Complexity Bounds for Ordinal-Based Termination[*] (Invited Talk)

Sylvain Schmitz

LSV, ENS Cachan & CNRS & INRIA, France

Abstract. 'What more than its truth do we know if we have a proof of a theorem in a given formal system?' We examine Kreisel's question in the particular context of program termination proofs, with an eye to deriving complexity bounds on program running times.

Our main tool for this are *length function theorems*, which provide complexity bounds on the use of well quasi orders. We illustrate how to prove such theorems in the simple yet until now untreated case of ordinals. We show how to apply this new theorem to derive complexity bounds on programs when they are proven to terminate thanks to a ranking function into some ordinal.

1998 ACM Subject Classification. F.2.0 Analysis of Algorithms and Problem Complexity; F.3.1 Logics and Meanings of Programs

Keywords: Fast-growing complexity, length function theorem, Ramsey-based termination, ranking function, well quasi order.

1 Introduction

Whenever we prove the termination of a program, we might also expect to gain some information on its complexity. The jump from termination to complexity analysis is however often involved. The question has already been studied for many termination techniques, e.g. termination orderings [21, 38, 39, 9, 25], polynomial interpretations [7], dependency pairs [20], size-change abstractions [5, 13], abstract interpretation [19], or ranking functions [2] to cite a few.

The purpose of this paper is to present the complexity bounds one can similarly derive from termination proofs relying on *well quasi orders* (wqo). There are already some accessible introductions to the subject [33, 34], with applications to algorithms for so-called 'well-structured systems.' Our emphasis here is however on the particular case of *well orders*, i.e. of ranking functions into ordinal numbers. Although this is arguably the oldest and best-understood termination proof technique, which can be tracked back for instance to works by Turing [36] or Floyd [18], deriving complexity bounds for well orders has only been considered in restricted cases in the wqo literature [1]. As we shall see, by

[*] Work funded in part by the ANR grant 11-BS02-001-01 ReacHard.

revisiting ideas by Buchholz, Cichoń, and Weiermann [10, 8] and the framework of [32], the case of well orders turns out to be fairly simple, and provides an introduction to the definitions and techniques employed for more complex wqos.

Contents. After setting the stage in Sec. 2 by recalling the definitions of well quasi orders, ranking functions, and order types, we work out the details of the proof of a *length function theorem* for ordinals below ε_0 in Sec. 3. Such combinatorial statements provide bounds on the length of so-called *bad sequences* of elements taken from a wqo—i.e. of descending sequences in the case of a well-order—, and thus on the running time of programs proved to terminate using the same wqos.

More precisely, we first recall in Sec. 3.1 the main notions employed in the proofs of such theorems in [32, 33], and apply them to the ordinal case in Sec. 3.3. This yields a new length function theorem, this time for ordinals (Thm. 3.3). As far as we know, this is an original contribution, which relies on ideas developed by Cichoń and others in the 1990's [10, 8] on the use of ordinal norms for sub-structural hierarchies (recalled in Sec. 3.2). Unlike the length function theorems for other wqos found in the literature [28, 12, 38, 11, 17, 32, 33, 1], Thm. 3.3 does not just provide an upper bound on the maximal length of bad sequences, but offers instead an *exact* explicit formulation for such lengths using Cichoń's hierarchy of functions.

Those bounds are often more precise than actually needed, and we show in Sec. 4 how to classify them into suitable *fast-growing* complexity classes [31]. We also zoom in on the bounds for lexicographic ranking functions in Sec. 5, and relate them to the bounds obtained in [17] for the Ramsey-based termination technique of Podelski and Rybalchenko [30].

2 Well Quasi Orders and Termination

In terms of operational semantics, a termination proof establishes that the relation between successive program configurations is well founded. Rather than proving well foundedness from first principles, it is much easier to rely on existing well founded relations, whether we are attempting to prove termination with pen and paper or using an automatic tool. Well quasi orders and well orders are in this regard very well studied and well behaved classes of well founded relations.

2.1 Well Quasi Orders

A *quasi order* (qo) $\langle A, \leq \rangle$ consists of a support set A along with a transitive reflexive relation $\leq \subseteq A \times A$. We call a finite or infinite sequence $x_0, x_1, x_2, \ldots$ over A *good* if there exist two indices $i < j$ such that $x_i \leq x_j$, and *bad* otherwise.

Definition 2.1. *A* well quasi order *(wqo) is a qo $\langle A, \leq \rangle$ such that any infinite sequence $x_0, x_1, x_2, \ldots$ of elements over A is good. Equivalently, any bad sequence over A is finite.*

There are many equivalent definitions for wqos [see e.g. 33, Chap. 1]. Notably, $\langle A, \leq \rangle$ is a wqo if and only if

1. $\leq$ is *well-founded*, i.e. there does not exist any infinite decreasing sequence $x_0 > x_1 > x_2 > \cdots$ of elements in A, where $< \overset{\text{def}}{=} \leq \setminus \geq$, and
2. there are *no infinite antichains* over A, i.e. infinite sets of mutually incomparable elements for $\leq$.

Well (Partial) Orders. A wqo where $\leq$ is antisymmetric is called a *well partial order* (wpo). Note that quotienting a wqo by the equivalence $\equiv \overset{\text{def}}{=} \leq \cap \geq$, i.e. equating elements x and y whenever $x \leq y$ and $y \leq x$, yields a wpo.

A wpo $\langle A, \leq \rangle$ where $\leq$ is linear (aka total), is a *well order* (wo). Because a wo has antichains of cardinal at most 1, this coincides with the usual definition as a well-founded linear order. Finally, any *linearisation* of a wpo $\langle A, \leq \rangle$, i.e. any linear order $\preceq \supseteq \leq$ defines a wo $\langle A, \preceq \rangle$. One can think of the linearisation process as one of 'orienting' pairs of incomparable elements; such a linearisation always exists thanks to the order-extension principle.

Examples. For a basic example, consider any finite set Q along with the equality relation, which is a wqo $\langle Q, = \rangle$ (even a wpo) by the pigeonhole principle. As explained above, any wo is a wqo, which provides us with another basic example: the set of natural numbers along with its natural ordering $\langle \mathbb{N}, \leq \rangle$.

Many more examples can be constructed using algebraic operations: for instance, if $\langle A, \leq_A \rangle$ and $\langle B, \leq_B \rangle$ are wqos (resp. wpos), then so is their *Cartesian product* $\langle A \times B, \leq_\times \rangle$, where $(x, y) \leq_\times (x', y')$ if and only if $x \leq_A x'$ and $y \leq_B y'$ is the *product ordering*; in the case of $\langle \mathbb{N}^d, \leq_\times \rangle$ this result is also known as Dickson's Lemma. Some further popular examples of operations that preserve wqos include the set of finite sequences over A with subword embedding $\langle A^*, \leq_* \rangle$ (a result better known as Higman's Lemma), finite trees labelled by A with the homeomorphic embedding $\langle T(A), \leq_T \rangle$ (aka Kruskal's Tree Theorem), and finite graphs labelled by A with the minor ordering $\langle G(A), \leq_{\text{minor}} \rangle$ (aka Robertson and Seymour's Graph Minor Theorem).

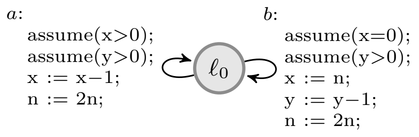
Turning to well orders, an operation that preserves wos is the *lexicographic product* $\langle A \times B, \leq_{\text{lex}} \rangle$ where $(x, y) \leq_{\text{lex}} (x', y')$ if and only if $x <_A x'$, or $x = x'$ and $y \leq_B y'$. This is typically employed in d-tuples of natural numbers ordered lexicographically $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$: observe that this is a linearisation of $\langle \mathbb{N}^d, \leq_\times \rangle$. Another classical well order employed in termination proofs is the *multiset* order $\langle \mathbb{M}(A), \leq_{\text{mset}} \rangle$ of Dershowitz and Manna [15]. There, $\mathbb{M}(A)$ denotes the set of finite multisets over the wo $\langle A, \leq \rangle$, i.e. of functions $m \colon A \to \mathbb{N}$ with finitely many x in A such that $m(x) > 0$, and $m \leq_{\text{mset}} m'$ if and only if for all x in A, if $m(x) > m'(x)$, then there exists $y >_A x$ such that $m(y) < m'(y)$ [see also 23].

