Complexity Hierarchies beyond Elementary

SYLVAIN SCHMITZ, LSV, ENS Cachan & CNRS & Inria, Université Paris-Saclay

We introduce a hierarchy of fast-growing complexity classes and show its suitability for completeness statements of many nonelementary problems. This hierarchy allows the classification of many decision problems with a nonelementary complexity, which occur naturally in areas such as logic, combinatorics, formal languages, and verification, with complexities ranging from simple towers of exponentials to Ackermannian and beyond.

Categories and Subject Descriptors: F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes; F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic

General Terms: Theory

Additional Key Words and Phrases: Fast-growing complexity, subrecursion, well-quasi-order

ACM Reference Format:
Sylvain Schmitz. 2016. Complexity hierarchies beyond elementary. ACM Trans. Comput. Theory 8, 1, Article 3 (February 2016), 36 pages.
DOI: http://dx.doi.org/10.1145/2858784

1. INTRODUCTION

Complexity classes, along with the associated notions of reductions and completeness, provide our best theoretical tools to classify and compare computational problems. The richness and liveness of this field can be experienced by taking a guided tour of the *Complexity Zoo*,[1] which presents succinctly most of the known specimens. The visitor will find there a wealth of classes at the frontier between tractability and intractability, starring the classes P and NP, as they help in understanding what can be solved efficiently by algorithmic means.

From this tractability point of view, it is not so surprising to find much less space devoted to the "truly intractable" classes, in the exponential hierarchy and beyond. Such classes are nevertheless quite useful for classifying problems and have been employed routinely in areas such as logic, combinatorics, formal languages, and verification since the 1970s and the exponential lower bounds proven by Meyer and Stockmeyer [Meyer 1975b; Stockmeyer and Meyer 1973].

Nonelementary problems. Actually, the two seminal articles of Meyer and Stockmeyer go further than mere exponential lower bounds: they respectively show that satisfiability of the weak monadic theory of one successor (WS1S) and equivalence of star-free

[1]https://complexityzoo.uwaterloo.ca.

This work was partially supported by ANR grant ReacHard 11-BS02-001-01.
Author's address: S. Schmitz, LSV, ENS de Cachan, 61 avenue du Président Wilson, 94235 CACHAN Cedex, France; email: schmitz@lsv.ens-cachan.fr.

DOI: http://dx.doi.org/10.1145/2858784

expressions (SFEq) are *nonelementary*, as they require space bounded above and below by towers of exponentials of height depending (elementarily) on the size of the input. Those are just two examples among many other problems with nonelementary complexities (e.g., see Meyer [1975a], Friedman [1999], and Vorobyov [2004]), but they are actually good representatives of problems with a tower of exponentials as complexity, i.e., one would expect them to be *complete* for some suitable complexity class.

What might then come as a surprise is the fact that presently, the Zoo does not provide any intermediate stops where classical problems like WS1S and SFEq would fit adequately: they are not in ELEMENTARY (henceforth ELEM), but the next class is PRIMITIVE-RECURSIVE (aka PR), which is far too big: WS1S and SFEq are not *hard* for PR under any reasonable notion of reduction. In other words, we seem to be missing a "TOWER" complexity class, which ought to sit somewhere between ELEM and PR. Going higher, we find a similar uncharted area between PR and RECURSIVE (aka R). These absences are not specific to the Complexity Zoo: on the contrary, they seem universal in textbooks on complexity theory, which seldom even mention ELEM or PR. Somewhat oddly, the complexities above R are better explored and can rely on the arithmetical and analytical hierarchies.

Drawing distinctions based on complexity characterizations can guide the search for practically relevant restrictions to the problems. In addition, nonelementary problems are much more pervasive now than in the 1970s, and they are also considered for practical applications, motivating the implementation of tools, e.g. MONA for WS1S [Elgaard et al. 1998]. It is therefore high time for the definition of hierarchies suited for their classification.

Our contribution. In this article, we propose an ordinal-indexed hierarchy $(\mathbf{F}_\alpha)_\alpha$ of *fast-growing* complexity classes for nonelementary complexities. Beyond the already mentioned TOWER $\stackrel{\text{def}}{=} \mathbf{F}_3$, for which WS1S and SFEq are examples of complete problems, this hierarchy includes nonprimitive-recursive classes, for which quite a few complete problems have arisen in the recent years, e.g.,

—$\mathbf{F}_\omega$ in Mayr and Meyer [1981], Urquhart [1999], Schnoebelen [2010], Figueira [2012], Bresolin et al. [2012], Lazić et al. [2013], Hofman and Totzke [2014], and Hague [2014].
—$\mathbf{F}_{\omega^\omega}$ in Chambart and Schnoebelen [2008b], Ouaknine and Worrell [2007], Lasota and Walukiewicz [2008], Atig et al. [2010], Chambart and Schnoebelen [2007], Barceló et al. [2013] and Rosa-Velardo [2014];
—$\mathbf{F}_{\omega^{\omega^\omega}}$ in Haddad et al. [2012]; and
—$\mathbf{F}_{\varepsilon_0}$ in Haase et al. [2014] and Decker and Thoma [2015].

The classes $\mathbf{F}_\alpha$ are related to the Grzegorczyk $(\mathscr{E}^k)_k$ [Grzegorczyk 1953] and extended Grzegorczyk $(\mathscr{F}_\alpha)_\alpha$ [Löb and Wainer 1970] hierarchies, which have been used in complexity statements for nonelementary bounds. The $(\mathscr{F}_\alpha)_\alpha$ classes are very well suited for characterizing various classes of functions, i.e., computed by forms of `for` programs [Meyer and Ritchie 1967] or terminating `while` programs [Fairtlough and Wainer 1992], or provably total in fragments of Peano arithmetic [Fairtlough and Wainer 1998; Schwichtenberg and Wainer 2012], and they characterize some important milestones like ELEM or PR. However, they are too large to classify our decision problems and do not lead to *completeness* statements—in fact, one can show that there are *no* "ELEM-complete" nor "PR-complete" problems (see Section 2). However, our $\mathbf{F}_\alpha$ share several nice properties with the $\mathscr{F}_\alpha$ classes, i.e., they form a strict hierarchy (Section 5) and are *robust* to slight changes in their generative functions and to changes in the underlying model of computation (Section 4) .

