{lpo}} t$ if $s <_{\mathsf{lpo}} t$ results as an instance of the above i^{th} case ($i = 1,2,3$). Corollary 1 is a consequence of the definition of LPOs, following from $<_{\mathbf{F}}$-minimality of constructors.

Corollary 1. *If $s <_{\mathsf{lpo}} t$ and $t \in \mathbf{T}(\mathbf{C})$, then $s <^{\langle 1 \rangle}_{\mathsf{lpo}} t$ and $s \in \mathbf{T}(\mathbf{C})$.*

A *quasi-interpretation* $(\!|\cdot|\!)$ for a signature $\mathbf{F}$ is a mapping from $\mathbf{F}$ to functions over naturals fulfilling (i) $(\!|f|\!) : \mathbb{N}^k \to \mathbb{N}$ for each k-ary function symbol $f \in \mathbf{F}$, (ii) $(\!|f|\!)(\ldots,m,\ldots) \leq (\!|f|\!)(\ldots,n,\ldots)$ whenever $m < n$, (iii) $m_j \leq (\!|f|\!)(m_1,\ldots,m_k)$ for any $j \in \{1,\ldots,k\}$, and (iv) $0 < (\!|f|\!)$ if f is a constant. A quasi-interpretation $(\!|\cdot|\!)$ for a signature $\mathbf{F}$ is extended to closed terms $\mathbf{T}(\mathbf{F})$ by $(\!|f(t_1,\ldots,t_k)|\!) = (\!|f|\!)((\!|t_1|\!),\ldots,(\!|t_k|\!))$. Such an interpretation $(\!|\cdot|\!)$ is called a quasi-interpretation for a program $\mathbf{R}$ if $(\!|r\theta|\!) \leq (\!|l\theta|\!)$ holds for each rule $l \to r \in \mathbf{R}$ and for any constructor substitution $\theta : \mathbf{V} \to \mathbf{T}(\mathbf{C})$. A program $\mathbf{R}$ *admits a polynomial quasi-interpretation* (PQI for short) if there exists a quasi-interpretation $(\!|\cdot|\!)$ for $\mathbf{R}$ such that $(\!|f|\!)$ is polynomially bounded for each $f \in \mathbf{F}$. A PQI $(\!|\cdot|\!)$ is called *kind* 0 (or *additive* [4]) if, for each constructor $c \in \mathbf{C}$, $(\!|c|\!)(m_1,\ldots,m_k) = d + \sum_{j=1}^{k} m_j$ holds for some constant $d > 0$. An LPO-terminating program $\mathbf{R}$ is called an $LPO^{Poly(0)}$-program if $\mathbf{R}$ admits a kind 0 PQI.

Theorem 1 ([3])**.** *Every function computed by an $LPO^{Poly(0)}$-program is computable in polynomial space.*

Conversely, every polynomial-space computable function can be computed by an $\mathrm{LPO}^{\mathrm{Poly}(0)}$-program [3, Theorem 1]. In [4] various examples of programs admitting (kind 0) PQIs are illustrated, including $\mathrm{LPO}^{\mathrm{Poly}(0)}$-programs $\mathbf{R}_{\mathsf{lcs}}$ and $\mathbf{R}_{\mathsf{QBF}}$ below.

Example 1. The length of the *longest common subsequences* of two strings can be computed by a program $\mathbf{R}_{\mathsf{lcs}}$ [4, Example 6], which consists of the following rewrite rules defined over a signature $\mathbf{F} = \mathbf{C} \cup \mathbf{D}$ where $\mathbf{C} = \{0,\mathsf{s},\varepsilon,\mathsf{a},\mathsf{b}\}$ and $\mathbf{D} = \{\mathsf{max},\mathsf{lcs}\}$.

$$
\begin{array}{llll}
\mathsf{max}(x,0) \to x & \mathsf{max}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{s}(\mathsf{max}(x,y)) & & \\
\mathsf{max}(0,y) \to y & & & \\
\mathsf{lcs}(x,\varepsilon) \to 0 & \mathsf{lcs}(\mathsf{i}(x),\mathsf{i}(y)) \to \mathsf{s}(\mathsf{lcs}(x,y)) & & (\mathsf{i} \in \{\mathsf{a},\mathsf{b}\}) \\
\mathsf{lcs}(\varepsilon,y) \to 0 & \mathsf{lcs}(\mathsf{i}(x),\mathsf{j}(y)) \to \mathsf{max}(\mathsf{lcs}(x,\mathsf{j}(y)),\mathsf{lcs}(\mathsf{i}(x),y)) & & (\mathsf{i} \neq \mathsf{j} \in \{\mathsf{a},\mathsf{b}\})
\end{array}
$$

Natural numbers are built of 0 and s and strings of a and b as $\mathsf{a}(u) = \mathsf{a}u$ for a string $u \in \{\mathsf{a},\mathsf{b}\}^*$. The symbol ε denotes the empty string. Define a precedence $<_{\mathbf{F}}$ on $\mathbf{F}$ by $\mathsf{max} <_{\mathbf{F}} \mathsf{lcs}$. Assuming that every constructor is $<_{\mathbf{F}}$-minimal, the program $\mathbf{R}_{\mathsf{lcs}}$ reduces under the LPO $<_{\mathsf{lpo}}$ induced by $<_{\mathbf{F}}$. For instance, the orientation $\mathsf{max}(\mathsf{lcs}(x,\mathsf{b}(y)),\mathsf{lcs}(\mathsf{a}(x),y)) <_{\mathsf{lpo}} \mathsf{lcs}(\mathsf{a}(x),\mathsf{b}(y))$ can be deduced as follows. The orientation $y <^{\langle 1 \rangle}_{\mathsf{lpo}} \mathsf{b}(y)$ yields $\mathsf{lcs}(\mathsf{a}(x),y) <^{\langle 3 \rangle}_{\mathsf{lpo}} \mathsf{lcs}(\mathsf{a}(x),\mathsf{b}(y))$ while $x <^{\langle 1 \rangle}_{\mathsf{lpo}} \mathsf{a}(x)$ and $\mathsf{b}(y) <^{\langle 1 \rangle}_{\mathsf{lpo}} \mathsf{lcs}(\mathsf{a}(x),\mathsf{b}(y))$ yield

$\mathsf{lcs}(x,\mathsf{b}(y)) <^{\langle 3\rangle}_{\mathsf{lpo}} \mathsf{lcs}(\mathsf{a}(x),\mathsf{b}(y))$. These together with $\mathsf{max} <_{\mathbf{F}} \mathsf{lcs}$ yield $\mathsf{max}(\mathsf{lcs}(x,\mathsf{b}(y)),\mathsf{lcs}(\mathsf{a}(x),y)) <^{\langle 2\rangle}_{\mathsf{lpo}} \mathsf{lcs}(\mathsf{a}(x),\mathsf{b}(y))$. It can be seen that the program $\mathbf{R}_{\mathsf{lcs}}$ admits the kind 0 PQI $(\!|\cdot|\!)$ defined by

$$\begin{aligned}(\!|0|\!) = (\!|\varepsilon|\!) &= 1,\\ (\!|\mathsf{s}|\!)(x) = (\!|\mathsf{a}|\!)(x) = (\!|\mathsf{b}|\!)(x) &= 1+x,\\ (\!|\mathsf{max}|\!)(x,y) = (\!|\mathsf{lcs}|\!)(x,y) &= \max(x,y).\end{aligned}$$

This is exemplified as $(\!|\mathsf{max}(\mathsf{lcs}(x,\mathsf{b}(y)),\mathsf{lcs}(\mathsf{a}(x),y))|\!) = \max\big(\max(x,1+y),\max(1+x,y)\big) \le \max(1+x,1+y) = (\!|\mathsf{lcs}(\mathsf{a}(x),\mathsf{b}(y))|\!)$. Thus Theorem 1 implies that the function $[\![\mathsf{lcs}]\!]$ can be computed in polynomial space.

Example 2. The *Quantified Boolean Formula* (QBF) problem can be solved by a program $\mathbf{R}_{\mathsf{QBF}}$ [4, Example 36], which consists of the following rewrite rules defined over a signature $\mathbf{F} = \mathbf{C} \cup \mathbf{D}$ where $\mathbf{C} = \{0,\mathsf{s},\mathsf{nil},\mathsf{cons},\top,\bot,\mathsf{var},\neg,\vee,\exists\}$ and $\mathbf{D} = \{=,\mathsf{not},\mathsf{or},\mathsf{in},\mathsf{verify},\mathsf{qbf}\}$.

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{not}(\top) & \to & \bot & \mathsf{not}(\bot) & \to & \top\\
\mathsf{or}(\top,x) & \to & \top & \mathsf{or}(\bot,x) & \to & x\\
0=0 & \to & \top & \mathsf{s}(x)=0 & \to & \bot\\
0=\mathsf{s}(x) & \to & \bot & \mathsf{s}(x)=\mathsf{s}(y) & \to & x=y\\
\mathsf{in}(x,\mathsf{nil}) & \to & \bot & \mathsf{in}(x,\mathsf{cons}(y,ys)) & \to & \mathsf{or}(x=y,\mathsf{in}(x,ys))
\end{array}$$

$$\begin{array}{rcl}
\mathsf{verify}(\mathsf{var}(x),xs) & \to & \mathsf{in}(x,xs)\\
\mathsf{verify}(\neg x,xs) & \to & \mathsf{not}(\mathsf{verify}(x,xs))\\
\mathsf{verify}(x\vee y,xs) & \to & \mathsf{or}\,(\mathsf{verify}(x,xs),\mathsf{verify}(y,xs))\\
\mathsf{verify}\,((\exists x)y,xs) & \to & \mathsf{or}\,(\mathsf{verify}(y,\mathsf{cons}(x,xs)),\mathsf{verify}(y,xs))\\
\mathsf{qbf}(x) & \to & \mathsf{verify}(x,\mathsf{nil})
\end{array}$$

The symbol $\top$ denotes the true Boolean value while $\bot$ the false one. Boolean variables are encoded with $\{0,\mathsf{s}\}$-terms, i.e., with naturals. Formulas are built from variables operating var, $\neg$, $\vee$ or $\exists$. Without loss of generality, we can assume that every QBF is built up in this way. As usual, terms of the forms $=(s,t)$, $\neg(t)$, $\vee(s,t)$ and $\exists(s,t)$ are respectively denoted as $s=t$, $\neg t$, $s\vee t$ and $(\exists s)t$. By definition, for a Boolean formula φ with Boolean variables $x_1,\ldots,x_k$, $[\![\mathsf{verify}]\!](\varphi,[\cdots]) = \top$ holds if and only if φ is true with the truth assignment that $x_j = \top$ if x_j appears in the list $[\cdots]$ and $x_j = \bot$ otherwise.