2.2 Termination

We illustrate the main ideas in this paper using a very simple program, given in pseudo-code in Fig. 1a. Formally, we see the operational semantics of a program

```
ℓ0: while x >= 0 and y > 0 do
  if x > 0 then
    a: x := x−1; n := 2n;
  else
    b: x := n; y := y−1; n := 2n;
done
```

(a) A program over integer variables



(b) The associated control-flow graph

Fig. 1. A simple terminating program

as the one in Fig. 1a as a transition system $\mathcal{S} = \langle Conf, \rightarrow_{\mathcal{S}} \rangle$ where $Conf$ denotes the set of program configurations and $\rightarrow_{\mathcal{S}} \subseteq Conf \times Conf$ a transition relation. In such a simple non-recursive program, the set of configurations is a variable valuation, including a program counter `pc` ranging over the finite set of program locations. For our simple program a single location suffices and we set

$$Conf = \{\ell_0\} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} , \tag{1}$$

where the last three components provide the values of `x`, `y`, and `n`, and the first component the value of `pc`. The corresponding transition relation contains for instance

$$(\ell_0, 3, 1, 4) \rightarrow_{\mathcal{S}} (\ell_0, 2, 1, 8) \tag{2}$$

using transition a in Fig. 1b.

Proving Termination. We say that a transition system $\mathcal{S} = \langle Conf, \rightarrow_{\mathcal{S}} \rangle$ *terminates* if every execution $c_0 \rightarrow_{\mathcal{S}} c_1 \rightarrow_{\mathcal{S}} \cdots$ is finite. For instance, in order to prove the termination of the program of Fig. 1 by a wqo argument, consider some (possibly infinite) execution

$$(\ell_0, x_0, y_0, n_0) \rightarrow_{\mathcal{S}} (\ell_0, x_1, y_1, n_1) \rightarrow_{\mathcal{S}} (\ell_0, x_2, y_2, n_2) \rightarrow_{\mathcal{S}} \cdots \tag{3}$$

over $Conf$. Because a negative value for `x` or `y` would lead to immediate termination, the associated sequence of pairs

$$(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots \tag{4}$$

is actually over $\mathbb{N}^2$. Consider now two indices $i < j$:

– either b is never fired throughout the execution between steps i and j, and then $y_i = \cdots = y_j$ and $x_i > x_j$,
– or b is fired at least once, and $y_i > y_j$.

In both cases $(x_i, y_i) \not\leq_{\times} (x_j, y_j)$, i.e. the sequence (4) is bad for the product ordering. Since $\langle \mathbb{N}^2, \leq_{\times} \rangle$ is a wqo, this sequence is necessarily finite, and so is the original sequence (3): the program of Fig. 1 terminates on all inputs.

Quasi-Ranking Functions. The above termination argument for our example program easily generalises:

Definition 2.2. *Given a transition system $\mathcal{S} = \langle Conf, \to_{\mathcal{S}} \rangle$, a* quasi-ranking function *is a map $f\colon Conf \to A$ into a wqo $\langle A, \leq \rangle$ such that, whenever $c \to_{\mathcal{S}}^{+} c'$ is a non-empty sequence of transitions of $\mathcal{S}$, $f(c) \not\leq f(c')$.*

In our treatment of the program of Fig. 1 above, we picked $f(\ell_0, x, y, z) = (x, y)$ and $\langle A, \leq \rangle = \langle \mathbb{N}^2, \leq_{\times} \rangle$. The existence of a quasi-ranking function always yields termination:

Proposition 2.3. *Given a transition system $\mathcal{S} = \langle Conf, \to_{\mathcal{S}} \rangle$, if there exists a quasi-ranking function for $\mathcal{S}$, then $\mathcal{S}$ terminates.*

Proof. Let f be a quasi-ranking function of $\mathcal{S}$ into a wqo $\langle A, \leq \rangle$. Any sequence of configurations $c_0 \to_{\mathcal{S}} c_1 \to_{\mathcal{S}} \cdots$ of $\mathcal{S}$ is associated by f to a bad sequence $f(c_0), f(c_1), \ldots$ over A and is therefore finite. □

Note that the converse statement also holds; see Remark 2.4 below.

Ranking Functions. The most typical method in order to prove that a program terminates for all inputs is to exhibit a *ranking function* f into some well-order, such that $\to_{\mathcal{S}}$-related configurations have decreasing rank [36, 18]. Note that this is a particular instance of quasi-ranking functions: a ranking function can be seen as a quasi-ranking function into a wo $\langle A, \leq \rangle$. Indeed, if $c \to_{\mathcal{S}} c'$, then the condition $f(c) \not\leq f(c')$ of Def. 2.2 over a wo is equivalent to requiring $f(c) > f(c')$, and then implies by transitivity $f(c) > f(c')$ whenever $c \to_{\mathcal{S}}^{+} c'$.

The program of Fig. 1 can easily be given a ranking function: define for this $f(\ell_0, x, y, n) = (y, x)$ ranging over the wo $\langle \mathbb{N}^2, \leq_{\text{lex}} \rangle$. Cook, See, and Zuleger [14] and Ben-Amram and Genaim [4] consider for instance the automatic synthesis of such lexicographic linear ranking functions for integer loops like Fig. 1a. Such ranking functions into $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ are described there by d functions $f_1, f_2, \ldots, f_d\colon Conf \to \mathbb{N}$ such that, whenever $c \to_{\mathcal{S}} c'$, then $(f_1(c), f_2(c), \ldots, f_d(c)) >_{\text{lex}} (f_1(c'), f_2(c'), \ldots, f_d(c'))$; in our example $f_1(c) = y$ and $f_2(c) = x$. Linearity means that each function f_i is a linear affine function of the values of the program variables.

Remark 2.4. Observe that any *deterministic* terminating program can be associated to a (quasi-)ranking function into $\mathbb{N}$, which maps each configuration to the number of steps before termination. We leave it as an exercise to the reader to figure out such a ranking function for Fig. 1—the answer can be found in Sec. 3. There are at least two motivations for considering other wqos:

– Programs can be nondeterministic, for instance due to interactions with an environment. Then the supremum of the number of steps along all the possible paths can be used as the range for a ranking function; this is a countable well-order.

– Whether by automated means or by manual means, such monolithic ranking functions are often too hard to synthesise and to check once found or guessed—note that the canonical 'number of steps' function is not recursive in general. This motivates employing more complex well (quasi-)orders in exchange for simpler ranking functions.

2.3 Ordinals

Write $\langle [d], \leq \rangle$ for the initial segment of the naturals $[d] = \{0, \ldots, d-1\}$; this is a finite linear order for each d. We can then replace our previous lexicographic ranking function for Fig. 1 with a multiset ranking function into $\langle \mathbb{M}([2]), \leq_{\text{mset}} \rangle$: $f(\ell_0, x, y, m) = \{1^y, 0^x\}$ is a ranking function that associates a multiset containing y copies of the element '1' and x copies of '0' to the configuration (ℓ_0, x, y, n).

This might seem like a rather artificial example of a multiset ranking function, and indeed more generally $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ and $\langle \mathbb{M}([d]), \leq_{\text{mset}} \rangle$ are *order-isomorphic* for every dimension d: indeed, $r(n_1, \ldots, n_d) = \{(d-1)^{n_1}, \ldots, 0^{n_d}\}$ is a bijection satisfying $(n_1, \ldots, n_d) \leq_{\text{lex}} (n'_1, \ldots, n'_d)$ if and only if $r(n_1, \ldots, n_d) \leq_{\text{mset}} r(n'_1, \ldots, n'_d)$.

In order to pick a unique representative for each isomorphism class of (simple enough) well orders, we are going to employ their *order types*, presented as *ordinal terms* in Cantor normal form. For instance ω^d is the order type of both $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ and $\langle \mathbb{M}([d]), \leq_{\text{mset}} \rangle$.