To argue for the suitability of the classes $\mathbf{F}_\alpha$ for the classification of high-complexity problems, we sketch two completeness proofs in Section 3 and present an already long

list of complete problems for $\mathbf{F}_\omega$ and beyond in Section 6. A general rule of thumb seems to be that statements of the form "L is in $\mathscr{F}_\alpha$ but not in $\mathscr{F}_\beta$ for any $\beta < \alpha$" found in the literature can often be replaced by the much more precise "L is $\mathbf{F}_\alpha$-complete."

Of course, there are essential limitations to our approach: there is no hope of defining such ordinal-indexed hierarchies that would exhaust R using sensible ordinal notations [Feferman 1962]; this is called the *subrecursive stumbling block* in Section 5.1 Schwichtenberg and Wainer [2012]. Our aim here is more modestly to provide suitable definitions "from below" for naturally occurring complexity classes above ELEM.

In an attempt not to drown the reader in the details of subrecursive functions and their properties, most of the technical contents appear in Appendix A at the end of the article.

2. FAST-GROWING COMPLEXITY CLASSES

In this section, we define the complexity classes $\mathbf{F}_\alpha$. We rely on the fast-growing functions F_α of Löb and Wainer [1970] as a standard against which we can measure high complexities (compare to Section 2.2.1). In logic and recursion theory, these functions are used to generate the classes of functions $\mathscr{F}_\alpha$ when closed under substitution and limited primitive recursion (see Section 5.3.1). However, these classes are not suitable for our complexity classification objectives: the class $\mathscr{F}_\alpha$ contains indeed arbitrary finite compositions of the function F_α. Instead, in Section 2.3, we define each $\mathbf{F}_\alpha$ class as the class of problems decidable within time bounded by a single application of F_α composed with any function p already defined in the lower levels $\mathscr{F}_\beta$ for $\beta < \alpha$.

These hierarchies of functions, function classes, and complexity classes that we employ to deal with nonelementary complexities are all indexed using ordinals, and we reuse the very rich literature on subrecursion (e.g., Rose [1984], Odifreddi [1999] and Schwichtenberg and Wainer [2012]). We strive to employ notations compatible with those of Chapter 4 of Schwichtenberg and Wainer [2012] and refer the interested reader to their monograph for proofs and additional material.

2.1. Cantor Normal Forms and Fundamental Sequences

In this article, we only deal with ordinals that can be denoted syntactically as terms in Cantor normal form (CNF):

$$\alpha = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_n} \cdot c_n \text{ where } \alpha > \alpha_1 > \cdots > \alpha_n \text{ and } \omega > c_1, \ldots, c_n > 0 \qquad \text{(CNF)}$$

and hereditarily $\alpha_1, \ldots, \alpha_n$ are also written in CNF. In this representation, $\alpha = 0$ if and only if $n = 0$. An ordinal α with CNF of form $\alpha' + 1$ is called a *successor* ordinal—it has $n > 0$ and $\alpha_n = 0$, and otherwise if $\alpha > 0$, it is called a *limit* ordinal and can be written as $\gamma + \omega^\beta$ by setting $\gamma = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_n} \cdot (c_n - 1)$ and $\beta = \alpha_n$. We usually employ "λ" to denote limit ordinals.

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

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

This particular assignment of fundamental sequences satisfies, e.g., $0 < \lambda(x) < \lambda(y)$ for all $x < y$ and limit ordinals λ. For instance, $\omega(x) = x + 1$, $(\omega^{\omega^4} + \omega^{\omega^{3}+\omega^2})(x) = \omega^{\omega^4} + \omega^{\omega^3+\omega\cdot(x+1)}$. We also consider the ordinal ε_0, which is the supremum of all ordinals writable in CNF, as a limit ordinal with fundamental sequence defined by $\varepsilon_0(0) \stackrel{\text{def}}{=} \omega$ and $\varepsilon_0(x+1) \stackrel{\text{def}}{=} \omega^{\varepsilon_0(x)}$, i.e., a tower of ω's of height $x + 1$.

2.2. The Extended Grzegorczyk Hierarchy

This is an ordinal-indexed infinite hierarchy of classes $(\mathscr{F}_\alpha)_{\alpha<\varepsilon_0}$ of functions with argument(s) and images in $\mathbb{N}$ [Löb and Wainer 1970]. The extended Grzegorczyk hierarchy has multiple natural characterizations—e.g., as `loop` programs for $\alpha < \omega$ [Meyer and Ritchie 1967], as ordinal-recursive functions with bounded growth [Wainer 1970], as functions computable with restricted resources as we will see in (5), and as functions that can be proven total in fragments of Peano arithmetic [Fairtlough and Wainer 1998].