Define a precedence $<_{\mathbf{F}}$ over $\mathbf{F}$ by $\mathsf{not},\mathsf{or},= <_{\mathbf{F}} \mathsf{in} <_{\mathbf{F}} \mathsf{verify} <_{\mathbf{F}} \mathsf{qbf}$. Assuming $<_{\mathbf{F}}$-minimality of constructor, the program $\mathbf{R}_{\mathsf{QBF}}$ reduces under the LPO $<_{\mathsf{lpo}}$ induced by $<_{\mathbf{F}}$. For instance, the orientation $\mathsf{or}(\mathsf{verify}(y,\mathsf{cons}(x,xs)),\mathsf{verify}(y,xs)) <_{\mathsf{lpo}} \mathsf{verify}(\exists(x,y),xs)$ can be deduced as follows. As well as $xs <^{\langle 1\rangle}_{\mathsf{lpo}} \mathsf{verify}(\exists(x,y),xs)$, the orientation $x <^{\langle 1\rangle}_{\mathsf{lpo}} \exists(x,y)$ yields $x <^{\langle 1\rangle}_{\mathsf{lpo}} \mathsf{verify}(\exists(x,y),xs)$. These together with the assumption $\mathsf{cons} <_{\mathbf{F}} \mathsf{verify}$ yield $\mathsf{cons}(x,xs) <^{\langle 2\rangle}_{\mathsf{lpo}} \mathsf{verify}(\exists(x,y),xs)$. This together with $y <^{\langle 1\rangle}_{\mathsf{lpo}} \exists(x,y)$ yields $\mathsf{verify}(y,\mathsf{cons}(x,xs)) <^{\langle 3\rangle}_{\mathsf{lpo}} \mathsf{verify}(\exists(x,y),xs)$ as well as $\mathsf{verify}(y,xs) <^{\langle 3\rangle}_{\mathsf{lpo}} \mathsf{verify}(\exists(x,y),xs)$. These orientations together with the assumption $\mathsf{or} <_{\mathbf{F}} \mathsf{verify}$ now allow us to deduce the desired orientation $\mathsf{or}(\mathsf{verify}(y,\mathsf{cons}(x,xs)),\mathsf{verify}(y,xs)) <^{\langle 2\rangle}_{\mathsf{lpo}} \mathsf{verify}(\exists(x,y),xs)$.

Furthermore, let us define a PQI $(\!|\cdot|\!)$ for the signature $\mathbf{F}$ by

$$\begin{aligned}
(\!|c|\!) &= 1 && \text{if } c \text{ is a constant,}\\
(\!|x_1,\ldots,x_k|\!) &= 1+\textstyle\sum_{j=1}^{k} x_j && \text{if } c \in \mathbf{C} \text{ with arity} > 0,\\
(\!|f|\!)(x_1,\ldots,x_k) &= \max_{j=1}^{k} x_j && \text{if } f \in \mathbf{D}\setminus\{\mathsf{verify},\mathsf{qbf}\},\\
(\!|\mathsf{verify}|\!)(x,y) &= x+y, &&\\
(\!|\mathsf{qbf}|\!)(x) &= x+1. &&
\end{aligned}$$

Clearly the PQI $(\!|\cdot|\!)$ is kind 0. Then the program $\mathbf{R}_{\mathsf{QBF}}$ admits the PQI. This is exemplified by the rule above as $(\!|\mathsf{or}(\mathsf{verify}(y,\mathsf{cons}(x,xs)),\mathsf{verify}(y,xs))|\!) = \max\big(y+(1+x+xs),y+xs\big) = (1+x+y)+xs = (\!|\mathsf{verify}(\exists(x,y),xs)|\!)$. Thus Theorem 1 implies that the function $\lceil\mathsf{qbf}\rceil$ can be computed in polynomial space. This is consistent with the well known fact that the QBF problem is PSPACE-complete.

4 A system U_2^1 of second order bounded arithmetic

In this section, we present the basics of second order bounded arithmetic following [1]. The original formulation is traced back to [6]. The non-logical language $\mathbf{L}_{\mathsf{BA}}$ of first order bounded arithmetic consists of the constant 0, the successor S, the addition $+$, the multiplication $\cdot$, $|x| = \lceil \log_2(x+1) \rceil$, the division by two $\lfloor x/2 \rfloor$, the smash $\#(x,y) = 2^{|x|\cdot|y|}$ and $\leq$. It is easy to see that $|m|$ is equal to the number of bits in the binary representation of a natural m. In addition to these usual symbols, we assume that the language $\mathbf{L}_{\mathsf{BA}}$ contains $\max(x,y)$. The assumption makes no change if an underlying system is sufficiently strong.

Definition 3 (Sharply-, bounded quantifiers, bounded formulas, S_2^1)**.** Quantifiers of the form $\exists x(x \leq t \wedge \cdots)$ or $\forall x(x \leq t \to \cdots)$ for some term t are called *bounded* and quantifiers of the form $(Qx \leq |t|)\cdots$ are called *sharply* bounded. *Bounded formulas* contain no unbounded first order quantifiers. The classes Σ_i^{b} ($i \in \mathbb{N}$) of bounded formulas are defined by counting the number of alternations of bounded quantifiers starting with an existential one, but ignoring sharply bounded ones. For each $i \in \mathbb{N}$, the first order system S_2^i of bounded arithmetic is axiomatized with a set BASIC of open axioms defining the $\mathbf{L}_{\mathsf{BA}}$-symbols together with the schema (Σ_i^{b}-PIND) of bit-wise induction for Σ_i^{b}-formulas.

$$\varphi(0) \wedge \forall x\big(\varphi(\lfloor x/2 \rfloor) \to \varphi(x)\big) \to \forall x \varphi(x) \quad (\varphi \in \Phi) \tag{Φ-PIND}$$

The precise definition of the basic axioms BASIC can be found, e.g., in [7, page 101].

Definition 4 (Second order bounded formulas, U_2^1)**.** In addition to the first order language, the language of second order bounded arithmetic contains second order variables $X, Y, Z, \ldots$ ranging over sets and the membership relation $\in$. In contrast to the classes Σ_i^{b}, the classes $\Sigma_i^{\mathrm{b},1}$ of second order bounded formulas are defined by counting alternations of second order quantifiers starting with an existential one, but ignoring first order ones. By definition, $\Sigma_0^{\mathrm{b},1}$ is the class of bounded formulas with no second order quantifiers. The second order system U_2^1 is axiomatized with BASIC, ($\Sigma_1^{\mathrm{b},1}$-PIND) and the axiom ($\Sigma_0^{\mathrm{b},1}$-CA) of comprehension for $\Sigma_0^{\mathrm{b},1}$-formulas.

$$\forall \vec{x}\, \forall \vec{X}\, \exists Y (\forall y \leq t)\big(y \in Y \leftrightarrow \varphi(y, \vec{x}, \vec{X})\big) \quad (\varphi \in \Phi) \tag{Φ-CA}$$

Unlike first order ones, second order quantifiers have no explicit bounding. However, due to the presence of a bounding term t in the schema ($\Sigma_0^{\mathrm{b},1}$-CA), one can only deduce the existence of a set with a bounded domain.

Example 3. The axiom ($\Sigma_0^{\mathrm{b},1}$-CA) of comprehension allows us to transform given sets $\vec{X}$ into another set Y via $\Sigma_0^{\mathrm{b},1}$-definable operations without inessential encodings. For an easy example, assume that two sets U and V encode binary strings respectively of length m and n in such a way that $j \in U \Leftrightarrow$ "the j^{th} bit of the string U is 1" and $j \notin U \Leftrightarrow$ "the j^{th} bit of the string U is 0" for each $j < m$. Then the *concatenation* $W = U^\frown V$, the string U followed by V, is defined by ($\Sigma_0^{\mathrm{b},1}$-CA) as follows.

$$(\forall j < m+n)\big[j \in W \leftrightarrow \big((j < m \wedge j \in U) \vee (m \leq j \wedge j - m \in V)\big)\big]$$

Definition 5 (Definable functions in formal systems)**.** Let T be one of the formal systems defined above and Φ be a class of bounded formulas. A function $f : \mathbb{N}^k \to \mathbb{N}$ is Φ-*definable in* T if there exists a formula $\varphi(x_1,\ldots,x_k,y) \in \Phi$ with no other free variables such that $\varphi(\vec{x},y)$ expresses the relation $f(\vec{x}) = y$ (under the standard semantics) and T proves the sentence $\forall \vec{x}\, \exists! y \varphi(\vec{x},y)$.

Theorem 2 ([6])**.** *1. A function is* Σ^{b}_1*-definable in* S^1_2 *if and only if it is computable in polynomial time.*

2. A function is $\Sigma^{\mathrm{b},1}_1$*-definable in* U^1_2 *if and only if it is computable in polynomial space.*

To readers who are not familiar with second order bounded arithmetic, it might be of interest to outline the proof that every polynomial-space computable function can be defined in U^1_2. The argument is commonly known as the *divide-and-conquer* method, which was originally used to show the classical inclusion NPSPACE $\subseteq$ PSPACE [18].

Proof of the "if" direction of Theorem 2.2 (Outline). Suppose that a function $f : \mathbb{N}^k \to \mathbb{N}$ is computable in polynomial space. This means that there exist a deterministic Turing machine M and a polynomial $p : \mathbb{N}^k \to \mathbb{N}$ such that, for any inputs $m_1,\ldots,m_k$, $f(m_1,\ldots,m_k)$ can be computed by M while the head of M only visits a number of cells bounded by $p(|m_1|,\ldots,|m_k|)$. Then, since the number of possible configurations under M on inputs $m_1,\ldots,m_k$ is bounded by $2^{q(|m_1|,\ldots,|m_k|)}$ for some polynomial q, the computation terminates in a step bounded by $2^{q(|\vec{m}|)}$ as well.