Ordinals in ε_0 can be canonically represented as *ordinal terms* α in Cantor normal form

$$\alpha = \omega^{\alpha_1} + \cdots + \omega^{\alpha_p} \tag{CNF}$$

with *exponents* $\alpha > \alpha_1 \geq \cdots \geq \alpha_p$. We write as usual 1 for the term ω^0 and ω for the term ω^1. Grouping equal exponents yields the strict form

$$\alpha = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_p} \cdot c_p$$

with $\alpha > \alpha_1 > \cdots > \alpha_p$ and *coefficients* $0 < c_1, \ldots, c_p < \omega$. The ordinal ε_0, i.e. the least solution of $\omega^x = x$, is the supremum of the ordinals presentable in this manner.

Computing Order Types. The order types $o(A, \leq_A)$ of the well orders $\langle A, \leq_A \rangle$ we already mentioned in this paper are well-known: $o([d], \leq) = d$, $o(\mathbb{N}, \leq) = \omega$, $o(A \times B, \leq_{\text{lex}}) = o(A, \leq_A) \cdot o(B, \leq_B)$, and $o(\mathbb{M}(A), \leq_{\text{mset}}) = \omega^{o(A, \leq_A)}$. The ranking function for the program in Fig. 1 can now be written as $f(\ell_0, x, y, n) = \omega \cdot y + x$ and ranges over the set of ordinal terms below ω^2. Note that we will identify the latter set with ω^2 itself as in the usual set-theoretic definition of ordinals; thus $\beta < \alpha$ if and only if $\beta \in \alpha$.

By extension, we also write $o(x)$ for the ordinal term in $o(A)$ associated to an element x in A; for instance in $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$, $o(n_1, \ldots, n_d) = \omega^{d-1} \cdot n_1 + \cdots + n_d$.

3 Complexity Bounds

We aim to provide complexity upper bounds for programs proven to terminate thanks to some (quasi-)ranking function. There are several results of this kind in the literature [28, 12, 38, 11, 17, 32, 1], which are well-suited for algorithms manipulating complex data structures—for which we can employ the rich wqo toolkit.

A major drawback of all these complexity bounds is that they are *very* high—i.e., non-elementary except in trivial cases—, whereas practitioners are mostly interested in polynomial bounds. Such high complexities are however unavoidable, because the class of programs terminating thanks to a quasi-ranking function encompasses programs with matching complexities. For instance, even integer loops can be deceivingly simple: recall that the program of Fig. 1 terminated using a straightforward ranking function into ω^2. Although this is just one notch above a ranking function into ω, we can already witness fairly complex computations. Observe indeed that the following are some execution steps of our program:

$$\begin{aligned}(\ell_0, x, y, 1) &\xrightarrow{a^x b}_{\mathcal{S}} (\ell_0, 2^x, y-1, 2^{x+1})\\ &\xrightarrow{a^{2^x} b}_{\mathcal{S}} (\ell_0, 2^{2^x+x+1}, y-2, 2^{2^x+x+2})\\ &\xrightarrow{a^{2^{2^x+x+1}} b}_{\mathcal{S}} (\ell_0, 2^{2^{2^x+x+1}+2^x+x+2}, y-3, 2^{2^{2^x+x+1}+2^x+x+3})\;.\end{aligned}$$

Continuing this execution, we see that our simple program exhibits executions of length greater than a tower of exponentials in y, i.e. it is non elementary.

3.1 Controlled Ranking Functions

By Def. 2.1, bad sequences in a wqo are always finite—which in turn yields the termination of programs with quasi-ranking functions—, but no statement is made regarding *how long* they can be. This is for a very good reason: they can be arbitrarily long.

For instance, over the wo $\langle \mathbb{N}, \leq \rangle$,

$$n, n-1, \ldots, 1, 0 \tag{5}$$

is a bad sequence of length $n+1$ for every n. Arguably, this is not so much of an issue, since what we are really interested in is the length *as a function* of the initial configuration—which includes the inputs to the program. Thus (5) is the maximal bad sequence over $\langle \mathbb{N}, \leq \rangle$ with initial element of 'size n.'

However, as soon as we move to more complex wqos, we can exhibit arbitrary bad sequence lengths even with fixed initial configurations. For instance, over $\langle \mathbb{N}^2, \leq_{\text{lex}} \rangle$,

$$(1,0), (0,n), (0,n-1), \ldots, (0,1), (0,0) \tag{6}$$

is a bad sequence of length $n+2$ for every n starting from the fixed $(1,0)$. Nonetheless, the behaviour of a program exhibiting such a sequence of ranks is rather unusual: such a sudden 'jump' from $(1, 0)$ to an arbitrary $(0, n)$ is not possible in a deterministic program once the user inputs have been provided.

Controlled Sequences. In the following, we will assume that no such arbitrary jump can occur. This comes at the price of some loss of generality in the context of termination analysis, where nondeterministic assignments of arbitrary values are typically employed to model values provided by the environment—for instance interactive user inputs or concurrently running programs—, or because of abstracted operations. Thankfully, in most cases it is easy to *control* how large the program variables can grow during the course of an execution.

Formally, given a wqo $\langle A, \leq_A\rangle$, we posit a *norm* function $|.|_A\colon A \to \mathbb{N}$ on the elements of A. In order to be able to derive combinatorial statements, we require

$$A_{\leq n} \overset{\text{def}}{=} \{x \in A \mid |x|_A \leq n\} \tag{7}$$

to be finite for every n. We will use the following norms on the wqos defined earlier: in a finite Q, all the elements have the same norm 0; in $\mathbb{N}$ or $[d]$, n has norm $|n|_{\mathbb{N}} = n$; for Cartesian or lexicographic products with support $A \times B$, (x, y) has the infinite norm $\max(|x|_A, |y|_B)$; finally, for multisets $\mathbb{M}(A)$, m has norm $\max_{x\in A, m(x)>0}(m(x), |x|_A)$.

Let $g\colon \mathbb{N} \to \mathbb{N}$ be a monotone and expansive function: for all x, x', $x \leq x'$ implies $g(x) \leq g(x')$ and $x \leq g(x)$. We say that a sequence $x_0, x_1, x_2, \ldots$ of elements in A is (g, n_0)*-controlled* for some n_0 in $\mathbb{N}$ if

$$|x_i|_A \leq g^i(n_0) \tag{8}$$

for all i, where g^i denotes the ith iterate of g. In particular $|x_0|_A \leq g^0(n_0) = n_0$, which prompts the name of *initial norm* for n_0, and amortised steps cannot grow faster than g the *control function.*

By extension, a quasi-ranking function $f\colon \mathit{Conf} \to A$ for a transition system $\mathcal{S} = \langle \mathit{Conf}, \to_{\mathcal{S}}\rangle$ and a normed wqo $\langle A, \leq_A, |.|_A\rangle$ is g*-controlled* if, whenever $c \to_{\mathcal{S}} c'$ is a transition in $\mathcal{S}$,

$$|f(c')|_A \leq g(|f(c)|_A)\ . \tag{9}$$

This ensures that any sequence $f(c_0), f(c_1), \ldots$ of ranks associated to an execution $c_0 \to_{\mathcal{S}} c_1 \to_{\mathcal{S}} \cdots$ of $\mathcal{S}$ is $(g, |f(c_0)|_A)$-controlled. For instance, our ranking function $f(\ell_0, x, y, n) = (y, x)$ for the program of Fig. 1 into $\langle \mathbb{N}^2, \leq_{\text{lex}}\rangle$ is g-controlled for $g(x) = 2x$.

Length Functions. The motivation for controlled sequences is that their length can be bounded. Consider for this the tree one obtains by sharing common prefixes of all the (g, n_0)-controlled bad sequences over a normed wqo $(A, \leq_A, |.|_A)$. This tree has

– finite branching by (7) and (8), more precisely branching degree bounded by the cardinal of $A_{\leq g^i(n_0)}$ for a node at depth i, and
– no infinite branches thanks to the wqo property.