2.2.1. Fast-Growing Functions. At the heart of each $\mathscr{F}_\alpha$ lies the αth *fast-growing function* $F_\alpha\colon \mathbb{N} \to \mathbb{N}$, which is defined inductively on the ordinal index: as the successor function at index 0

$$F_0(x) \stackrel{\text{def}}{=} x + 1, \tag{2}$$

by iteration at successor indices $\alpha + 1$

$$F_{\alpha+1}(x) \stackrel{\text{def}}{=} F_\alpha^{\omega(x)}(x) = \overbrace{F_\alpha(\cdots(F_\alpha}^{\omega(x)\text{ times}}(x))\cdots), \tag{3}$$

and by diagonalization on the fundamental sequence at limit indices λ

$$F_\lambda(x) \stackrel{\text{def}}{=} F_{\lambda(x)}(x). \tag{4}$$

For instance, $F_1(x) = 2x + 1$, $F_2(x) = 2^{x+1}(x + 1) - 1$, F_3 is a nonelementary function that grows faster than $\text{tower}(x) \stackrel{\text{def}}{=} 2^{\cdot^{\cdot^{\cdot^{2}}}}\}x\text{ times}$, F_ω a nonprimitive-recursive "Ackermannian" function, F_{ω^ω} a nonmultiply-recursive "hyper-Ackermannian" function, and $F_{\varepsilon_0}(x)$ cannot be proven total in Peano arithmetic. For every α, the F_α function is strictly *monotone* in its argument, i.e., $x < y$ implies $F_\alpha(x) < F_\alpha(y)$. As $F_\alpha(0) = 1$, it is therefore also strictly *expansive*, i.e., $F_\alpha(x) > x$ for all x.

2.2.2. Computational Characterization. The extended Grzegorczyk hierarchy itself is defined by means of recursion schemes with the $(F_\alpha)_\alpha$ as generators (see Section 5.3.1). Nevertheless, for $\alpha \geq 2$, each of its levels $\mathscr{F}_\alpha$ is also characterized as a class of functions computable with bounded resources [Wainer 1970]. More precisely, for $\alpha \geq 2$, it is the class of functions computable by deterministic Turing machines in time bounded by $O(F_\alpha^c(n))$ for some constant c, when given an input of size n:

$$\mathscr{F}_\alpha = \bigcup_{c<\omega} \text{FDTIME}\left(F_\alpha^c(n)\right). \tag{5}$$

Note that the choice between deterministic and nondeterministic, or between time-bounded and space-bounded computations in (5), is irrelevant, because $\alpha \geq 2$ and F_2 is already a function of exponential growth.

2.2.3. Main Properties. Each class $\mathscr{F}_\alpha$ is closed under (finite) composition. Every function f in $\mathscr{F}_\alpha$ is *honest*, i.e., it can be computed in time bounded by some function also in $\mathscr{F}_\alpha$ [Wainer 1970; Fairtlough and Wainer 1998]—this is a relaxation of the *time constructible* condition, which asks instead for computability in time $O(f(n))$. Since each f in $\mathscr{F}_\alpha$ is also bounded by F_α^c for some c [Löb and Wainer 1970, Theorem 2.10], this means that

$$\mathscr{F}_\alpha = \bigcup_{f\in\mathscr{F}_\alpha} \text{FDTIME}(f(n)). \tag{6}$$

In particular, for every α, the function F_α belongs to $\mathscr{F}_\alpha$, and therefore F_α^c also belongs to $\mathscr{F}_\alpha$.

Every f in $\mathscr{F}_\beta$ is also eventually bounded by F_α if $\beta < \alpha$ [Löb and Wainer 1970], i.e., there exists a rank x_0 such that for all $x_1, \ldots, x_n$, if $\max_i x_i \geq x_0$, then $f(x_1, \ldots, x_n) \leq F_\alpha(\max_i x_i)$—a fact that we will use copiously. However, for all $\alpha > \beta > 0$, $F_\alpha \notin \mathscr{F}_\beta$, and the hierarchy $(\mathscr{F}_\alpha)_{\alpha<\varepsilon_0}$ is therefore strict for $\alpha > 0$.

2.2.4. Milestones. At the lower levels, $\mathscr{F}_0 = \mathscr{F}_1$ contains (among others) all linear functions (see Section 5.3.2). However, in this article, we focus on the nonelementary classes by restricting ourselves to $\alpha \geq 2$. Writing

$$\mathscr{F}_{<\alpha} \stackrel{\text{def}}{=} \bigcup_{\beta<\alpha} \mathscr{F}_\beta, \tag{7}$$

we find, i.e., $\mathscr{F}_2 = \mathscr{F}_{<3} = \text{FE}\textsc{lem}$, the set of Kalmar-elementary functions; $\mathscr{F}_{<\omega} = \text{FPR}$, the set of primitive-recursive functions; $\mathscr{F}_{<\omega^\omega} = \text{FMR}$, the set of multiply-recursive functions; and $\mathscr{F}_{<\varepsilon_0} = \text{FOR}$, the set of ordinal-recursive functions (up to ε_0). We are dealing here with classes of functions, but writing $\mathscr{F}^*_\alpha$ for the restriction of $\mathscr{F}_\alpha$ to $\{0,1\}$-valued functions, i.e.,

$$\mathscr{F}^*_\alpha = \bigcup_{c<\omega} \text{DT}\textsc{ime}\left(F^c_\alpha(n)\right), \qquad \mathscr{F}^*_{<\alpha} \stackrel{\text{def}}{=} \bigcup_{\beta<\alpha} \mathscr{F}^*_\beta, \tag{8}$$

we obtain the corresponding classes for decision problems $\mathscr{F}^*_{<3} = \textsc{Elem}$, $\mathscr{F}^*_{<\omega} = \text{PR}$, $\mathscr{F}^*_{<\omega^\omega} = \text{MR}$, and $\mathscr{F}^*_{<\varepsilon_0} = \text{OR}$.

2.3. Fast-Growing Complexity Classes

Unfortunately, the classes in the extended Grzegorczyk hierarchy are not quite satisfying for some interesting problems, which are nonelementary (or or nonprimitive recursive or nonmultiply recursive, etc) but only *barely* so. The issue is that complexity classes like, e.g., $\mathscr{F}^*_3$, which is the first class to contain nonelementary problems, are very large: i.e., $\mathscr{F}^*_3$ contains problems that require space $F^{100}_3(n)$, more than a hundredfold compositions of towers of exponentials. As a result, hardness for $\mathscr{F}^*_3$ cannot be obtained for many classical examples of nonelementary problems.