Let $\psi_M(m_1,\ldots,m_k,n,w_0,W)$ denote a $\Sigma^{\mathrm{b},1}_0$-formula expressing that the set W encodes the concatenation $w_1{}^\frown \cdots {}^\frown w_{2^{|n|}}$ of configurations under M, where w_j is the next configuration of w_{j-1}, writing $w_j = \mathsf{Next}_M(w_{j-1})$ $(1 \le j \le 2^{|n|})$. Reasoning informally in U^1_2, the $\Sigma^{\mathrm{b},1}_1$-formula $\varphi(\vec{m},n) :\equiv (\forall w \le 2^{p(|\vec{m}|)}) \exists W \psi_M(\vec{m},n,w,W)$ can be deduced by $(\Sigma^{\mathrm{b},1}_1\text{-}\mathsf{PIND})$ on n. In case $n = 0$, W can be defined identical to $\mathsf{Next}_M(w)$. For the induction step, given a configuration $w_0 \le 2^{p(|\vec{m}|)}$, the induction hypothesis yields a set U such that $\psi_M(\vec{m},\lfloor n/2 \rfloor, w_0, U)$ holds. Another instance of the induction hypothesis yields a set V such that $\psi_M(\vec{m},\lfloor n/2 \rfloor, w_{2^{|n|-1}}, V)$ holds. Since $2^{|n|} = 2^{|n|-1} + 2^{|n|-1}$, $\psi_M(\vec{m},n,w_0,W)$ holds for the set $W := U{}^\frown V$.

Now instantiating n with $2^{q(|\vec{m}|)}$ yields a set W such that $\psi_M(\vec{m},2^{q(|\vec{m}|)},\mathsf{Init}_M(\vec{m}),W)$ holds for the initial configuration $\mathsf{Init}_M(\vec{m})$ on inputs $\vec{m}$. The set W yields the final configuration and thus the result $f(\vec{m})$ of the computation. The uniqueness of the result can be deduced in U^1_2 accordingly. □

The "only if" direction of Theorem 2.2 follows from a bit more general statement.

Lemma 1. *If* U^1_2 *proves* $\exists y \varphi(x_1,\ldots,x_k,y)$ *for a* $\Sigma^{\mathrm{b},1}_1$*-formula* $\varphi(x_1,\ldots,x_k,y)$ *with no other free variables, then there exists a function* $f : \mathbb{N}^k \to \mathbb{N}$ *such that, for any naturals* $\vec{m} = m_1,\ldots,m_k \in \mathbb{N}$, (i) $f(\vec{m})$ *is computable with the use of space bounded by a polynomial in* $|m_1|,\ldots,|m_k|$*, and* (ii) $\varphi(\underline{\vec{m}}, \underline{f(\vec{m})})$ *holds under the standard semantics, where* $\underline{m}$ *denotes the numeral* $\mathrm{S}^m(0)$ *for a natural* m*.*

It is also known that the second order system axiomatized with the schema $(\Sigma^{\mathrm{b},1}_1\text{-}\mathsf{IND})$, instead of $(\Sigma^{\mathrm{b},1}_1\text{-}\mathsf{PIND})$, of the usual induction $\varphi(0) \wedge \forall x(\varphi(x) \to \varphi(\mathrm{S}(x))) \to \forall x \varphi(x)$ for $\Sigma^{\mathrm{b},1}_1$-formulas, called V^1_2, captures the exponential-time computable functions of polynomial growth rate in the sense of Theorem 2. Though there is no common notion about what is bounded arithmetic, the exponential function $m \mapsto 2^m$ is not definable in any existing system of bounded arithmetic.

5 Minimal function graphs

The *minimal function graph* semantics was described in [12] as denotational semantics, cf. [23, Chapter 9], and afterward used for termination analysis of functional programs without exponential size-

explosions in [11, Chapter 24.2] and [15]. In this section, we explain how minimal function graphs work, how they are defined inductively, and how they can be defined without inductive definitions.

To see how minimal function graphs work, consider the program $\mathbf{R}_{\mathsf{lcs}}$ in Example 1. Let us observe that the following reduction starting with the basic term $\mathsf{lcs}(\mathsf{a}(\mathsf{a}(\varepsilon)),\mathsf{b}(\mathsf{b}(\varepsilon)))$ is possible.

$$\begin{array}{ll} & \mathsf{lcs}(\mathsf{a}(\mathsf{a}(\varepsilon)),\mathsf{b}(\mathsf{b}(\varepsilon))) \\ \xrightarrow{\mathsf{i}}_{\mathbf{R}_{\mathsf{lcs}}} & \mathsf{max}(\mathsf{lcs}(\mathsf{a}(\varepsilon),\mathsf{b}(\mathsf{b}(\varepsilon))),\mathsf{lcs}(\mathsf{a}(\mathsf{a}(\varepsilon)),\mathsf{b}(\varepsilon))) \\ \xrightarrow{\mathsf{i}}_{\mathbf{R}_{\mathsf{lcs}}} & \mathsf{max}(\mathsf{lcs}(\mathsf{a}(\varepsilon),\mathsf{b}(\mathsf{b}(\varepsilon))),\mathsf{max}(\mathsf{lcs}(\mathsf{a}(\varepsilon),\mathsf{b}(\varepsilon)),\mathsf{lcs}(\mathsf{a}(\mathsf{a}(\varepsilon)),\varepsilon))) \\ \xrightarrow{\mathsf{i}}_{\mathbf{R}_{\mathsf{lcs}}} & \mathsf{max}(\mathsf{max}(\mathsf{lcs}(\varepsilon,\mathsf{b}(\mathsf{b}(\varepsilon))),\mathsf{lcs}(\mathsf{a}(\varepsilon),\mathsf{b}(\varepsilon))),\mathsf{max}(\mathsf{lcs}(\mathsf{a}(\varepsilon),\mathsf{b}(\varepsilon)),\mathsf{lcs}(\mathsf{a}(\mathsf{a}(\varepsilon)),\varepsilon))) \end{array}$$

In the reduction, the term $t := \mathsf{lcs}(\mathsf{a}(\varepsilon),\mathsf{b}(\varepsilon))$ is duplicated, and hence costly re-computations potentially occur. For the same reason, there can be an exponential explosion in the size of the reduction tree rooted at $\mathsf{lcs}(\mathsf{a}(\mathsf{a}(\varepsilon)),\mathsf{b}(\mathsf{b}(\varepsilon)))$ that contains all the possible rewriting sequences starting with the basic term. A minimal function graph G, or *cache* in other words, is defined so that G stores pairs of a basic term and its normal form. Thus, once the term t is normalized to 0 (because the two strings a and b have no common subsequence), the pair $\langle t,0\rangle$ is stored in G and any other reduction of t can be simulated by replacing the occurrence of t with 0.

Given a program $\mathbf{R}$, a (variant of) minimal function graph G is defined as the least fixed point of the following operator Γ over $\mathscr{P}(\mathbf{B}(\mathbf{F})\times\mathbf{T}(\mathbf{C}))$, where $X \subseteq \mathbf{B}(\mathbf{F})\times\mathbf{T}(\mathbf{C})$.

$$\begin{array}{rcl} \langle t,s\rangle \in \Gamma(X) & :\Longleftrightarrow & \exists l\to r\in\mathbf{R}, \exists\theta:\mathbf{V}\to\mathbf{T}(\mathbf{C}), \exists\langle t_0,s_0\rangle,\ldots,\langle t_{\|r\|-1},s_{\|r\|-1}\rangle\in X \text{ s.t.} \\ & & t = l\theta \;\&\; s = \big((r\theta)[s_0/t_0]\cdots\big)[s_{\|r\|-1}/t_{\|r\|-1}] \end{array}$$

The operator Γ is monotone, i.e., $X\subseteq Y \Rightarrow \Gamma(X)\subseteq\Gamma(Y)$, and hence there exists the least fixed point of Γ. Suppose that $\mathbf{R}$ is quasi-reducible. On one side, the fixed-ness of G yields that $t \xrightarrow{\mathsf{i}}{}^{!}_{\mathbf{R}} s \Rightarrow \langle t,s\rangle\in G$. On the other side, since the set $\{\langle t,s\rangle \mid t\in\mathbf{B}(\mathbf{F}) \;\&\; t\xrightarrow{\mathsf{i}}{}^{!}_{\mathbf{R}} s\}$ is a fixed point of Γ, the least-ness of G yields that $\langle t,s\rangle\in G \Rightarrow t\xrightarrow{\mathsf{i}}{}^{!}_{\mathbf{R}} s$. Thus, to conclude that every closed basic term has an (innermost) $\mathbf{R}$-normal form, it suffices to show that, for every term $t\in\mathbf{B}(\mathbf{F})$, there exists a term s such that $\langle t,s\rangle\in G$. Now there are two important observations.

1. It suffices to show that, for every term $t\in\mathbf{B}(\mathbf{F})$, there exist a *subset* $G_t\subseteq G$ and a term s such that $\langle t,s\rangle\in G_t$. If $t = l\theta$ and $s = \big((r\theta)[s_0/t_0]\cdots\big)[s_{\|r\|-1}/t_{\|r\|-1}]$ as in the definition of Γ above and, for each $j<\|r\|$, $\langle t_j,s_j\rangle\in G_{t_j}$ holds for such a set $G_{t_j}\subseteq G$, then G_t can be simply defined as $G_t = \{\langle t,s\rangle\}\cup G_{t_0}\cup\cdots\cup G_{t_{\|r\|-1}}$.[1]
2. Additionally suppose that the program $\mathbf{R}$ reduces under an LPO $<_{\mathsf{lpo}}$. Then it turns out that the definition of Γ is equivalent to a form restricted in such a way that $t_j <_{\mathsf{lpo}} t$ for each $j<\|r\|$.[2]

For these reasons, the schema $(\forall t\in\mathbf{B}(\mathbf{F}))\big((\forall s<_{\mathsf{lpo}} t)\varphi(s)\to\varphi(t)\big)\to(\forall t\in\mathbf{B}(\mathbf{F}))\varphi(t)$ of transfinite induction along $<_{\mathsf{lpo}}$ will imply the termination of a quasi-reducible LPO-terminating program $\mathbf{R}$ in the sense above.