By Kőnig's Lemma, this tree of bad sequences is therefore finite, of some height $L_{g,n_0,A}$ representing the length of the maximal (g, n_0)-controlled bad sequence(s) over A. In the following, since we are mostly interested in this length as a function of the initial norm n_0, we will see this as a *length function* $L_{g,A}(n_0)$.

Length Function Theorems. Observe that $L_{g,A}$ also bounds the asymptotic execution length in a program endowed with a g-controlled quasi-ranking function into $\langle A, \leq_A, |.|_A\rangle$. Our purpose will thus be to obtain explicit complexity bounds on $L_{g,A}$ depending on g and A. We call such combinatorial statements *length function theorems*; see [28, 12, 38, 11, 17, 32, 1] for some examples.

For applications to termination analysis, we are especially interested in the case of well orders. Somewhat oddly, this particular case has seldom been considered; to our knowledge the only instance is due to Abriola, Figueira, and Senno [1] who derive upper bounds for multisets of tuples of naturals ordered lexicographically, i.e. for $L_{g,\mathbb{M}(\mathbb{N}^d)}$ (beware that their notion of control is defined slightly differently).

3.2 Hardy and Cichoń Hierarchies

As we saw with the example of Fig. 1, even simple terminating programs can have a very high complexity. In order to express such high bounds, a convenient tool is found in *subrecursive hierarchies*, which employ recursion over ordinal indices to define faster and faster growing functions. We define in this section two such hierarchies.

Fundamental Sequences and Predecessors. Let us first introduce some additional notions on ordinal terms. Consider an ordinal term α in Cantor normal form $\omega^{\alpha_1} + \cdots + \omega^{\alpha_p}$. In this representation, $\alpha = 0$ if and only if $p = 0$. An ordinal α of the form $\alpha' + 1$ (i.e. with $p > 0$ and $\alpha_p = 0$) is called a *successor* ordinal, and otherwise if $\alpha > 0$ it is called a *limit* ordinal, and can be written as $\gamma + \omega^\beta$ by setting $\gamma = \omega^{\alpha_1} + \cdots + \omega^{\alpha_{p-1}}$ and $\beta = \alpha_p$. We usually write 'λ' to denote a limit ordinal.

A *fundamental sequence* for a limit ordinal λ is a sequence $(\lambda(x))_{x<\omega}$ of ordinal terms with supremum λ. We use the standard assignment of fundamental sequences to limit ordinals defined inductively by

$$(\gamma + \omega^{\beta+1})(x) \overset{\text{def}}{=} \gamma + \omega^\beta \cdot (x + 1) , \qquad (\gamma + \omega^\lambda)(x) \overset{\text{def}}{=} \gamma + \omega^{\lambda(x)} . \tag{10}$$

This particular assignment satisfies e.g. $0 < \lambda(x) < \lambda(y)$ for all $x < y$. For instance, $\omega(x) = x + 1$, $(\omega^{\omega^4} + \omega^{\omega^3+\omega^2})(x) = \omega^{\omega^4} + \omega^{\omega^3+\omega\cdot(x+1)}$.

The *predecessor* $P_x(\alpha)$ of an ordinal term $\alpha > 0$ at a value x in $\mathbb{N}$ is defined inductively by

$$P_x(\alpha + 1) \overset{\text{def}}{=} \alpha , \qquad P_x(\lambda) \overset{\text{def}}{=} P_x(\lambda(x)) . \tag{11}$$

In essence, the predecessor of an ordinal is obtained by repeatedly taking the xth element in the fundamental sequence of limit ordinals, until we finally reach a successor ordinal and remove 1. For instance, $P_x(\omega^2) = P_x(\omega \cdot (x + 1)) = P_x(\omega \cdot x + x + 1) = \omega \cdot x + x$.

Subrecursive Hierarchies. In the context of controlled sequences, the hierarchies of Hardy and Cichoń turn out to be especially well-suited [11]. Let $h\colon \mathbb{N} \to \mathbb{N}$ be a function. The *Hardy hierarchy* $(h^\alpha)_{\alpha\in\varepsilon_0}$ is defined for all $0 < \alpha < \varepsilon_0$ by[1]

$$h^0(x) \overset{\text{def}}{=} x\;, \qquad\qquad h^\alpha(x) \overset{\text{def}}{=} h^{P_x(\alpha)}(h(x))\;, \tag{12}$$

and the *Cichoń hierarchy* $(h_\alpha)_{\alpha\in\varepsilon_0}$ is similarly defined for all $0 < \alpha < \varepsilon_0$ by

$$h_0(x) \overset{\text{def}}{=} 0\;, \qquad\qquad h_\alpha(x) \overset{\text{def}}{=} 1 + h_{P_x(\alpha)}(h(x))\;. \tag{13}$$

Observe that h^k for some finite k is the kth iterate of h. This intuition carries over: h^α is a transfinite iteration of the function h, using diagonalisation in the fundamental sequences to handle limit ordinals.

For instance, starting with the successor function $H(x) \overset{\text{def}}{=} x+1$, we see that a first diagonalisation yields $H^\omega(x) = H^x(x+1) = 2x+1$. The next diagonalisation occurs at $H^{\omega\cdot 2}(x) = H^{\omega+x}(x+1) = H^\omega(2x+1) = 4x+3$. Fast-forwarding a bit, we get for instance a function of exponential growth $H^{\omega^2}(x) = 2^{x+1}(x+1) - 1$, and later a non-elementary function H^{ω^3}, an 'Ackermannian' non primitive-recursive function H^{ω^ω}, and a 'hyper-Ackermannian' non multiply recursive-function $H^{\omega^{\omega^\omega}}$. Regarding the Cichoń functions, an easy induction on α shows that $H^\alpha(x) = H_\alpha(x) + x$.

On the one hand, Hardy functions are well-suited for expressing large iterates of a control function, and therefore for bounding the norms of elements in a controlled sequence. For instance, the program in Fig. 1 computes $g^{\omega\cdot y+x}(n)$ for the function $g(x) = 2x$ when run on non-negative inputs x, y, n. On the other hand, Cichoń functions are well-suited for expressing the length of controlled sequences. For instance, $g_{\omega\cdot y+x}(n)$ is the length of the execution of the program. This relation is a general one: we can compute how many times we should iterate h in order to compute $h^\alpha(x)$ using the corresponding Cichoń function:

$$h^\alpha(x) = h^{h_\alpha(x)}(x)\;. \tag{14}$$

Monotonicity Properties. Assume h is monotone and expansive. Then both h^α and h_α are monotone and expansive [see 11, 33, 35]. However, those hierarchies are not monotone in the ordinal indices: for instance, $H^\omega(x) = 2x+1 < 2x+2 = H^{x+2}(x)$ although $\omega > x+2$.

Some refinement of the ordinal ordering is needed in order to obtain monotonicity of the hierarchies. Define for this the *pointwise ordering* $\prec_x$ at some x in $\mathbb{N}$ as the smallest transitive relation such that

$$\alpha \prec_x \alpha + 1\;, \qquad\qquad \lambda(x) \prec_x \lambda\;. \tag{15}$$

The relation '$\beta \prec_x \alpha$' is also noted '$\beta \in \alpha[x]$' in [35, pp. 158–163]. The $\prec_x$ relations form a strict hierarchy of refinements of the ordinal ordering $<$:

$$\prec_0 \subsetneq \prec_1 \subsetneq \cdots \subsetneq \prec_x \subsetneq \cdots \subsetneq <\;. \tag{16}$$

[1] Note that this is equivalent to defining $h^{\alpha+1}(x) \overset{\text{def}}{=} h^\alpha(h(x))$ and $h^\lambda(x) \overset{\text{def}}{=} h^{\lambda(x)}(x)$.