We therefore introduce smaller classes of problems:

$$\mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{p\in\mathscr{F}_{<\alpha}} \text{DT}\textsc{ime}(F_\alpha(p(n))). \tag{9}$$

In contrast with $\mathscr{F}^*_\alpha$ in (8), only a single application of F_α is possible, composed with some "lower" *reduction* function p from $\mathscr{F}_{<\alpha}$. As previously, the choice of DTime rather than NTime or Space is irrelevant for $\alpha \geq 3$ (see Lemma 4.6 later).

This definition yields, i.e., the desired class $\textsc{Tower} \stackrel{\text{def}}{=} \mathbf{F}_3$, closed under elementary reductions (i.e., reductions in $\mathscr{F}_2$), but also a class $\textsc{Ack} \stackrel{\text{def}}{=} \mathbf{F}_\omega$ of Ackermannian problems closed under primitive-recursive reductions, a class $\textsc{HAck} \stackrel{\text{def}}{=} \mathbf{F}_{\omega^\omega}$ of hyper-Ackermannian problems closed, e.g., under multiply-recursive reductions. In each case, we can think of $\mathbf{F}_\alpha$ as the class of problems not solvable with resources in $\mathscr{F}_{<\alpha}$, but barely so: nonelementary problems for $\mathbf{F}_3$, nonprimitive-recursive ones for $\mathbf{F}_\omega$, nonmultiply-recursive ones for $\mathbf{F}_{\omega^\omega}$, and so on. Figure 1 presents the first main stops of the hierarchy.

2.3.1. Reduction Classes. Of course, we could replace in (9) the class of reductions $\mathscr{F}_{<\alpha}$ by a more traditional one, like logarithmic space (FL) or polynomial time (FP) functions. However, we feel that our definition in (9) better captures the intuition that we have of a problem being "complete for F_α." Moreover, using at least $\mathscr{F}_2$ as our class of reductions allows one to effectively compute the F_α function in the functional version $\text{F}\mathbf{F}_\alpha$ of $\mathbf{F}_\alpha$

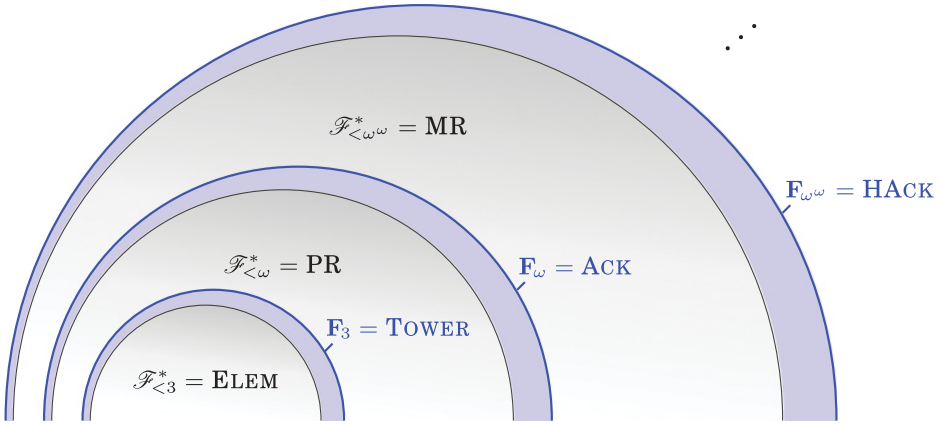

Fig. 1. Some complexity classes beyond ELEM.

(see Section 5.1), leading to interesting combinatorial algorithms (see Section 3.2.3 for an example).

Unless stated differently, we always assume many-one $\mathscr{F}_{<\alpha}$ reductions when discussing hardness for $\mathbf{F}_\alpha$ in the remainder of this article, but we could just as easily consider Turing reductions (see Section 4.2.3).

2.3.2. Basic $\mathbf{F}_\alpha$-Complete Problems. By (9), $\mathbf{F}_\alpha$-hardness proofs can reduce from the acceptance problem of some input string x by some deterministic Turing machine M working in time $F_\alpha(p(n))$ for some p in $\mathscr{F}_{<\alpha}$. This can be simplified to a machine M' working in time $F_\alpha(n)$. Indeed, because p in $\mathscr{F}_{<\alpha}$ is honest, $p(n)$ can be computed in $\mathscr{F}_{<\alpha}$. Thus, the acceptance of x by M can be reduced to the acceptance problem of a #-padded input string $x' \stackrel{\text{def}}{=} x\#^{p(|x|)-|x|}$ of length $p(|x|)$ by a machine M' that simulates M and treats # as a blank symbol—now M' works in time $F_\alpha(n)$. Another similarly basic $\mathbf{F}_\alpha$-hard problem is the halting problem for Minsky machines with the sum of counters bounded by $F_\alpha(n)$ (see Fischer et al. [1968]).

To sum up, by definition of the $(\mathbf{F}_\alpha)_\alpha$ classes, we have the following two $\mathbf{F}_\alpha$-complete problems—which incidentally have been used in most of the master reductions in the literature to prove nonprimitive-recursiveness, nonmultiple-recursiveness, and other hardness results [Jančar 2001; Urquhart 1999; Schnoebelen 2010; Chambart and Schnoebelen 2008b; Haddad et al. 2012; Haase et al. 2014; Lazić et al. 2013; Rosa-Velardo 2014; Decker and Thoma 2015]:

F_α-Bounded Turing machine acceptance (F_α-TM).