6 Formalizing LPO-termination proofs under PQIs in U^1_2

In this section, we show that, if $\mathbf{R}$ is a quasi-reducible $\mathrm{LPO}^{\mathrm{Poly}(0)}$-program, then an innermost $\mathbf{R}$-normal form of any closed basic term can be found in the system U^1_2 (Theorem 3).

[1] To be precise, in [11, 15], the *minimal function graph* was used to denote such a subset G_t for a given basic t.

[2] Namely, every function computed by an $<_{\mathsf{lpo}}$-reducing program is defined recursively along $<_{\mathsf{lpo}}$. Therefore, as a reviewer pointed out, in this case the minimal function graphs can be regarded as fixed-point semantics for recursive definitions of functions, cf. [19, Chapter 10].

Given a program $\mathbf{R}$ over a signature $\mathbf{F} = \mathbf{C} \cup \mathbf{D}$, we use the notation $V_{\mathbf{R}}$ to denote the finite set $\{x \in \mathbf{V} \mid x \text{ appears in some rule } \rho \in \mathbf{R}\}$ of variables. Let $\ulcorner \cdot \urcorner$ be an *efficient* binary encoding for $\mathbf{T}(\mathbf{F}, \mathbf{V}_{\mathbf{R}})$-terms. The efficiency means that:

(i) $t \mapsto \ulcorner t \urcorner$ is $\Sigma_0^{\mathrm{b},1}$-definable in U_2^1.

(ii) There exists a polynomial (term) $p(x)$ with a free variable x such that $|\ulcorner t \urcorner| \leq p(\|t\|)$ (provably) holds for any $t \in \mathbf{T}(\mathbf{F}, \mathbf{V}_{\mathbf{R}})$.

Without loss of generality, we can assume that:

(iii) $\|t\| \leq |\ulcorner t \urcorner|$.

(iv) $|\ulcorner s \urcorner| < |\ulcorner t \urcorner|$ if s is a proper subterm of t.

Such an encoding can be defined, for example, by representing terms as directed graphs not as trees.

Lemma 2. *The relation $<_{\mathsf{lpo}}$ is $\Sigma_0^{\mathrm{b},1}$-definable in U_2^1.*

Proof (Sketch). It suffices to show that, given two terms s and t, the relation "there exists a derivation tree according to the rules 1–3 (on page 36) that results in $s <_{\mathsf{lpo}} t$" is $\Sigma_0^{\mathrm{b},1}$-definable in U_2^1. Let T denote such a derivation tree resulting in $s <_{\mathsf{lpo}} t$. By induction according to the inductive definition of $<_{\mathsf{lpo}}$ it can be shown that the number of nodes in T is bounded by $\|s\| \cdot \|t\|$. Hence, by the assumption (ii) on the encoding $\ulcorner \cdot \urcorner$, the code $\ulcorner T \urcorner$ of T is polynomially bounded in $\|s\| \cdot \|t\|$ and thus in $\ulcorner s \urcorner \cdot \ulcorner t \urcorner$. On the other hand, by definition, the relation $s_0 <_{\mathsf{lpo}} t_0$ between two terms s_0 and t_0 is reduced to a tuple $s_j <_{\mathsf{lpo}} t_j$ $(j = 1, \ldots, k)$ of relations between some subterms $s_1, \ldots, s_k$ of s_0 and subterms $t_1, \ldots, t_k$ of t_0. Thanks to the assumption (iv) on the encoding $\ulcorner \cdot \urcorner$, $|\ulcorner s_j \urcorner| + |\ulcorner t_j \urcorner| < |\ulcorner s_0 \urcorner| + |\ulcorner t_0 \urcorner|$, i.e., $2^{|\ulcorner s_j \urcorner| + |\ulcorner t_j \urcorner|} \leq \lfloor \left(2^{|\ulcorner s_0 \urcorner| + |\ulcorner t_0 \urcorner|}\right)/2 \rfloor$, holds for any $j \in \{1, \ldots, k\}$. From these observations, it can be seen that the construction of the derivation tree T is performed in U_2^1, and hence the relation $s <_{\mathsf{lpo}} t$ is $\Sigma_0^{\mathrm{b},1}$-definable in U_2^1. □

As observed in [5], in which an optimal LPO-termination proof was described, every program $\mathbf{R}$ reducing under an LPO $<_{\mathsf{lpo}}$ already reduces under a finite restriction $<_\ell$ of $<_{\mathsf{lpo}}$ for some $\ell \in \mathbb{N}$ and every quantifier of the form $(Qs <_\ell t)$ can be regarded as a bounded one. Adopting the restriction, we introduce an even more restrictive relation $<_\ell$ $(\ell \in \mathbb{N})$ motivated by the following properties of PQIs.

Proposition 1. *Let $(\!| \cdot |\!)$ be a kind 0 PQI and $t \in \mathbf{B}(\mathbf{F})$. Then the following two properties hold.*

1. *$(\!|t|\!) \leq p(|\ulcorner t \urcorner|)$ holds for some polynomial p.*
2. *Suppose additionally that a program $\mathbf{R}$ admits the PQI $(\!| \cdot |\!)$ and that $t \xrightarrow{\mathrm{i}}{}^*_{\mathbf{R}} s$ holds. If $s \in \mathbf{T}(\mathbf{C})$, then $\|s\| \leq (\!|t|\!)$ holds. If $s = f(s_1, \ldots, s_k) \in \mathbf{B}(\mathbf{F})$, then $\|s_j\| \leq (\!|t|\!)$ holds for each $j \in \{1, \ldots, k\}$.*

Proof. PROPERTY 1. Let $t = g(t_1, \ldots, t_l)$. Since the PQI $(\!| \cdot |\!)$ is kind 0, one can find a constant d depending only on the set $\mathbf{C}$ of constructors and the PQI $(\!| \cdot |\!)$ such that $(\!|t_j|\!) \leq d \cdot \|t_j\|$ holds for any $j \in \{1, \ldots, l\}$. This yields a polynomial p such that $(\!|t|\!) \leq p(\|t\|)$ and thus $(\!|t|\!) \leq p(|\ulcorner t \urcorner|)$ holds by the assumption (iii) on the encoding $\ulcorner \cdot \urcorner$.

PROPERTY 2. In case $s \in \mathbf{T}(\mathbf{C})$, $\|s\| \leq (\!|s|\!) \leq (\!|t|\!)$ holds. In case $s = f(s_1, \ldots, s_k) \in \mathbf{B}(\mathbf{F})$, $\|s_j\| \leq (\!|s_j|\!) \leq (\!|s|\!) \leq (\!|t|\!)$ holds for each $j \in \{1, \ldots, k\}$. □

Definition 6 ($\mathbf{T}_\ell(\mathbf{C})$, $\mathbf{B}_\ell(\mathbf{F})$, $<_\ell$, $<_\ell^{\mathsf{lex}}$). Let $\mathbf{T}_\ell(\mathbf{C})$ denote a set $\{t \in \mathbf{T}(\mathbf{C}) \mid \|t\| \leq \ell\}$ of constructor terms and $\mathbf{B}_\ell(\mathbf{F})$ a set $\{f(t_1, \ldots, t_k) \in \mathbf{B}(\mathbf{F}) \mid \|t_1\|, \ldots, \|t_k\| \leq \ell\}$ of basic terms. Then we write $s <_\ell t$ if $s <_{\mathsf{lpo}} t$ and additionally $s \in \mathbf{T}_\ell(\mathbf{C}) \cup \mathbf{B}_\ell(\mathbf{F})$ hold. We use the notation $s <_\ell^{\langle i \rangle} t$ $(i = 1, 2, 3)$ accordingly. Moreover, we define a *lexicographic extension* $<_\ell^{\mathsf{lex}}$ of $<_\ell$ over $\mathbf{T}(\mathbf{C})$. For constructor terms $s_1, \ldots, s_k, t_1, \ldots, t_k$, we write $(s_1, \ldots, s_k) <_\ell^{\mathsf{lex}} (t_1, \ldots, t_k)$ if there exists an index $i \in \{1, \ldots, k\}$ such that $s_j = t_j$ for every $j < i$, $s_i <_\ell^{\langle 1 \rangle} t_i$, and $s_j \in \mathbf{T}_\ell(\mathbf{C})$ for every $j > i$.

Corollary 2 follows from the definitions of $<_\ell$ and $<_\ell^{\mathsf{lex}}$ and from $<_{\mathbf{F}}$-minimality of constructors.

Corollary 2. *For two basic terms $f(s_1,\dots,s_k), f(t_1,\dots,t_k) \in \mathbf{B}_\ell(\mathbf{F})$, $f(s_1,\dots,s_k) <_\ell^{\langle 3\rangle} f(t_1,\dots,t_k)$ holds if and only if $(s_1,\dots,s_k) <_\ell^{\mathsf{lex}} (t_1,\dots,t_k)$ holds.*

For most of interesting $\mathrm{LPO}^{\mathrm{Poly}(0)}$-programs including Example 1 and 2, interpreting polynomials consist of $+$, $\cdot$, $\max_{j=1}^k x_j$ together with additional constants. This motivates us to formalize PQIs limiting interpreting polynomial terms to those built up only from 0, S, $+$, $\cdot$ and max to make the formalization easier. Then the constraints (ii) and (iii) on PQIs follow from defining axioms for these function symbols.