As desired, our hierarchies are monotone for the pointwise ordering [11, 33, 35]:

$$\beta \prec_x \alpha \qquad\qquad \text{implies} \qquad\qquad h_\beta(x) \leq h_\alpha(x) \; . \tag{17}$$

Ordinal Norms. As a first application of the pointwise ordering, define the *norm* of an ordinal as the maximal coefficient that appears in its associated CNF: if $\alpha = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_p} \cdot c_p$ with $\alpha_1 > \cdots > \alpha_p$ and $c_1, \ldots, c_p > 0$, then

$$N\alpha \overset{\text{def}}{=} \max\{c_1, \ldots, c_p, N\alpha_1, \ldots, N\alpha_p\} \; . \tag{18}$$

Observe that this definition essentially matches the previously defined norms over multisets and tuples of vectors: e.g. in $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$, the ordinal norm satisfies $No(n_1, \ldots, n_d) = \max(d, |(n_1, \ldots, n_d)|_{\mathbb{N}^d})$, and in $\langle \mathbb{M}(\mathbb{N}^d), \leq_{\text{mset}} \rangle$, $No(m) = \max(d, |m|_{\mathbb{M}(\mathbb{N}^d)})$. The relation between ordinal norms and the pointwise ordering is that [33, 35, p. 158]

$$\beta < \alpha \qquad\qquad \text{implies} \qquad\qquad \beta \prec_{N\beta} \alpha \; . \tag{19}$$

Together with (16) and (17), this entails that for all $x \geq N\beta$, $h_\beta(x) \leq h_\alpha(x)$.

3.3 A Length Function Theorem for ε_0

We are now equipped to prove a length function theorem for all ordinals α below ε_0, i.e. an explicit expression for $L_{g,\alpha}$ for the wo $\langle \alpha, \leq, N \rangle$. This proof relies on two main ingredients: a *descent equation* established in [32] for all normed wqos, and an alternative characterisation of the Cichoń hierarchy in terms of maximisations inspired by [10, 8].

Residuals and a Descent Equation. Let $\langle A, \leq, |.| \rangle$ be a normed wqo and x be an element of A. We write

$$A/x \overset{\text{def}}{=} \{y \in A \mid x \not\leq y\} \tag{20}$$

for the *residual* of A in x. Observe that by the wqo property, there cannot be infinite sequences of residuations $A/x_0/x_1/x_2/\cdots$ since $x_i \not\leq x_j$ for all $i < j$.

Consider now a (g, n_0)-controlled bad sequence $x_0, x_1, x_2, \ldots$ over $\langle A, \leq, |.|_A \rangle$. Assuming the sequence is not empty, then because this is a bad sequence we see that for all $i > 0$, $x_0 \not\leq x_i$, i.e. that the suffix $x_1, x_2, \ldots$ is actually a bad sequence over A/x_0. This suffix is now $(g(n), n_0)$-controlled, and thus of length bounded by $L_{g,A/x_0}(g(n_0))$. This yields the following *descent equation* when considering all the possible (g, n)-controlled bad sequences:

$$L_{g,A}(n) = \max_{x \in A_{\leq n}} 1 + L_{g,A/x}(g(n)) \; . \tag{21}$$

In the case of a wo $\langle \alpha, \leq, N \rangle$, residuals can be expressed more simply for $\beta \in \alpha$ as

$$\alpha/\beta = \{\gamma \in \alpha \mid \beta > \gamma\} = \beta \; . \tag{22}$$

Thus the descent equation simplifies into

$$L_{g,\alpha}(n) = \max_{\beta < \alpha, N\beta \leq n} 1 + L_{g,\beta}(g(n)) \; . \tag{23}$$

Norm Maximisation. The reader might have noticed a slight resemblance between the ordinal descent equation (23) and the definition of the Cichoń hierarchy (13). It turns out that they are essentially the same functions: indeed, we are going to show in Prop. 3.2 that if $N\alpha \leq x$, then choosing $\beta = P_x(\alpha)$ maximises $h_\beta(h(x))$ among those $\beta < \alpha$ with $N\beta \leq x$; we follow in this [10, 8]. This is a somewhat technical proof, so the reader might want to skip the details and jump directly to Thm. 3.3.

Lemma 3.1. *Let $\alpha < \varepsilon_0$ and $x \geq N\alpha$. Then $P_x(\alpha) = \max_{\beta<\alpha, N\beta\leq x} \beta$.*

Proof. We prove the lemma through a sequence of claims.

Claim 3.1.1. $P_x(\alpha) < \alpha$.

We show for this first claim that, by transfinite induction over $\alpha > 0$, for all x

$$P_x(\alpha) \prec_x \alpha \tag{24}$$

Indeed, $P_x(\alpha+1) = \alpha \prec_x \alpha+1$ for the successor case, and $P_x(\lambda) = P_x(\lambda(x)) \prec_x \lambda(x) \prec_x$ by induction hypothesis on $\lambda(x) < \lambda$ for the limit case. Then (16) allows to conclude.

Let us introduce a variant of the ordinal norm. Let $\alpha = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_p} \cdot c_p$ be an ordinal in CNF with $\alpha > \alpha_1 > \cdots > \alpha_p$ and $\omega > c_1, \ldots, c_p > 0$. We say that α is *almost x-lean* if either (i) $c_p = x+1$ and both $N\sum_{i<p} \omega^{\alpha_i} \leq x$ and $N\alpha_p \leq x$, or (ii) $c_p \leq x$, $N\sum_{i<m} \omega^{\alpha_i} \leq x$, and α_p is almost x-lean. Note that an almost x-lean ordinal α has *not* norm x; it has however norm $x+1$. Here are several properties of note for almost x-lean ordinals:

Claim 3.1.2. If $N\lambda \leq x$, then $\lambda(x)$ is almost x-lean.

We prove this claim by induction on λ, letting $\lambda = \omega^{\lambda_1} \cdot c_1 + \cdots + \omega^{\lambda_p} \cdot c_p$ as above, where necessarily $N\lambda_p \leq x$. If λ_p is a successor ordinal $\beta+1$ (and thus $N\beta \leq x$), $\lambda(x) = \omega^{\lambda_1} \cdot c_1 + \cdots + \omega^{\lambda_p} \cdot (c_p - 1) + \omega^\beta \cdot (x+1)$ is almost x-lean by case (i). If λ_p is a limit ordinal, $\lambda(x) = \omega^{\lambda_1} \cdot c_1 + \cdots + \omega^{\lambda_p} \cdot (c_p - 1) + \omega^{\lambda_p(x)}$ is x-lean by case (ii) and the induction hypothesis on $\lambda_p < \lambda$.

Claim 3.1.3. If $\alpha + 1$ is almost x-lean, then $N\alpha \leq x$.

Let $\alpha+1 = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_p} \cdot c_p$ with $\alpha_p = 0$. We must be in case (i) since $\alpha_p = 0$ cannot be x-lean, thus $c_p = x+1$ and $N\alpha = N\omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_p} \cdot (c_p - 1) \leq x$.

Claim 3.1.4. If λ is almost x-lean, then $\lambda(x)$ is almost x-lean.

We prove the claim by induction on λ, letting $\lambda = \omega^{\lambda_1} \cdot c_1 + \cdots + \omega^{\lambda_p} \cdot c_p$:

If λ_p is a successor ordinal $\beta + 1$, $\lambda(x) = \omega^{\lambda_1} \cdot c_1 + \cdots + \omega^{\lambda_p} \cdot (c_p - 1) + \omega^\beta \cdot (x+1)$, and either (i) $c_p = x+1$ and $N\lambda_p \leq x$, and then $\lambda(x)$ also verifies (i), or (ii) $c_p \leq x$ and $\beta + 1$ is almost x-lean and thus $N\beta \leq x$ by Claim 3.1.3, and $\lambda(x)$ is again almost x-lean verifying condition (i).

If λ_p is a limit ordinal, then $\lambda(x) = \omega^{\lambda_1} \cdot c_1 + \cdots + \omega^{\lambda_p} \cdot (c_p - 1) + \omega^{\lambda_p(x)}$. Either (i) $c_p = x + 1$ and $N\lambda_p \leq x$, and by Claim 3.1.2 $\lambda_p(x)$ is almost x-lean and thus $\lambda(x)$ is almost x-lean by condition (ii), or (ii) $c_p \leq x$ and λ_p is almost x-lean, and by induction hypothesis $\lambda_p(x)$ is almost x-lean, and therefore $\lambda(x)$ is again almost x-lean by condition (ii).