Instance: A deterministic Turing machine M working in time F_α and an input x.
Question: Does M accept x?

F_α-Bounded Minsky machine halting (F_α-MM).

Instance: A deterministic Minsky machine M with sum of counters bounded by $F_\alpha(|M|)$.
Question: Does M halt?

See Section 6 for a catalogue of natural complete problems, which should be easier to employ in reductions.

3. FAST-GROWING COMPLEXITIES IN ACTION

Now we present two short tutorials for the use of fast-growing complexities, namely for the equivalence problem for start-free expressions (Section 3.1) and reachability in lossy counter systems (Section 3.2), pointing to the relevant technical results from later sections. In each case, we also briefly discuss the palliatives employed so far in the literature for expressing such complexities.

3.1. A TOWER-Complete Example

Such an example can be found in the seminal paper of Stockmeyer and Meyer [1973] and is quite likely already known by many readers. Define a *star-free expression* over some alphabet Σ as a term e with abstract syntax

$$e ::= a \mid \varepsilon \mid \emptyset \mid e + e \mid ee \mid \neg e,$$

where "a" ranges over Σ and "ε" denotes the empty string. Such expressions are inductively interpreted as languages included in Σ^* by

$$[\![a]\!] \stackrel{\text{def}}{=} \{a\} \qquad [\![\varepsilon]\!] \stackrel{\text{def}}{=} \{\varepsilon\} \qquad [\![\emptyset]\!] \stackrel{\text{def}}{=} \emptyset$$
$$[\![e_1 + e_2]\!] \stackrel{\text{def}}{=} [\![e_1]\!] \cup [\![e_2]\!] \qquad [\![e_1 e_2]\!] \stackrel{\text{def}}{=} [\![e_1]\!] \cdot [\![e_2]\!] \qquad [\![\neg e]\!] \stackrel{\text{def}}{=} \Sigma^* \setminus [\![e]\!].$$

The decision problem SFEq asks, given two such expressions e_1, e_2, whether they are *equivalent*, i.e., whether $[\![e_1]\!] = [\![e_2]\!]$. Stockmeyer and Meyer [1973] show that this problem is hard for tower$(\log n)$ space under FL reductions if $|\Sigma| \geq 2$. The problem WS1S can be shown similarly hard thanks to a reduction from SFEq.

3.1.1. Completeness. Recall that TOWER is defined as $\mathbf{F}_3$, i.e., by the instantiation of (9) for $\alpha = 3$, as the problems decidable by a Turing machine working in time F_3 of some elementary function of the input size:

$$\text{TOWER} \stackrel{\text{def}}{=} \mathbf{F}_3 = \bigcup_{p \in \text{FELEM}} \text{DTIME}\left(F_3(p(n))\right). \tag{10}$$

Once hardness for TOWER$(\log n)$ is established, hardness for TOWER under elementary reductions is immediate; a detailed proof can apply Theorem 4.1 and Equation (22) to show that

$$\text{TOWER} = \bigcup_{p \in \text{FELEM}} \text{SPACE}(\text{tower}(p(n)) \tag{11}$$

and use a padding argument as in Section 2.3.2 to conclude.

That SFEq is in TOWER can be checked using an automaton-based algorithm: construct automata recognizing $[\![e_1]\!]$ and $[\![e_2]\!]$, respectively, using determinization to handle each complement operator at the expense of an exponential blowup and check equivalence of the obtained automata in PSPACE—the overall procedure is in space polynomial in TOWER(n), thus in $\mathbf{F}_3$. A similar automata-based procedure yields the upper bound for WS1S.

3.1.2. Discussion. Regarding upper bounds, there was a natural candidate in the literature for the missing class TOWER: Grzegorczyk [1953] defines an infinite hierarchy of function classes $(\mathscr{E}^k)_{k \in \mathbb{N}}$ inside FPR with $\mathscr{E}^{k+1} = \mathscr{F}_k$ for $k \geq 2$. This yields FELEM $= \mathscr{E}^3$, and the TOWER function is in $\mathscr{E}^4 \setminus \mathscr{E}^3$. Thus, WS1S and SFEq are in "time $\mathscr{E}^4$," and such a notation has occasionally been employed, i.e., for β-Eq, the β equivalence of simply typed λ-terms [Statman 1979; Schwichtenberg 1982; Beckmann 2001]. Again, we face the issue that $\mathscr{E}^4$ is much too large a resource bound, as it contains, i.e., all of the finite

iterates of the Tower function, and there is therefore no hope of proving the hardness for $\mathscr{E}^4$ of WS1S, SFEq, or indeed β-Eq, at least if using a meaningful class of reductions.

Regarding nonelementary lower bounds, recent papers typically establish hardness for k-ExpTime (or k-ExpSpace) *for infinitely many* k (possibly through a suitable parameterization of the problem at hand), such as by reducing from the acceptance of an input of size n by a $\underbrace{2^{\cdot^{\cdot^{2^n}}}}_{k \text{ times}}$ time-bounded Turing machine. Provided that such a lower bound argument is *uniform* for those infinitely many k, it immediately yields a Tower-hardness proof, by choosing $k \geq n$. On a related topic, note that in contrast with, e.g., the relationship between PH and PSpace, because the exponential hierarchy is known to be strict, we know for certain that

—for all k, k-ExpTime $\subsetneq$ Elem $= \bigcup_k k$-ExpTime,
—there are no "Elem-complete problems," and
—Elem $\subsetneq$ Tower.