Let us consider a reduction $t_0 \xrightarrow{\mathrm{i}}_{\mathbf{R}}^* t \xrightarrow{\mathrm{i}}_{\mathbf{R}}^* s$ under a program $\mathbf{R}$ admitting a kind 0 PQI $(\!|\cdot|\!)$, where $t_0, t \in \mathbf{B}(\mathbf{F})$ and $s \in \mathbf{T}(\mathbf{C}) \cup \mathbf{B}(\mathbf{F})$. If $s <_{\mathsf{lpo}} t$ for some LPO $<_{\mathsf{lpo}}$, then Proposition 1 yields a polynomial p such that $s <_{p(|\ulcorner t_0 \urcorner|)} t$ holds by Definition 6. Hence we can assume that ℓ is (the result of substituting t_0 for) a polynomial $p(|x|)$. More precisely, ℓ can be expressed by an $\mathbf{L}_{\mathsf{BA}}$-term built up from 0 and $|x|, |y|, |z|, \dots$ by S, $+$ and $\cdot$. By assumption, ℓ does not contain # nor $\lfloor \cdot/2 \rfloor$. Thus $\ell = \ell(x_1,\dots,x_k)$ denotes a polynomial with non-negative coefficients in $|x_1|,\dots,|x_k|$. Since ℓ contains no smash # in particular, $2^{p(\ell)}$ can be regarded as an $\mathbf{L}_{\mathsf{BA}}$-term for any polynomial $p(x)$. By the assumption (ii) on the encoding $\ulcorner \cdot \urcorner$, $|\ulcorner t \urcorner|$ is polynomially bounded in the size $\|t\|$ of t, and hence $\ulcorner t \urcorner \le 2^{p(\|t\|)}$ for some polynomial $p(x)$. Therefore any quantifier of the forms $(Qs <_\ell t)$, $(Qt \in \mathbf{T}_\ell(\mathbf{C}))$ and $(Qt \in \mathbf{B}_\ell(\mathbf{F}))$ can be treated as a bounded one.

We deduce the schema $(\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{B}_\ell(\mathbf{F}), <_\ell))$ of $<_\ell$-transfinite induction over $\mathbf{B}_\ell(\mathbf{F})$ for $\Sigma_1^{\mathrm{b},1}$-formulas (Lemma 5). Since the relation $f(s_1,\dots,s_k) <_\ell^{\langle 3\rangle} f(t_1,\dots,t_k)$ relies on the comparison $(s_1,\dots,s_k) <_\ell^{\mathsf{lex}} (t_1,\dots,t_k)$ by Corollary 2, we previously have to deduce the schema $(\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{T}_\ell(\mathbf{C})^k, <_\ell^{\mathsf{lex}}))$ of $<_\ell^{\mathsf{lex}}$-transfinite induction over k-tuples of $\mathbf{T}_\ell(\mathbf{C})$-terms (Lemma 4). We start with deducing the instance in the base case $k = 1$.

Lemma 3. *The following schema of $<_\ell$-transfinite induction over $\mathbf{T}_\ell(\mathbf{C})$ holds in U_2^1, where $\varphi \in \Sigma_1^{\mathrm{b},1}$.*

$$(\forall t \in \mathbf{T}_\ell(\mathbf{C}))\big((\forall s <_\ell t)\varphi(s) \to \varphi(t)\big) \to (\forall t \in \mathbf{T}_\ell(\mathbf{C}))\varphi(t) \qquad (\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{T}_\ell(\mathbf{C}), <_\ell))$$

Proof. Reason in U_2^1. Suppose $(\forall t \in \mathbf{T}_\ell(\mathbf{C}))\big((\forall s <_\ell t)\varphi(s) \to \varphi(t)\big)$ and let $t \in \mathbf{T}_\ell(\mathbf{C})$. We show that $\varphi(t)$ holds by $(\Sigma_1^{\mathrm{b},1}\text{-PIND})$ on $\ulcorner t \urcorner$. The case $\ulcorner t \urcorner = 0$ trivially holds. Suppose $\ulcorner t \urcorner > 0$ for induction step. By assumption, it suffices to show that $\varphi(s)$ holds for any $s <_\ell t$. Thus let $s <_\ell t$. Since $t \in \mathbf{T}_\ell(\mathbf{C})$, s is a proper subterm of t by Corollary 1 and $<_{\mathbf{F}}$-minimality of constructors. Thus, the assumption (iv) on the encoding $\ulcorner \cdot \urcorner$ yields $\ulcorner s \urcorner \le \lfloor \ulcorner t \urcorner/2 \rfloor$, and hence $\varphi(s)$ holds by induction hypothesis. □

Remark 1. In the proof of Lemma 3, we employed a bit-wise form of *course of values* induction $\varphi(0) \wedge \forall t\big(\forall s(\ulcorner s \urcorner \le \lfloor \ulcorner t \urcorner/2 \rfloor \to \varphi(s)) \to \varphi(t)\big) \to \forall t \varphi(t)$ for a $\Sigma_1^{\mathrm{b},1}$-formula $\varphi(x)$, which is not an instance of $(\Sigma_1^{\mathrm{b},1}\text{-PIND})$. Formally, one should apply $(\Sigma_1^{\mathrm{b},1}\text{-PIND})$ for the $\Sigma_1^{\mathrm{b},1}$-formula $\psi(x) \equiv \forall t\big(\ulcorner t \urcorner \le 2^{|x|} \to \varphi(t)\big)$ to deduce $(\forall t \in \mathbf{T}_\ell(\mathbf{C}))\,\varphi(t)$. To ease presentation, we will use similar informal arguments in the sequel.

Lemma 4. *The schema $(\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{T}_\ell(\mathbf{C}), <_\ell))$ can be extended to tuples of $\mathbf{T}_\ell(\mathbf{C})$-terms, i.e., the following schema holds in U_2^1, where $\varphi(\vec{t}) \equiv \varphi(t_1,\dots,t_k) \in \Sigma_1^{\mathrm{b},1}$.*

$$(\forall \vec{t} \in \mathbf{T}_\ell(\mathbf{C}))\big((\forall \vec{s} <_\ell^{\mathsf{lex}} \vec{t})\varphi(\vec{s}) \to \varphi(\vec{t})\big) \to (\forall \vec{t} \in \mathbf{T}_\ell(\mathbf{C}))\varphi(\vec{t}) \qquad (\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{T}_\ell(\mathbf{C})^k, <_\ell^{\mathsf{lex}}))$$

Proof. We show that the schema $(\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{T}_\ell(\mathbf{C})^k, <_\ell^{\mathsf{lex}}))$ holds in U_2^1 by (meta) induction on $k \ge 1$. In case $k = 1$, the schema is an instance of $(\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{T}_\ell(\mathbf{C}), <_\ell))$. Suppose that $k > 1$ and $(\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{T}_\ell(\mathbf{C})^{k-1}, <_\ell^{\mathsf{lex}}))$ holds by induction hypothesis. Assume that

$$(\forall t_1,\dots,t_k \in \mathbf{T}_\ell(\mathbf{C}))\big((\forall (s_1,\dots,s_k) <_\ell^{\mathsf{lex}} (t_1,\dots,t_k))\varphi(s_1,\dots,s_k) \to \varphi(t_1,\dots,t_k)\big) \qquad (1)$$

holds for some $\Sigma_1^{\mathrm{b},1}$-formula $\varphi(t_1,\dots,t_k)$. Let $\varphi_{<_\ell^{\mathsf{lex}}}(t,t_2,\dots,t_k)$, $\psi(t)$ and $\psi_{<_\ell}(t)$ denote $\Sigma_1^{\mathrm{b},1}$-formulas specified as follows.

$$\begin{array}{rcl}\varphi_{<_\ell^{\mathsf{lex}}}(t,t_2,\dots,t_k) & :\equiv & t_2,\dots,t_k \in \mathbf{T}_\ell(\mathbf{C}) \wedge \big(\forall(s_2,\dots,s_k) <_\ell^{\mathsf{lex}} (t_2,\dots,t_k)\big)\, \varphi(t,s_2,\dots,s_k);\\ \psi(t) & :\equiv & (\forall t_2,\dots,t_k \in \mathbf{T}_\ell(\mathbf{C}))\, \varphi(t,t_2,\dots,t_k);\\ \psi_{<_\ell}(t) & :\equiv & t \in \mathbf{T}_\ell(\mathbf{C}) \wedge (\forall s <_\ell t)\psi(s).\end{array}$$

Note, in particular, that $\psi(t)$ is still a $\Sigma_1^{\mathrm{b},1}$-formula since every quantifier of the form $(\forall s \in \mathbf{T}_\ell(\mathbf{C}))$ can be regarded as a bounded one under which the class $\Sigma_1^{\mathrm{b},1}$ is closed. One can see that $\varphi_{<_\ell^{\mathsf{lex}}}(t,t_2,\dots,t_k)$ and $\psi_{<_\ell}(t)$ imply $t,t_2,\dots,t_k \in \mathbf{T}_\ell(\mathbf{C})$ and $\big(\forall(s,s_2,\dots,s_k) <_\ell^{\mathsf{lex}} (t,t_2,\dots,t_k)\big)\varphi(s,s_2,\dots,s_k)$. Hence, by the assumption (1), $\psi_{<_\ell}(t)$ implies $(\forall t_2,\dots,t_k \in \mathbf{T}_\ell(\mathbf{C}))\big(\varphi_{<_\ell^{\mathsf{lex}}}(t,t_2,\dots,t_k) \to \varphi(t,t_2,\dots,t_k)\big)$, which denotes

$$(\forall t_2,\dots,t_k \in \mathbf{T}_\ell(\mathbf{C}))\big((\forall(s_2,\dots,s_k) <_\ell^{\mathsf{lex}} (t_2,\dots,t_k))\varphi(t,s_1,\dots,s_k) \to \varphi(t,t_2,\dots,t_k)\big).$$

This together with $(\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{T}_\ell(\mathbf{C})^{k-1},<_\ell^{\mathsf{lex}}))$ yields $(\forall t_2,\dots,t_k \in \mathbf{T}_\ell(\mathbf{C}))\, \varphi(t,t_2,\dots,t_k)$, denoting $\psi(t)$. This means that $(\forall t \in \mathbf{T}_\ell(\mathbf{C}))\big((\forall s <_\ell t)\psi(s) \to \psi(t)\big)$ holds. Since $\psi(t) \in \Sigma_1^{\mathrm{b},1}$ as noted above, this together with $(\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{T}_\ell(\mathbf{C}),<_\ell))$ yields $(\forall t \in \mathbf{T}_\ell(\mathbf{C}))\psi(t)$ and thus $(\forall t_1,\dots,t_k \in \mathbf{T}_\ell(\mathbf{C}))\varphi(t_1,\dots,t_k)$ holds. □

Lemma 5. *Let* $\mathbf{F} = \mathbf{C} \cup \mathbf{D}$. *The* $<_\ell$*-transfinite induction over* $\mathbf{B}_\ell(\mathbf{F})$ *holds in* U_2^1, *where* $\varphi \in \Sigma_1^{\mathrm{b},1}$.

$$(\forall t \in \mathbf{B}_\ell(\mathbf{F}))\big((\forall s \in \mathbf{B}_\ell(\mathbf{F}))(s <_\ell t \to \varphi(s)) \to \varphi(t)\big) \to (\forall t \in \mathbf{B}_\ell(\mathbf{F}))\varphi(t) \qquad (\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{B}_\ell(\mathbf{F}),<_\ell))$$

Given a precedence $<_{\mathbf{F}}$ on the finite signature $\mathbf{F}$, let $\mathsf{rk} : \mathbf{F} \to \mathbb{N}$ denote the *rank*, a finite function compatible with $<_{\mathbf{F}}$: $\mathsf{rk}(f) < \mathsf{rk}(g) \Leftrightarrow f <_{\mathbf{F}} g$.