Claim 3.1.5. If α is almost x-lean, then $NP_x(\alpha) \leq x$.

By induction over $\alpha > 0$: we see for the successor case that $NP_x(\alpha{+}1) = N\alpha \leq x$ by Claim 3.1.3, and for the limit case that $\lambda(x)$ is almost x-lean by Claim 3.1.4 and thus $P_x(\lambda(x)) \leq x$ by induction hypothesis.

Claim 3.1.6. If $N\alpha \leq x$, then $NP_x(\alpha) \leq x$.

Indeed, either α is a successor and this is immediate, or it is a limit λ and then $\lambda(x)$ is almost x-lean by Claim 3.1.2 and therefore $NP_x(\lambda) = NP_x(\lambda(x)) \leq x$ by Claim 3.1.5.

Claim 3.1.7. If $\beta < \alpha$ and $N\beta \leq x$, then $\beta \preceq_x P_x(\alpha)$.

Because the hypotheses entail $\beta \prec_x \alpha$ by (19), we can consider a sequence of atomic steps according to (15) for the pointwise ordering: $\beta = \beta_n \prec_x \cdots \prec_x \beta_1 \prec_x \alpha$. If α is a successor, then $\beta \preceq_x \beta_1 = P_x(\alpha)$. Otherwise β_1 is almost x-lean by Claim 3.1.2. Because $N\beta \leq x$, β is not almost x-lean, and by Claim 3.1.3 and Claim 3.1.4 there must be a greatest index $1 \leq i < n$ such that all the β_j's for $1 \leq j < i$ are almost x-lean limit ordinals and β_i is a successor almost x-lean ordinal. Thus $\beta \preceq_x \beta_{i+1} = P_x(\alpha)$.

To conclude the proof, $P_x(\alpha) < \alpha$ by Claim 3.1.1, $NP_x(\alpha) \leq x$ by Claim 3.1.6, and if $\beta < \alpha$ is such that $N\beta \leq x$, then $\beta \leq P_x(\alpha)$ by Claim 3.1.7 and (16), which together prove the lemma. ☐

Proposition 3.2. *Let $\alpha < \varepsilon_0$ and $x \geq N\alpha$. Then $h_\alpha(x) = \max_{\beta<\alpha, N\beta\leq x} 1 + h_\beta(h(x))$.*

Proof. If $\alpha = 0$ then there are no $\beta < \alpha$ and $\max_{\beta<\alpha,N\beta\leq x} 1 + h_\beta(h(x)) = 0 = h_0(x)$.

Otherwise by Lem. 3.1, since $P_x(\alpha) < \alpha$ and $NP_x(\alpha) \leq x$, $h_\alpha(x) = 1 + h_{P_x(\alpha)}(h(x)) \leq \max_{\beta<\alpha,N\beta\leq x} 1 + h_\beta(h(x))$. Conversely, let $\beta < \alpha$ with $N\beta \leq x$ be such that $\max_{\beta<\alpha,N\beta\leq x} 1+h_\beta(h(x)) = 1+h_\beta(h(x))$. By Lem. 3.1, $\beta \leq P_x(\alpha)$ and therefore by (19) $\beta \preceq_x P_x(\alpha)$. Since h is expansive, by (16), $\beta \preceq_{h(x)} P_x(\alpha)$. Therefore by (17), $1 + h_\beta(h(x)) \leq 1 + h_{P_x(\alpha)}(h(x)) = h_\alpha(x)$. ☐

Theorem 3.3 (Length Function Theorem for Ordinals). *Let $\alpha < \varepsilon_0$ and $x \geq N\alpha$. Then $L_{g,\alpha}(x) = g_\alpha(x)$.*

Proof. We use the ordinal descent equation (23) and Prop. 3.2. ☐

As an immediate corollary, we can bound the asymptotic complexity of programs proven to terminate through a g-controlled ranking function:

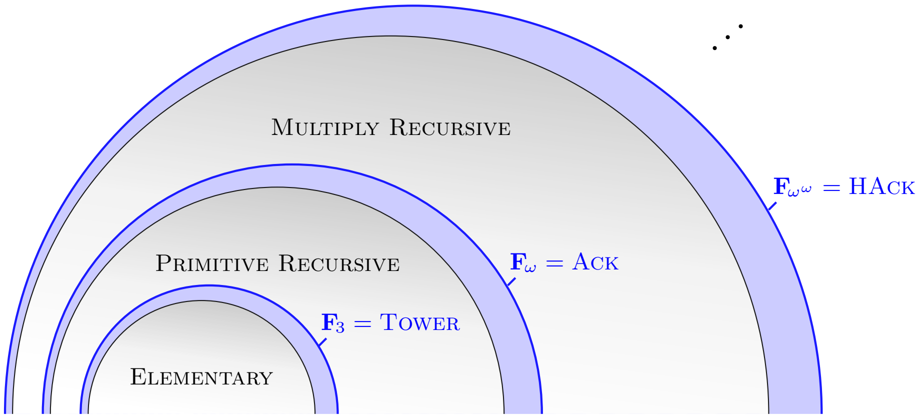


Fig. 2. Some complexity classes beyond Elementary

Corollary 3.4. *Given a transition system $\mathcal{S} = \langle Conf, \to_\mathcal{S}\rangle$, if there exists a g-controlled ranking function into $\alpha < \varepsilon_0$, then $\mathcal{S}$ runs in time $O(g_\alpha(n))$.*

As an illustration, a program proven to terminate thanks to a g-controlled ranking function into $\langle \mathbb{N}^d, \leq_{\mathrm{lex}}, |.|_{\mathbb{N}^d}\rangle$ has therefore an $O(g_{\omega^d}(n))$ bound on its worst-case asymptotic complexity. In the case of the program of Fig. 1, this yields an upper bound of $g_{\omega^2}(m) = 1 + g_{\omega\cdot m+m}(m)$ on its complexity for $g(x) \stackrel{\text{def}}{=} 2x$ and $m \stackrel{\text{def}}{=} \max(x, y, n)$. This matches its actual complexity.

4 Complexity Classification

As already mentioned, the complexity bounds provided by Thm. 3.3 are so high that they are only of interest for algorithms of very high complexity. Rather than obtaining precise complexity statements as in Thm. 3.3, the purpose is then to classify the complexity in rather broad terms: e.g., is the algorithm elementary? primitive-recursive? multiply-recursive?

4.1 Fast-Growing Classes

In order to tackle the complexities derived from Thm. 3.3, we need to employ complexity classes for very high complexity problems. For $\alpha > 2$, we define respectively the *fast-growing function* classes $(\mathscr{F}_\alpha)_\alpha$ of Löb and Wainer [27] and the *fast-growing complexity* classes $(\mathbf{F}_\alpha)_\alpha$ of [31] by

$$\mathscr{F}_{<\alpha} \stackrel{\text{def}}{=} \bigcup_{\beta<\omega^\alpha} \mathrm{FDT{\scriptsize IME}}\big(H^\beta(n)\big) , \qquad \mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{p\in\mathscr{F}_{<\alpha}} \mathrm{DT{\scriptsize IME}}\big(H^{\omega^\alpha}(p(n))\big) . \tag{25}$$

Recall that H^α denotes the αth function in the Hardy hierarchy with generative function $H(x) \stackrel{\text{def}}{=} x + 1$, and that $\mathrm{FDT{\scriptsize IME}}(t(n))$ (resp. $\mathrm{DT{\scriptsize IME}}(t(n))$) denotes the set of functions computable (resp. problems decidable) in deterministic time $O(t(n))$.