3.2. An Ack-Complete Example

Possibly the most popular complete problem for Ack in use in reductions, LCM Reachability asks whether a given configuration is reachable in a *lossy counter machine* (LCM) [Schnoebelen 2010]. Such counter machines are syntactically identical to Minsky machines $\langle Q, \mathtt{C}, \delta, q_0 \rangle$, where transitions $\delta \subseteq Q \times \mathtt{C} \times \{\mathtt{=0?}, \mathtt{++}, \mathtt{--}\} \times Q$ operate on a set $\mathtt{C}$ of counters through *zero-tests* `c=0?`, *increments* `c++`, and *decrements* `c--`. However, the semantics of an LCM differ from the usual, "reliable" semantics of a counter machine in that the counter values can decrease in an uncontrolled manner at any point of the execution. These unreliable behaviors make several problems decidable on LCMs, contrasting with the situation with Minsky machines.

Formally, a configuration $\sigma = (q, \bar{\mathtt{v}})$ associates a control location q in Q with a counter valuation $\bar{\mathtt{v}}$ in $\mathbb{N}^{\mathtt{C}}$, i.e., counter values can never go negative. A transition of the form $(q, \mathtt{c}, \mathtt{op}, q')$ defines a computation step $(q, \bar{\mathtt{v}}) \to (q', \bar{\mathtt{v}}')$ where $\bar{\mathtt{v}}(\mathtt{c}') \leq \bar{\mathtt{v}}'(\mathtt{c}')$ for all $\mathtt{c} \neq \mathtt{c}'$ in $\mathtt{C}$, and

—if $\mathtt{op} = \mathtt{=0?}$, then $\bar{\mathtt{v}}(\mathtt{c}) \geq \bar{\mathtt{v}}'(\mathtt{c}) = 0$,
—if $\mathtt{op} = \mathtt{++}$, then $\bar{\mathtt{v}}(\mathtt{c}) + 1 \geq \bar{\mathtt{v}}'(\mathtt{c})$, and
—if $\mathtt{op} = \mathtt{--}$, then $\bar{\mathtt{v}}(\mathtt{c}) \geq \bar{\mathtt{v}}'(\mathtt{c}) + 1$.

Let the initial configuration be $(q_0, \bar{0})$. The reachability problem for such a system asks whether a given configuration τ can be reached in a finite number of steps, i.e., whether $(q_0, \bar{0}) \to^* \tau$. The hardness proof of Schnoebelen [2010] immediately yields that this problem is Ack-hard (see also Urquhart [1999] and Schnoebelen [2002]), where Ack is defined as an instance of (9): it is the class of problems decidable with F_ω resources of some primitive-recursive function of the input size:

$$\text{Ack} \stackrel{\text{def}}{=} \mathbf{F}_\omega = \bigcup_{p \in \text{FPR}} \text{DTime}\big(F_\omega(p(n))\big). \tag{12}$$

3.2.1. Decidability of LCM. LCMs define *well-structured* transition systems over the set of configurations $Q \times \mathbb{N}^{\mathtt{C}}$, for which generic algorithms have been designed [Abdulla et al. 2000; Finkel and Schnoebelen 2001], which rely on the existence of a well-quasi-ordering (WQO; see Kruskal [1972]) over the set of configurations. The particular variant of the algorithm we present here is well suited for a complexity analysis and is taken from Schmitz and Schnoebelen [2013].

Call a sequence of configurations $\sigma_0, \sigma_1, \ldots, \sigma_n$ a *witness* if $\sigma_0 = \tau$ is the target configuration, $\sigma_n = (q_0, \bar{0})$ is the initial configuration, and $\sigma_{i+1} \to \sigma_i$ for all $0 \leq i < n$. An

instance of LCM is positive if and only if there exists a witness, which we will search for backward, starting from τ and attempting to reach the initial configuration $(q_0, \bar{0})$.

Consider the ordering over configurations defined by $(q, \bar{\mathrm{v}}) \leq (q', \bar{\mathrm{v}}')$ if and only if $q = q'$ and $\bar{\mathrm{v}} \leq_\times \bar{\mathrm{v}}'$, the latter being defined as $\bar{\mathrm{v}}(\mathtt{c}) \leq \bar{\mathrm{v}}'(\mathtt{c})$ for all $\mathtt{c}$ in $\mathtt{C}$. Observe that if $\sigma_0, \sigma_1, \ldots, \sigma_n$ is a *shortest* witness, then for all $i < j$, $\sigma_i \not\leq \sigma_j$, i.e., it is a *bad* sequence for $\leq$, or we could have picked σ_j at step i and obtained a strictly shorter witness. Furthermore, if at some step i there existed $s'_i \leq s_i$ with $s'_i \to s_{i-1}$, then we could substitute s'_i for s_i and still have a witness, because $s_{i+1} \to s'_i$. Thus, if there exists a witness, then there is a *minimal bad* one, i.e., a bad one where for all $0 < i < n$, $\sigma_{i+1} \in \mathrm{MinPre}(\sigma_i)$ where $\mathrm{MinPre}(\sigma) \stackrel{\text{def}}{=} \min_{\leq}\{\sigma' \mid \sigma' \to \sigma\}$.

Now, because Q and $\mathtt{C}$ are finite, $(Q \times \mathbb{N}^{\mathtt{C}}, \leq)$ is a well-quasi-order by Dickson's lemma, and thus

(i) for all i, the set $\mathrm{MinPre}(\sigma_i)$ is finite, and
(ii) any bad sequence, i.e., any sequence $\sigma_0, \sigma_1, \ldots$ where $\sigma_i \not\leq \sigma_j$ for all $i < j$, is finite.

Therefore, an algorithm for LCM can proceed by exploring a tree of prefixes of potential minimal witnesses, which has finite degree by (i) and finite height by (ii), hence by Kőnig's lemma is finite.