Proof. Reason in U_2^1. Assume the premise of $(\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{B}_\ell(\mathbf{F}),<_\ell))$:

$$(\forall t \in \mathbf{B}_\ell(\mathbf{F}))\big((\forall s \in \mathbf{B}_\ell(\mathbf{F}))(s <_\ell t \to \varphi(s)) \to \varphi(t)\big) \tag{2}$$

Let $g \in \mathbf{D}$. We show that $(\forall t_1,\dots,t_l \in \mathbf{T}_\ell(\mathbf{C}))\varphi(g(t_1,\dots,t_l))$ holds by $(\Sigma_1^{\mathrm{b},1}\text{-}\mathsf{PIND})$ on $2^{\mathsf{rk}(g)}$, or in other words by finitary induction on $\mathsf{rk}(g)$. Let $t_1,\dots,t_l \in \mathbf{T}_\ell(\mathbf{C})$ and $t := g(t_1,\dots,t_l)$. By the assumption (2), it suffices to show that $\varphi(s)$ holds for any $s \in \mathbf{B}_\ell(\mathbf{F})$ such that $s <_\ell t$. Thus, let $s \in \mathbf{B}_\ell(\mathbf{F})$ and $s <_\ell t$.

Case. $s <_\ell^{\langle 1\rangle} t$: In this case $s \leqslant_\ell t_i$ for some $i \in \{1,\dots,l\}$. Since $t_i \in \mathbf{T}_\ell(\mathbf{C})$, $s \in \mathbf{T}_\ell(\mathbf{C})$ as well by Corollary 1, and hence this case is excluded.

Case. $s := f(s_1,\dots,s_k) <_\ell^{\langle 2\rangle} t$: In this case, $f <_{\mathbf{F}} g$ and hence $\mathsf{rk}(f) < \mathsf{rk}(g)$. This allows us to reason as $2^{\mathsf{rk}(g)} \leq 2^{\mathsf{rk}(f)-1} = \lfloor 2^{\mathsf{rk}(f)}/2 \rfloor$. Thus the induction hypothesis yields $\varphi(s)$.

Case. $s := g(s_1,\dots,s_l) <_\ell^{\langle 3\rangle} t$: We show that the following condition holds.

$$(\forall v_1,\dots,v_l \in \mathbf{T}_\ell(\mathbf{C}))\big((\forall(u_1,\dots,u_l) <_\ell^{\mathsf{lex}} (v_1,\dots,v_l))\varphi(g(u_1,\dots,u_l)) \to \varphi(g(v_1,\dots,v_l))\big) \tag{3}$$

Let $v_1,\dots,v_l \in \mathbf{T}_\ell(\mathbf{C})$. By Corollary 2, the premise $(\forall(u_1,\dots,u_l) <_\ell^{\mathsf{lex}} (v_1,\dots,v_l))\varphi(g(u_1,\dots,u_l)$ of (3) yields $\big(\forall s' <_\ell^{\langle 3\rangle} g(v_1,\dots,v_l)\big)\, \varphi(s')$. On the other side, the previous two cases yield $(\forall s' \in \mathbf{B}_\ell(\mathbf{F}))\big(s' <_\ell^{\langle i\rangle} g(v_1,\dots,v_l) \to \varphi(s')\big)$ $(i = 1,2)$ and hence $(\forall s' \in \mathbf{B}_\ell(\mathbf{F}))\big(s' <_\ell g(v_1,\dots,v_l) \to \varphi(s')\big)$ holds. Therefore $\varphi(g(v_1,\dots,v_l))$ holds by the assumption (2), yielding the statement (3). Since (3) is the premise of an instance of the schema $(\mathsf{TI}_{\Sigma_1^{\mathrm{b},1}}(\mathbf{T}_\ell(\mathbf{C})^l,<_\ell^{\mathsf{lex}}))$, Lemma 4 yields $(\forall v_1,\dots,v_l \in \mathbf{T}_\ell(\mathbf{C}))\varphi(g(v_1,\dots,v_l))$, and thus $\varphi(g(s_1,\dots,s_l))$ holds in particular. □

To derive, from $(\mathsf{TI}_{\Sigma_1^{b,1}}(\mathbf{B}_\ell(\mathbf{F}),<_\ell))$, the existence of a minimal function graph under an LPO-terminating program, we need the following technical lemma.

Lemma 6. (in U_2^1) *Let $(\!|\cdot|\!)$ be a kind 0 PQI for a signature $\mathbf{F}=\mathbf{C}\cup\mathbf{D}$, $t\in\mathbf{B}(\mathbf{F})$, $s\in\mathbf{T}(\mathbf{F})$ and $<_{\mathsf{lpo}}$ an LPO induced by a precedence $<_\mathbf{F}$. If $s<_{\mathsf{lpo}}t$ and $(\!|s|\!)\le(\!|t|\!)\le\ell$, then, for any basic subterm t' of s and for any $s'\in\mathbf{T}(\mathbf{C})$ such that $(\!|s'|\!)\le(\!|t'|\!)$, $v<_\ell t$ holds for any basic subterm v of $s[s'/t']$.*

Proof. By $<_\mathbf{F}$-minimality of constructors, $s'<_{\mathsf{lpo}}t'$ holds. Hence $s[s'/t']<_{\mathsf{lpo}}t$ from the assumption $s<_{\mathsf{lpo}}t$. This yields $v<_{\mathsf{lpo}}t$ by the definition of LPOs. Write $v=f(v_1,\ldots,v_k)$ for some $f\in\mathbf{D}$ and $v_1,\ldots,v_k\in\mathbf{T}(\mathbf{C})$. Let $i\in\{1,\ldots,k\}$. Then $\|v_i\|\le(\!|v_i|\!)\le(\!|v|\!)\le(\!|s[s'/t']|\!)\le(\!|t|\!)$. The last inequality follows from the monotonicity (ii) of the PQI $(\!|\cdot|\!)$. This yields $\|v_i\|\le\ell$ and hence $v<_\ell t$. □

Theorem 3. (in U_2^1) *Suppose that $\mathbf{R}$ is a quasi-reducible $\mathrm{LPO}^{Poly(0)}$-program. Then, for any basic term t, there exists a minimal function graph G (in the sense of Section 5) such that that $\langle t,s\rangle\in G$ holds for an $\mathbf{R}$-normal form s of t.*

Proof. Suppose that $\mathbf{R}$ is a quasi-reducible $\mathrm{LPO}^{\mathrm{Poly}(0)}$-program witnessed by an LPO $<_{\mathsf{lpo}}$ and a kind 0 PQI $(\!|\cdot|\!)$ and that $<_\ell$ is a finite restriction of $<_{\mathsf{lpo}}$. Let $\psi_\ell(x,y,X)$ denote a $\Sigma_0^{b,1}$-formula with no free variables other than x, y and X expressing that $X\subseteq\mathbf{B}_\ell(\mathbf{F})\times\mathbf{T}_\ell(\mathbf{C})$ is a set of pairs of terms such that $\langle x,y\rangle\in X$, and, for any $\langle t,s\rangle\in X$, $(\!|s|\!)\le(\!|t|\!)\le\ell$ and $\exists l\to r\in\mathbf{R}$, $\exists\theta:V_\mathbf{R}\to\mathbf{T}_\ell(\mathbf{C})$ s.t. $t=l\theta$ and one of the following cases holds.

1. $s=r\theta\in\mathbf{T}_\ell(\mathbf{C})$.
2. $\exists\big\langle\langle t_j,s_j\rangle\in X\mid j<\|r\|\big\rangle$ s.t. $s=\big((r\theta)[s_0/t_0]\cdots\big)[s_{\|r\|-1}/t_{\|r\|-1}]$, where $s'[u/v]$ is identical if no v occurs in s'.