Some important complexity milestones can be characterised through these classes. Regarding the function classes, $\mathscr{F}_{<3}$ is the class of elementary functions, $\mathscr{F}_{<\omega}$ the class of primitive-recursive functions, $\mathscr{F}_{<\omega^\omega}$ the class of multiply-recursive functions, and $\mathscr{F}_{<\varepsilon_0}$ the class of ordinal-recursive functions. Turning to the complexity classes, $\mathbf{F}_3 = \textsc{Tower}$ is the class of problems with complexity bounded by a tower of exponentials of height bounded by an elementary function of the input, $\mathbf{F}_\omega = \textsc{Ack}$ the class of problems with complexity bounded by the Ackermann function of some primitive-recursive function of the input, and $\mathbf{F}_\omega^\omega = \textsc{HAck}$ of problems with complexity bounded by the hyper-Ackermann function $H^{\omega^{\omega^\omega}}$ composed with some multiply-recursive function. In other words, $\mathbf{F}_3$ (resp. $\mathbf{F}_\omega$ and $\mathbf{F}_{\omega^\omega}$) is the smallest complexity class $\mathbf{F}_\alpha$ which contains non elementary problems (resp. non primitive recursive and non multiply recursive problems); see Fig. 2.

4.2 Classification

The explicit formulation for the length function provided by Thm. 3.3 yields upper bounds in the $(\mathbf{F}_\alpha)_\alpha$ complexity classes. Assume that g belongs to the function class $\mathscr{F}_{<\gamma}$ for some γ. Then, by [31, Thm. 4.2], an algorithm with a g_{ω^α} complexity yields an upper bound in $\mathbf{F}_{\gamma+\alpha}$. In particular, a decision procedure terminating thanks to a lexicographic ranking function into $\langle \mathbb{N}^d, \leq_{\text{lex}}, |.|_{\mathbb{N}^d}\rangle$ with a linear control yields an $\mathbf{F}_{d+1}$ complexity upper bound. At greater complexities, if g is primitive recursive—i.e. is in $\mathscr{F}_{<\omega}$—and $\alpha \geq \omega$, then we obtain an upper bound in $\mathbf{F}_\alpha$ [31, Cor. 4.3].

5 Product vs. Lexicographic Orderings

Although we focus in this paper on ranking functions, automated termination provers employ many different techniques. While lexicographic ranking functions are fairly common [e.g. 14, 4, 37, for recent references], *disjunctive termination arguments* (aka Ramsey-based termination proofs) [30] are also a popular alternative.

5.1 Disjunctive Termination Arguments

In order to prove a program transition relation $\rightarrow_{\mathcal{S}}$ to be well-founded, Podelski and Rybalchenko [30] show that it suffices to exhibit a finite set of well-founded relations $T_1, \ldots, T_d \subseteq \mathit{Conf} \times \mathit{Conf}$ and prove that the transitive closure $\rightarrow_{\mathcal{S}}^+$ is included in the union $T_1 \cup \cdots \cup T_d$. In practice, we can assume each of the T_j for $1 \leq j \leq d$ to be proved well-founded through a quasi-ranking function f_j into a wqo $\langle A_j, \leq_j\rangle$. In the case of the program in Fig. 1, choosing

$$T_1 = \{((\ell_0, x, y, n), (\ell_0, x', y', n')) \mid x > 0 \wedge x' < x\} \tag{26}$$

$$T_2 = \{((\ell_0, x, y, n), (\ell_0, x', y', n')) \mid y > 0 \wedge y' < y\} \tag{27}$$

yields such a disjunctive termination argument, with $A_1 = A_2 = \mathbb{N}$.

Another way of understanding disjunctive termination arguments is that they define a quasi-ranking function f into the product wqo $\langle A_1 \times \cdots \times A_d, \leq_\times\rangle$, which maps a configuration c to the tuple $\langle f_1(c), \ldots, f_d(c)\rangle$, c.f. [17, Sec. 7.1].

5.2 A Comparison

Let us consider disjunctive termination arguments where each of the d relations T_j has a ranking function into $\mathbb{N}$, i.e. defining a quasi-ranking function into $\langle \mathbb{N}^d, \leq_\times\rangle$. A natural question at this point is how does it compare with a ranking function into $\langle \mathbb{N}^d, \leq_{\text{lex}}\rangle$, which seems fairly similar? Which programs can be shown to terminate with either method?

We might attempt to differentiate them through their *maximal order types* [22, 6]. In general, this is the supremum of the order types of all the linearisations of a wqo:

$$o(A, \leq) \overset{\text{def}}{=} \sup\{o(A, \preceq) \mid \preceq \text{ is a linearisation of } \leq\} \;. \tag{28}$$

However, in the case of $\langle \mathbb{N}^d, \leq_\times\rangle$, this maximal order type is ω^d, matching the order type of $\langle \mathbb{N}^d, \leq_{\text{lex}}\rangle$.

We can consider instead the maximal length of their controlled bad sequences. Those are different: the following example taken from [17, Remark 6.2] is a $(g, 1)$-controlled bad sequence over $\langle \mathbb{N}^2, \leq_\times\rangle$, which is good for $\langle \mathbb{N}^2, \leq_{\text{lex}}\rangle$, where $g(x) \overset{\text{def}}{=} x + 2$:

$$(1,1), (3,0), (2,0), (1,0), (0,9), (0,8), \ldots, (0,1), (0,0) \tag{29}$$

This sequence has length 14 whereas the maximal $(g, 1)$-controlled bad sequence for $\langle \mathbb{N}^2, \leq_{\text{lex}}\rangle$ is of length $g_{\omega^2}(1) = 8$:

$$(1,1), (1,0), (0,5), (0,4), \ldots, (0,1), (0,0) \;. \tag{30}$$

5.3 Length Functions for the Product Ordering

More generally, the length function theorems for $\langle \mathbb{N}^d, \leq_\times\rangle$ provide larger upper bounds than the g_{ω^d} bound provided by Thm. 3.3 [28, 12, 17, 33, 1]. The following version from [33, Chap. 2] is easy to compare with Thm. 3.3:

Fact 5.1 ([33]). *Let $d \geq 0$ and $h(x) \overset{\text{def}}{=} d \cdot g(x)$. Then $L_{g,\mathbb{N}^d}(x) \leq h_{\omega^d}(dx)$.*

Fact 5.1 allows to bound the running time of programs proven to terminate with d transition invariants T_j, each shown well-founded through some g-controlled ranking function into $\mathbb{N}$. In particular, for linearly controlled ranking functions, d-dimensional transition invariants entail again upper bounds in $\mathbf{F}_{d+1}$, just like linearly controlled ranking functions into $\langle \mathbb{N}^d, \leq_{\text{lex}}\rangle$ do. Thus, at the coarse-grained level of the fast-growing complexity classes, the differences between Thm. 3.3 and Fact. 5.1 disappear.

5.4 Controlling Abstractions

The previous classifications into primitive recursive complexity classes $\mathbf{F}_{d+1}$ might be taken to imply that non-primitive recursive programs are beyond the reach of the current automated termination methods, which usually rely on the synthesis of affine ranking functions. This is not the case, as we can better see with the example of *size-change termination* proofs: Lee, Jones, and Ben-Amram [24] consider as their Example 3 the two-arguments Ackermann function:

```
a(m, n) = if m = 0 then n + 1 else
          if n = 0 then a(m−1, 1)
                   else a(m−1, a(m, n−1))
```

They construct a size-change graph on two variables to prove its termination. The longest decreasing sequence in such a graph is of length $O(n^2)$.[2] Here we witness an even larger gap between the actual program complexity and the complexity derived from its termination argument: the Ackermann function vs. an $O(n^2)$ bound.

The source of this apparent paradox is abstraction: the size-change graph for a(m, n) terminates if and only if the original program does, but its complexity is 'lost' during this abstraction. In the example of the Ackermann function, the call stack is abstracted away, whereas we should include it for Thm. 3.3 to apply. This is done by Dershowitz and Manna [15, Example 3], who prove the termination of the Ackermann function by exhibiting a H-controlled ranking function into $\langle \mathbb{M}(\mathbb{N}^2), \leq_{\mathrm{mset}} \rangle$, for which Thm. 3.3 yields an $O(H_{\omega^{\omega^2}}(n))$ complexity upper bound—this is pretty much optimal.

The question at this point is how to deal with abstractions. For size-change abstractions, Ben-Amram [3] shows for instance that the programs provable to terminate are always multiply recursive, but this type of analysis is missing for other abstraction techniques, e.g. for abstract interpretation ones [37].