3.2.2. Length Function Theorems. A nondeterministic version of this search for a witness for LCM will see its complexity depend essentially on the height of the tree, i.e., on the length of bad sequences. Define the size of a configuration as its infinity norm $|(q, \bar{\mathrm{v}})| = \max_{\mathtt{c} \in \mathtt{C}} \bar{\mathrm{v}}(\mathtt{c})$, and note that any σ in $\mathrm{MinPre}(\sigma_i)$ is of size $|\sigma| \leq |\sigma_i| + 1$. This means that in any sequence $\sigma_0, \sigma_1, \ldots$ where $\tau = \sigma_0$ and $\sigma_{i+1} \in \mathrm{MinPre}(\sigma_i)$ for all i, $|\sigma_i| \leq |\tau| + i = \mathrm{succ}^i(|\tau|)$, the ith iterate of the successor function $\mathrm{succ}(x) \stackrel{\text{def}}{=} x + 1$. We call such a sequence *controlled* by succ.

What a *length function theorem* provides is an upper bound on the length of controlled bad sequences over a WQO, depending on the control function—here the successor function—and the maximal order type of the WQO—here $\omega^{|\mathtt{C}|} \cdot |Q|$. In our case, the theorems in Schmitz and Schnoebelen [2011, 2012] provide an

$$F^{|Q|}_{h,|\mathtt{C}|}(|\tau|) \leq F_{h,\omega}(\max\{|\mathtt{C}|, |Q|, |\tau|\}) \stackrel{\text{def}}{=} \ell \tag{13}$$

upper bound on both this length and the maximal size of any configuration in the sequence, where

—$h\colon \mathbb{N} \to \mathbb{N}$ is an increasing polynomial function (which depends on the control function) and
—for any increasing $h\colon \mathbb{N} \to \mathbb{N}$, $(F_{h,\alpha})_\alpha$ is a *relativized* fast-growing hierarchy that uses h instead of the successor function as base function with index 0:

$$F_{h,0}(x) \stackrel{\text{def}}{=} h(x), \qquad F_{h,\alpha+1}(x) \stackrel{\text{def}}{=} F^{\omega(x)}_{h,\alpha}(x), \qquad F_{h,\lambda}(x) \stackrel{\text{def}}{=} F_{h,\lambda(x)}(x). \tag{14}$$

3.2.3. A Combinatorial Algorithm. We have established an upper bound on the length of a shortest minimal witness, entailing that if a witness exists, then it is of length bounded by ℓ defined in (13). This bound can be exploited by a nondeterministic *forward* algorithm, which

(1) computes ℓ in a first phase: as we will see with Theorem 5.1, this can be performed in time $F_{h,\omega}(e(n)))$ for some elementary function e,
(2) then nondeterministically explores the reachable configurations, starting from the initial configuration $(q_0, \bar{0})$ and attempting to reach the target configuration τ—but aborts if the upper bound on the length is reached. This second phase uses at most ℓ steps, and each step can be performed in time polynomial in the size of the current

configuration, itself bounded by ℓ. The whole phase can thus be performed in time polynomial in ℓ, which is bounded by $F_{h,\omega}(f(n))$ for some primitive-recursive f by Lemma 4.6.

Thus, the overall complexity of this algorithm can be bounded by $F_{h,\omega}(p(n))$ where h and p are primitive-recursive. Because by Corollary 4.3 and Equation (22), for any primitive-recursive strictly increasing h,

$$\textsc{Ack} = \bigcup_{p\in\mathrm{FPR}} \textsc{NTime}\big(F_{h,\omega}(p(n))\big), \tag{15}$$

this means that LCM is in Ack.

3.2.4. Discussion. The oldest statement of Ack-completeness (under polynomial time Turing reductions) of which we are aware is due to Clote [1986] for FCP, the finite containment problem for Petri nets (see Section 6.1.1). As observed by Clote, his definition of Ack as $\textsc{DTime}\big(F_\omega(n)\big)$ is somewhat problematic, since the class is not robust under changes in the model of computation, i.e., RAM versus multitape Turing machines. A similar issue arises with the definition $\bigcup_{c<\omega}\textsc{DTime}(F_\omega(n+c))$ employed in Haddad et al. [2012]: although robust under changes in the model of computation, it is not closed under reductions. Those classes are too tight to be convenient.

Conversely, stating that a problem is "in $\mathscr{F}_\omega^*$ but not in $\mathscr{F}_k^*$ for any k" (e.g., Figueira et al. [2011]) is much less informative than stating that it is $\mathbf{F}_\omega$-complete: $\mathscr{F}_\omega^*$ is too large to allow for completeness statements (see Section 5).

4. ROBUSTNESS

In the applications of fast-growing classes we discussed in Sections 3.1 and 3.2, we relied on both counts on their "robustness" to minor changes in their definition. More precisely, we employed space or time hierarchies indifferently, and alternative generative functions: first for the lower bound of SFEq and WS1S, when we used the tower function instead of F_3 in the reduction, and later for the upper bound of LCM, where we relied on a relativized version of F_ω. In this section, we prove these and other small changes to be innocuous.

4.1. Generative Functions

There are many variants for the definition of the fast-growing functions $(F_\alpha)_\alpha$, but they are all known to generate essentially the same hierarchy $(\mathscr{F}_\alpha)_\alpha$.[2] Nevertheless, because the fast-growing complexity classes $\mathbf{F}_\alpha$ we defined are smaller, there is no guarantee for these classical results to hold for them.

4.1.1. Ackermann Hierarchy. We start with one particular variant, which is rather common in the literature: define $A_\alpha\colon\mathbb{N}\to\mathbb{N}$ for $\alpha>0$ by

$$A_1(x) \stackrel{\text{def}}{=} 2x, \qquad A_{\alpha+1}(x) \stackrel{\text{def}}{=} A_\alpha^x(1), \qquad A_\lambda(x) \stackrel{\text{def}}{=} A_{\lambda(x)}(x). \tag{16}$$

The hierarchy differs in the treatment of successor indices, where the argument is reset to 1 instead of keeping x as in (3). This definition results, i.e., in $A_2(x)=2^x$ and $A_3(x)=\mathrm{tower}(x)$ and is typically used in lower bound proofs.