Note that, since $V_\mathbf{R}$ is a finite set of variables, $\exists\theta:V_\mathbf{R}\to\mathbf{T}_\ell(\mathbf{C})$ can be regarded as a (first order) bounded quantifier. By Proposition 1.1, we can find a polynomial term $p(x)$ such that $(\!|t|\!)\le p(|\ulcorner t\urcorner|)$ holds for any $t\in\mathbf{B}(\mathbf{F})$. The rest of the proof is devoted to deduce $(\forall t\in\mathbf{B}(\mathbf{F}))(\exists s\in\mathbf{T}_\ell(\mathbf{C}))\exists G\,\psi_{p(|\ulcorner t\urcorner|)}(t,s,G)$ for such a bounding polynomial p. Fix an input basic term $t_0\in\mathbf{B}(\mathbf{F})$ and let $\varphi_\ell(t)$ denote the $\Sigma_1^{b,1}$-formula $(\exists s\in\mathbf{T}_\ell(\mathbf{C}))\exists G\,\psi_\ell(t,s,G)$, where $\ell=p(|\ulcorner t_0\urcorner|)$. Since $t_0\in\mathbf{B}_\ell(\mathbf{F})$, it suffices to deduce $(\forall t\in\mathbf{B}_\ell(\mathbf{F}))\varphi_\ell(t)$. By Lemma 5, this follows from $(\forall t\in\mathbf{B}_\ell(\mathbf{F}))\big((\forall s\in\mathbf{B}_\ell(\mathbf{F}))(s<_\ell t\to\varphi_\ell(s))\to\varphi_\ell(t)\big)$, which is the premise of an instance of $(\mathsf{TI}_{\Sigma_1^{b,1}}(\mathbf{B}_\ell(\mathbf{F}),<_\ell))$. Thus let $t\in\mathbf{B}_\ell(\mathbf{F})$ and assume the condition

$$(\forall s\in\mathbf{B}_\ell(\mathbf{F}))(s<_\ell t\to\varphi_\ell(s)). \tag{4}$$

Since $\mathbf{R}$ is quasi-reducible, there exist a rule $l\to r\in\mathbf{R}$ and a substitution $\theta:V_\mathbf{R}\to\mathbf{T}_\ell(\mathbf{C})$ such that $t=l\theta$. The remaining argument splits into two cases depending on the shape of $r\theta$.

CASE 1. $r\theta\in\mathbf{T}_\ell(\mathbf{C})$: In this case $\psi_\ell(t,r\theta,G)$ holds for the singleton $G:=\{\langle t,r\theta\rangle\}$.

CASE 2. $r\theta\notin\mathbf{T}_\ell(\mathbf{C})$: In this case there exists a basic subterm v_0 of $r\theta$. Fix a term $u_0\in\mathbf{T}_\ell(\mathbf{C})$ such that $(\!|u_0|\!)\le(\!|v_0|\!)$. We show the following claim by finitary induction on $m<\|r\|$.

Claim 1. *There exists a sequence $\big\langle\langle t_j,s_j,G_j\rangle\mid j\le m\big\rangle$ of triplets such that, for each $j\le m$,* (i) *$t_j<_\ell t$,* (ii) *$\psi_\ell(t_j,s_j,G_j)$ holds, and* (iii) *$\big((r\theta)[s_0/t_0]\cdots\big)[s_j/t_j]$ is not identical to $\big((r\theta)[s_0/t_0]\cdots\big)[s_{j-1}/t_{j-1}]$ as long as $\big((r\theta)[s_0/t_0]\cdots\big)[s_{j-1}/t_{j-1}]$ has a basic subterm.*

In the base case $m=0$, let t_0 be an arbitrary basic subterm of $r\theta$. Then, since $(\!|r\theta|\!)\le(\!|l\theta|\!)$, $t_0<_\ell t$ follows from the definition of LPOs. Hence, by the assumption (4), there exist a term $s_0\in\mathbf{T}_\ell(\mathbf{C})$ and a set G_0 such that $\psi_\ell(t_0,s_0,G_0)$ holds. Clearly, $(r\theta)[s_0/t_0]$ is not identical to $r\theta$. For induction step,

suppose that there exists a sequence $\langle\langle t_j, s_j, G_j\rangle \mid j \le m\rangle$ fulfilling the conditions (i)–(iii) in the claim. In case that $\big((r\theta)[s_0/t_0]\cdots\big)[s_m/t_m]$ has no basic subterm, let $(t_{m+1}, s_{m+1}) = (v_0, u_0)$. Otherwise, let t_{m+1} be an arbitrary basic subterm. Then $t_{m+1} <_\ell t$ holds by Lemma 6. Hence, as in the base case, the assumption (4) yields a term $s_{m+1} \in \mathbf{T}_\ell(\mathbf{C})$ and a set G_{m+1} such that $\psi_\ell(t_{m+1}, s_{m+1}, G_{m+1})$ holds. By the choice of t_{m+1}, $\big((r\theta)[s_0/t_0]\cdots\big)[s_{m+1}/t_{m+1}]$ is not identical to $\big((r\theta)[s_0/t_0]\cdots\big)[s_m/t_m]$.

Now let $s := \big((r\theta)[s_0/t_0]\cdots\big)[s_{\|r\|-1}/t_{\|r\|-1}]$ for a sequence $\langle\langle t_j, s_j, G_j\rangle \mid j < \|r\|\rangle$ witnessing the claim in case $m = \|r\| - 1$. Then $s \in \mathbf{T}_\ell(\mathbf{C})$ since $|\{f \in \mathbf{D} \mid f \text{ appears in } \big((r\theta)[s_0/t_0]\cdots\big)[s_j/t_j]\}| \le \|r\| - (j+1)$ holds for each $j < \|r\|$ by the condition (iii) in the claim. Defining a set G by $G = \{\langle t, s\rangle\} \cup \Big(\bigcup_{j<\|r\|} G_j\Big)$ now allows us to conclude $\psi_\ell(t, s, G)$. □

7 Application

In the last section, to convince readers that the formalization of termination proofs described in Theorem 3 for $\mathrm{LPO}^{\mathrm{Poly}(0)}$-programs is optimal, we show that the formalization yields an alternative proof of Theorem 1, i.e., that $\mathrm{LPO}^{\mathrm{Poly}(0)}$-programs can only compute polynomial-space computable functions.

The next lemma ensures that the set G constructed in Theorem 3 is indeed a minimal function graph.

Lemma 7. *Suppose that* $\mathbf{R}$ *is a quasi-reducible* $\mathrm{LPO}^{Poly(0)}$*-program. Let* $\psi_\ell(x, y, X)$ *denote the* $\Sigma_0^{\mathrm{b},1}$*-formula defined in the proof of Theorem 3. Then, for any* $t \in \mathbf{B}(\mathbf{F})$ *and for any* $t \in \mathbf{T}(\mathbf{C})$, $t \xrightarrow{\mathrm{i}}{}^{!}_{\mathbf{R}} s$ *if and only if* $\exists G\ \psi_{p(|\ulcorner t \urcorner|)}(t, s, G)$ *holds under the standard semantics.*

Proof. Let $\mathbf{R}$ reduce under an LPO $<_{\mathsf{lpo}}$. For the "if" direction, it can be shown that $(\forall t \in \mathbf{B}(\mathbf{F}))(\forall s \in \mathbf{T}(\mathbf{C}))\big(\exists G\ \psi_{p(|\ulcorner t \urcorner|)}(t, s, G) \Rightarrow t \xrightarrow{\mathrm{i}}{}^{!}_{\mathbf{R}} s\big)$ holds by (external) transfinite induction along $<_{\mathsf{lpo}}$. For the "only if" direction, it can be shown that $(\forall t \in \mathbf{B}(\mathbf{F}))(\forall s \in \mathbf{T}(\mathbf{C}))\big(t \xrightarrow{\mathrm{i}}{}^{m}_{\mathbf{R}} s \Rightarrow \exists G\ \psi_{p(|\ulcorner t \urcorner|)}(t, s, G)\big)$ holds by induction on m, where $\xrightarrow{\mathrm{i}}{}^{m}_{\mathbf{R}}$ denotes the m-fold iteration of $\xrightarrow{\mathrm{i}}_{\mathbf{R}}$. □

Now Theorem 3 and Lemma 7 yield an alternative proof of (a variant of) Theorem 1.

Corollary 3. *Every function computed by a quasi-reducible* $\mathrm{LPO}^{Poly(0)}$*-program is computable in polynomial space.*

Proof. By Theorem 3, U_2^1 proves the formula

$$\mathsf{QR}(\mathbf{R}) \wedge \mathsf{LPO}(\mathbf{R}, <_{\mathsf{lpo}}) \wedge \mathsf{PQI}(\mathbf{R}, (\!|\cdot|\!)) \to (\forall t \in \mathbf{B}(\mathbf{F}))(\exists s \in \mathbf{T}_{p(|\ulcorner t \urcorner|)}(\mathbf{C}))\exists G\ \psi_{p(|\ulcorner t \urcorner|)}(t, s, G),$$

where $\mathsf{QR}(\mathbf{R})$, $\mathsf{LPO}(\mathbf{R}, <_{\mathsf{lpo}})$ and $\mathsf{PQI}(\mathbf{R}, (\!|\cdot|\!))$ respectively express that any $\mathbf{B}(\mathbf{F})$-term is reducible, $\mathbf{R}$ reduces under $<_{\mathsf{lpo}}$, and $(\forall (l \to r) \in \mathbf{R})(\forall \theta : V_{\mathbf{R}} \to \mathbf{T}(\mathbf{C}))(\!|r\theta|\!) \le (\!|l\theta|\!)$. By Lemma 2, $\mathsf{LPO}(\mathbf{R}, <_{\mathsf{lpo}})$ can be expressed with a $\Sigma_0^{\mathrm{b},1}$-formula, but neither $\mathsf{QR}(\mathbf{R})$ nor $\mathsf{PQI}(\mathbf{R}, (\!|\cdot|\!))$ is literally expressible with a bounded formula. Nonetheless, the proof can be easily modified to a proof of the statement

$$(\forall t \in \mathbf{B}(\mathbf{F}))(\exists s \in \mathbf{T}_\ell(\mathbf{C}))\big(\mathsf{QR}_\ell(\mathbf{R}) \wedge \mathsf{LPO}(\mathbf{R}, <_{\mathsf{lpo}}) \wedge \mathsf{PQI}_\ell(\mathbf{R}, (\!|\cdot|\!)) \to \exists G\ \psi_\ell(t, s, G)\big),$$

where $\ell = p(|\ulcorner t \urcorner|)$, and $\mathsf{QR}_\ell(\mathbf{R})$ and $\mathsf{PQI}_\ell(\mathbf{R}, (\!|\cdot|\!))$ respectively express that any $\mathbf{B}_\ell(\mathbf{F})$-term is reducible, and $(\forall (l \to r) \in \mathbf{R})(\forall \theta : V_{\mathbf{R}} \to \mathbf{T}_\ell(\mathbf{C}))(\!|r\theta|\!) \le (\!|l\theta|\!)$. Both $\mathsf{QR}_\ell(\mathbf{R})$ and $\mathsf{PQI}_\ell(\mathbf{R}, (\!|\cdot|\!))$ can be regarded as $\Sigma_0^{\mathrm{b},1}$-formulas, and hence the formula $\varphi_\ell(t, s) :\equiv \mathsf{QR}_\ell(\mathbf{R}) \wedge \mathsf{LPO}(\mathbf{R}, <_{\mathsf{lpo}}) \wedge \mathsf{PQI}_\ell(\mathbf{R}, (\!|\cdot|\!)) \to \exists G\ \psi_\ell(t, s, G)$ lies in $\Sigma_1^{\mathrm{b},1}$.