6 Concluding Remarks

Length function theorems often seem to relate the length function $L_{g,A}$ for (g, n)-controlled bad sequences over a wqo $\langle A, \leq \rangle$ with a Cichoń function $h_{o(A,\leq)}$ indexed by the maximal order type $o(A, \leq)$ (recall Eq. (28)) for some 'reasonable' generative function h. This is certainly the case of e.g. Thm. 3.3, where $h(x) = g(x)$, but also of Fact. 5.1 where $h(x) = d \cdot g(x)$, and of the corresponding theorem in [32] for Higman's Lemma, where $h(x) = x \cdot g(x)$.

This is a relaxation of *Cichoń's Principle* [10], who observed that rewriting systems with a termination ordering of order type α [16] often had a complexity bounded by the slow-growing function G_α (defined by choosing $G(x) \overset{\text{def}}{=} x$ as generative function in Cichoń's hierarchy). A counter-example to the principle was given by Lepper [26] using the Knuth-Bendix order; however it did not

[2] Colcombet, Daviaud, and Zuleger [13] recently showed that the asymptotic worst-case complexity of a size-change graph is $\Theta(n^r)$ for a computable rational r.

disprove the relaxed version of Cichoń's Principle, where the generative function h can be chosen more freely. A recent analysis of generalised Knuth-Bendix orders by Moser [29] exhibits a counter-example to the relaxed version. An open question at the moment is therefore to find general conditions which ensure that this relaxed Cichoń Principle holds.

Acknowledgements. The author thanks Christoph Haase, Georg Moser, and Philippe Schnoebelen for helpful discussions.

References

1. Abriola, S., Figueira, S., Senno, G.: Linearizing bad sequences: Upper bounds for the product and majoring well quasi-orders. In: Ong, L., de Queiroz, R. (eds.) WoLLIC 2012. LNCS, vol. 7456, pp. 110–126. Springer, Heidelberg (2012)
2. Alias, C., Darte, A., Feautrier, P., Gonnord, L.: Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In: Cousot, R., Martel, M. (eds.) SAS 2010. LNCS, vol. 6337, pp. 117–133. Springer, Heidelberg (2010)
3. Ben-Amram, A.M.: General size-change termination and lexicographic descent. The Essence of Computation, pp. 3–17. Springer (2002)
4. Ben-Amram, A.M., Genaim, S.: Ranking functions for linear-constraint loops (2013), `http://arxiv.org/abs/1208.4041`
5. Ben-Amram, A.M., Vainer, M.: Bounded termination of monotonicity-constraint transition systems (preprint, 2014), `http://arxiv.org/abs/1202.4281`
6. Blass, A., Gurevich, Y.: Program termination and well partial orderings. ACM Trans. Comput. Logic 9(3) (2008)
7. Bonfante, G., Cichoń, A.E., Marion, J.Y., Touzet, H.: Algorithms with polynomial interpretation termination proof. J. Funct. Programming 11, 33–53 (2001)
8. Buchholz, W., Cichoń, E.A., Weiermann, A.: A uniform approach to fundamental sequences and hierarchies. Math. Logic Quart. 40(2), 273–286 (1994)
9. Bucholz, W.: Proof-theoretic analysis of termination proofs. Ann. Pure App. Logic 75(1–2), 57–65 (1995)
10. Cichoń, E.A.: Termination orderings and complexity characterisations. Proof Theory, pp. 171–194. Cambridge University Press (1993)
11. Cichoń, E.A., Tahhan Bittar, E.: Ordinal recursive bounds for Higman's Theorem. Theor. Comput. Sci. 201(1-2), 63–84 (1998)
12. Clote, P.: On the finite containment problem for Petri nets. Theor. Comput. Sci. 43, 99–105 (1986)
13. Colcombet, T., Daviaud, L., Zuleger, F.: Size-change abstraction and max-plus automata. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014, Part I. LNCS, vol. 8634, pp. 208–219. Springer, Heidelberg (2014)
14. Cook, B., See, A., Zuleger, F.: Ramsey vs. Lexicographic termination proving. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013 (ETAPS 2013). LNCS, vol. 7795, pp. 47–61. Springer, Heidelberg (2013)
15. Dershowitz, N., Manna, Z.: Proving termination with multiset orderings. Commun. ACM 22(8), 465–476 (1979)
16. Dershowitz, N., Okada, M.: Proof-theoretic techniques for term rewriting theory. In: LICS 1988, pp. 104–111 (1988)
17. Figueira, D., Figueira, S., Schmitz, S., Schnoebelen, P.: Ackermannian and primitive-recursive bounds with Dickson's Lemma. In: LICS 2011., pp. 269–278. IEEE (2011)

18. Floyd, R.W.: Assigning meaning to programs. Mathematical Aspects of Computer Science. In: Proceedings of Symposia in Applied Mathematics, vol. 19, pp. 19–32. AMS (1967)
19. Gulwani, S.: SPEED: Symbolic complexity bound analysis. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 51–62. Springer, Heidelberg (2009)
20. Hirokawa, N., Moser, G.: Automated complexity analysis based on the dependency pair method. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 364–379. Springer, Heidelberg (2008)
21. Hofbauer, D.: Termination proofs by multiset path orderings imply primitive recursive derivation lengths. Theor. Comput. Sci. 105(1), 129–140 (1992)
22. de Jongh, D.H.J., Parikh, R.: Well-partial orderings and hierarchies. Indag. Math. 39(3), 195–207 (1977)
23. Jouannaud, J.P., Lescanne, P.: On multiset orderings. Inf. Process. Lett. 15(2), 57–63 (1982)
24. Lee, C.S., Jones, N.D., Ben-Amram, A.M.: The size-change principle for program termination. In: POPL 2001, pp. 81–92. ACM (2001)
25. Lepper, I.: Derivation lengths and order types of Knuth-Bendix orders. Theor. Comput. Sci. 269(1-2), 433–450 (2001)
26. Lepper, I.: Simply terminating rewrite systems with long derivations. Arch. Math. Logic 43(1), 1–18 (2004)
27. Löb, M.H., Wainer, S.S.: Hierarchies of number theoretic functions, I. Arch. Math. Logic 13, 39–51 (1970)
28. McAloon, K.: Petri nets and large finite sets. Theor. Comput. Sci. 32(1-2), 173–183 (1984)
29. Moser, G.: KBOs, ordinals, subrecursive hierarchies and all that. J. Logic Comput. (to appear, 2014)
30. Podelski, A., Rybalchenko, A.: Transition invariants. In: LICS 2004. pp. 32–41. IEEE (2004)
31. Schmitz, S.: Complexity hierarchies beyond Elementary (2013), `http://arxiv.org/abs/1312.5686` (preprint)
32. Schmitz, S., Schnoebelen, P.: Multiply-recursive upper bounds with higman's lemma. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part II. LNCS, vol. 6756, pp. 441–452. Springer, Heidelberg (2011)
33. Schmitz, S., Schnoebelen, P.: Algorithmic aspects of wqo theory. Lecture notes (2012), `http://cel.archives-ouvertes.fr/cel-00727025`
34. Schmitz, S., Schnoebelen, P.: The power of well-structured systems. In: D'Argenio, P.R., Melgratti, H. (eds.) CONCUR 2013 – Concurrency Theory. LNCS, vol. 8052, pp. 5–24. Springer, Heidelberg (2013), `http://arxiv.org/abs/1402.2908`
35. Schwichtenberg, H., Wainer, S.S.: Proofs and Computation. Perspectives in Logic. Cambridge University Press (2012)
36. Turing, A.M.: Checking a large routine. In: EDSAC 1949, pp. 67–69 (1949)
37. Urban, C., Miné, A.: An abstract domain to infer ordinal-valued ranking functions. In: Shao, Z. (ed.) ESOP 2014 (ETAPS). LNCS, vol. 8410, pp. 412–431. Springer, Heidelberg (2014)
38. Weiermann, A.: Complexity bounds for some finite forms of Kruskal's Theorem. J. Symb. Comput. 18(5), 463–488 (1994)
39. Weiermann, A.: Termination proofs for term rewriting systems by lexicographic path orderings imply multiply recursive derivation lengths. Theor. Comput. Sci. 139(1-2), 355–362 (1995)