Now suppose that a function $[\![\mathsf{f}]\!] : \mathbf{T}(\mathbf{C})^k \to \mathbf{T}(\mathbf{C})$ is computed by a quasi-reducible $\mathrm{LPO}^{\mathrm{Poly}(0)}$-program $\mathbf{R}$ for some k-ary function symbol $\mathsf{f} \in \mathbf{D}$. Then Lemma 1 yields a polynomial-space computable

function $f:\mathbb{N}^k\to\mathbb{N}$ such that $\varphi_{p(|\ulcorner \mathsf{f}(t_1,\ldots,t_k)\urcorner|)}\big(\mathsf{f}(t_1,\ldots,t_k),\underline{f(\ulcorner t_1\urcorner,\ldots,\ulcorner t_k\urcorner)}\big)$ holds for any $t_1,\ldots,t_k\in\mathbf{T}(\mathbf{C})$ under the standard semantics. Hence, by assumption, $\psi_{p(|\ulcorner \mathsf{f}(t_1,\ldots,t_k)\urcorner|)}\big(\mathsf{f}(t_1,\ldots,t_k),\underline{f(\ulcorner t_1\urcorner,\ldots,\ulcorner t_k\urcorner)},G\big)$ holds for some set $G\subseteq\mathbf{B}(\mathbf{F})\times\mathbf{T}(\mathbf{C})$. By Lemma 7, this means the correspondence $\llbracket\mathsf{f}\rrbracket(t_1,\ldots,t_k)=s\Leftrightarrow f(\ulcorner t_1\urcorner,\ldots,\ulcorner t_k\urcorner)=\ulcorner s\urcorner$. Therefore, $\ulcorner\llbracket\mathsf{f}\rrbracket(t_1,\ldots,t_k)\urcorner$ can be computed with space bounded by a polynomial in $|\ulcorner t_1\urcorner|,\ldots,|\ulcorner t_k\urcorner|$ and thus bounded by a polynomial in $\|t_1\|,\ldots,\|t_k\|$. □

8 Conclusion

This work is concerned with optimal termination proofs for functional programs in the hope of establishing logical foundations of computational resource analysis. Optimal termination proofs were limited for programs that compute functions lying in complexity classes closed under exponentiation. In this paper, employing the notion of minimal function graph, we showed that termination proofs under $\mathrm{LPO}^{\mathrm{Poly}(0)}$-programs can be optimally formalized in the second order system U^1_2 of bounded arithmetic that is complete for polynomial-space computable functions, lifting the limitation. The crucial idea is that inductive definitions of minimal function graphs under $\mathrm{LPO}^{\mathrm{Poly}(0)}$-programs can be approximated with transfinite induction along LPOs. As a small consequence, compared to the original result, Theorem 1, when we say "a program $\mathbf{R}$ computes a function", the quasi-reducibility of $\mathbf{R}$ is explicitly needed to enable the formalization.

Finally, let us call a program $\mathbf{R}$ an $\mathit{MPO}^{\mathit{Poly}(0)}$ one if $\mathbf{R}$ reduces under an MPO (with product status only) and $\mathbf{R}$ admits a kind 0 PQI. In [4, Theorem 42], Theorem 1 is refined so that a function can be computed by an $\mathrm{MPO}^{\mathrm{Poly}(0)}$-program if and only if it is computable in polynomial time. The program $\mathbf{R}_{\mathsf{lcs}}$ described in Example 1 is an example of $\mathrm{MPO}^{\mathrm{Poly}(0)}$-programs, and hence the length of the longest common subsequences is computable even in polynomial time. By Theorem 2.1, it is quite natural to expect that minimal function graphs under $\mathrm{MPO}^{\mathrm{Poly}(0)}$-programs can be constructed in the first order system S^1_2. However, we then somehow have to adopt the formula $\varphi_\ell(t,s)\equiv\mathsf{QR}_\ell(\mathbf{R})\wedge\mathsf{LPO}(\mathbf{R},<_{\mathsf{lpo}})\wedge\mathsf{PQI}_\ell(\mathbf{R},(\!|\cdot|\!))\to\exists G\ \psi_\ell(t,s,G)$ (in the proof of Corollary 3) to a Σ^{b}_1-formula, which is clearly more involved than the present case.

References

[1] A. Beckmann & S.R. Buss (2014): *Improved Witnessing and Local Improvement Principles for Second-order Bounded Arithmetic*. *ACM Transactions on Computational Logic* 15(1), p. 2, doi:10.1145/2559950.

[2] G. Bonfante, A. Cichon, J.-Y. Marion & H. Touzet (2001): *Algorithms with Polynomial Interpretation Termination Proof*. *Journal of Functional Programming* 11(1), pp. 33–53, doi:10.1017/S0956796800003877.

[3] G. Bonfante, J.-Y. Marion & J.-Y. Moyen (2001): *On Lexicographic Termination Ordering with Space Bound Certifications*. In: *Perspectives of System Informatics, Lecture Notes in Computer Science* 2244, pp. 482–493, doi:10.1007/3-540-45575-2_46.

[4] G. Bonfante, J.-Y. Marion & J.-Y. Moyen (2011): *Quasi-interpretations A Way to Control Resources*. *Theoretical Computer Science* 412(25), pp. 2776–2796, doi:10.1016/j.tcs.2011.02.007.

[5] W. Buchholz (1995): *Proof-theoretic Analysis of Termination Proofs*. *Annals of Pure and Applied Logic* 75(1–2), pp. 57–65, doi:10.1016/0168-0072(94)00056-9.

[6] S.R. Buss (1986): *Bounded Arithmetic*. Bibliopolis, Napoli.

[7] S.R. Buss (1998): *First-Order Proof Theory of Arithmetic*. In S.R. Buss, editor: *Handbook of Proof Theory*, North Holland, Amsterdam, pp. 79–147, doi:10.1016/S0049-237X(98)80017-7.

[8] N. Dershowitz (1982): *Orderings for Term-Rewriting Systems*. *Theoretical Computer Science* 17, pp. 279–301, doi:10.1016/0304-3975(82)90026-3.

[9] N. Eguchi (2010): *A Term-rewriting Characterization of PSPACE*. In T. Arai, C.T. Chong, R. Downey, J. Brendle, Q. Feng, H. Kikyo & H. Ono, editors: *Proceedings of the 10th Asian Logic Conference 2008*, World Scientific, pp. 93–112, doi:10.1142/9789814293020_0004.

[10] D. Hofbauer (1990): *Termination Proofs by Multiset Path Orderings Imply Primitive Recursive Derivation Lengths*. In: *Proceedings of the 2nd International Conference on Algebraic and Logic Programming, Lecture Notes in Computer Science* 463, pp. 347–358, doi:10.1007/3-540-53162-9_50.

[11] N.D. Jones (1997): *Computability and Complexity - from a Programming Perspective*. Foundations of Computing Series, MIT Press, doi:10.1007/978-94-010-0413-8_4.

[12] N.D. Jones & A. Mycroft (1986): *Data Flow Analysis of Applicative Programs Using Minimal Function Graphs*. In: *Proceedings of the 13th ACM Symposium on Principles of Programming Languages*, pp. 296–306, doi:10.1145/512644.512672.

[13] S. Kamin & J.-J. Lévy (1980): *Two Generalizations of the Recursive Path Ordering*. Unpublished manuscript, University of Illinois.

[14] D. Leivant & J.-Y. Marion (1995): *Ramified Recurrence and Computational Complexity II: Substitution and Poly-space*. *Lecture Notes in Computer Science* 933, pp. 486–500, doi:10.1007/BFb0022277.

[15] J.-Y. Marion (2003): *Analysing the Implicit Complexity of Programs*. *Information and Computation* 183(1), pp. 2–18, doi:10.1016/S0890-5401(03)00011-7.

[16] I. Oitavem (2001): *Implicit Characterizations of Pspace*. In: *Proof Theory in Computer Science, Lecture Notes in Computer Science* 2183, Springer, pp. 170–190, doi:10.1007/3-540-45504-3_11.

[17] I. Oitavem (2002): *A Term Rewriting Characterization of the Functions Computable in Polynomial Space*. *Archive for Mathematical Logic* 41(1), pp. 35–47, doi:10.1007/s001530200002.

[18] W.J. Savitch (1970): *Relationships Between Nondeterministic and Deterministic Tape Complexities*. *Journal of Computer and System Sciences* 4(2), pp. 177–192, doi:10.1016/S0022-0000(70)80006-X.

[19] K. Slonneger & B.L. Kurtz (1995): *Formal Syntax and Semantics of Programming Languages - A Laboratory Based Approach*. Addison-Wesley.

[20] Terese (2003): *Term Rewriting Systems*. *Cambridge Tracts in Theoretical Computer Science* 55, Cambridge University Press.

[21] D.B. Thompson (1972): *Subrecursiveness: Machine-Independent Notions of Computability in Restricted Time and Storage*. *Mathematical Systems Theory* 6(1), pp. 3–15, doi:10.1007/BF01706069.

[22] A. Weiermann (1995): *Termination Proofs for Term Rewriting Systems by Lexicographic Path Orderings Imply Multiply Recursive Derivation Lengths*. *Theoretical Computer Science* 139(1&2), pp. 355–362, doi:10.1016/0304-3975(94)00135-6.

[23] G. Winskel (1993): *The Formal Semantics of Programming Languages - An Introduction*. Foundations of Computing Series, MIT Press.