[2]See Ritchie [1965] and Löb and Wainer [1970, pp. 48–51] for such results—and the works of Weiermann et al. on phase transitions for investigations of when changes *do* have an impact, e.g., Omri and Weiermann [2009].

We can define a hierarchy of decision problems generated from the $(A_\alpha)_\alpha$ by analogy with (9):

$$\mathbf{A}_\alpha \stackrel{\text{def}}{=} \bigcup_{p \in \mathscr{F}_{<\alpha}} \text{DTIME}(A_\alpha(p(n))). \tag{17}$$

For two functions $g: \mathbb{N} \to \mathbb{N}$ and $h: \mathbb{N} \to \mathbb{N}$, let us write $g \leq h$ if $g(x) \leq h(x)$ for all x in $\mathbb{N}$. Because $A_\alpha \leq F_\alpha$ for all $\alpha > 0$, it follows that $\mathbf{A}_\alpha \subseteq \mathbf{F}_\alpha$. The converse inclusion also holds: to prove it, it suffices to exhibit for all $\alpha > 0$ a function p_α in $\mathscr{F}_{<\alpha}$ such that $F_\alpha \leq A_\alpha \circ p_\alpha$. It turns out that a uniform choice $p_\alpha(x) \stackrel{\text{def}}{=} 6x+5$ fits those requirements—it is a linear function in $\mathscr{F}_0$ and $F_\alpha \leq A_\alpha \circ p_\alpha$ as shown in Lemma A.4, and thus we have the following theorem.

THEOREM 4.1. *For all $\alpha > 0$, $\mathbf{A}_\alpha = \mathbf{F}_\alpha$.*

4.1.2. Relativized Hierarchies. Another means of defining a variant of the fast-growing functions is to pick a different definition for F_0: recall the relativized fast-growing functions employed in (14). The corresponding relativized complexity classes are then defined by

$$\mathbf{F}_{h,\alpha} \stackrel{\text{def}}{=} \bigcup_{p \in \mathscr{F}_{<\alpha}} \text{DTIME}(F_{h,\alpha}(p(n))). \tag{18}$$

It is easy to check that if $g \leq h$, then $F_{g,\alpha} \leq F_{h,\alpha}$ for all α. Because we assumed h to be strictly increasing, this entails $F_\alpha \leq F_{h,\alpha}$, and we have the inclusion $\mathbf{F}_\alpha \subseteq \mathbf{F}_{h,\alpha}$ for all strictly increasing h.

The converse inclusion does not hold, since, i.e., $F_{h,1}$ is nonelementary for $h(x) = 2^x$. However, observe that in this instance, $h \leq F_2$, and we can see that $F_{F_2,k} = F_{2+k}$ for all k in $\mathbb{N}$. This entails that $\mathbf{F}_{h,1} \subseteq \mathbf{F}_3$ for $h(x) = 2^x$. Thus, when working with relativized classes, one should somehow "offset" the ordinal index by an appropriate amount.

There is nevertheless a difficulty with relativized functions: it is rather straightforward to show that $F_{h,\alpha} \leq F_{\beta+\alpha}$ if $h \leq F_\beta$, *assuming* that the direct sum $\beta + \alpha$ does not "discard" any summand from the CNF of β, e.g., $F_{F_1,k} = F_{k+1}$ and $F_{F_{\omega,\omega}} = F_{\omega \cdot 2}$. However, observe that $F_{F_{1,\omega}}(x) = F_{F_{1,x+1}}(x) = F_{x+2}(x) > F_{x+1}(x) = F_\omega(x)$. Thanks to the closure of $\mathbf{F}_\alpha$ under reductions in $\mathscr{F}_{<\alpha}$, this issue can be solved by composing with an appropriate function, e.g., $F_{F_{1,\omega}}(x) \leq F_\omega(x+1)$. This idea is formalized in Section A.4 and allows to show the following theorem.

THEOREM 4.2. *Let $h: \mathbb{N} \to \mathbb{N}$ be a strictly increasing function and α, β be two ordinals.*

(i) *If $h \in \mathscr{F}_\beta$, then $\mathbf{F}_{h,\alpha} \subseteq \mathbf{F}_{\beta+1+\alpha}$.*
(ii) *If $h \leq F_\beta$, then $\mathbf{F}_{h,\alpha} \subseteq \mathbf{F}_{\beta+\alpha}$.*

PROOF. For (i), if h is in $\mathscr{F}_\beta$, then there exists x_h in $\mathbb{N}$ such that for all $x \geq x_h$, $h(x) \leq F_{\beta+1}(x)$ [Löb and Wainer 1970, Lemma 2.7]. By Lemma A.5, this entails that for all $x \geq x_h$, $F_{h,\alpha}(x) \leq F_{\beta+1+\alpha}(F_\gamma(x))$ for some $\gamma < \beta + 1 + \alpha$. Define the function f_h by $f_h(x) \stackrel{\text{def}}{=} x + x_h$; then for all x, $F_{h,\alpha}(x) \leq F_{h,\alpha}(f_h(x)) \leq F_{\beta+1+\alpha}(F_\gamma(f_h(x)))$. Observe that $F_\gamma \circ f_h$ is in $\mathscr{F}_{<\beta+1+\alpha}$, and thus $\mathbf{F}_{h,\alpha} \subseteq \mathbf{F}_{\beta+1+\alpha}$.

For (ii), if $\beta + \alpha = 0$, then $\beta = \alpha = 0$, and thus $h(x) = x + 1$ since it has to be strictly increasing, and $F_{h,0} = F_0$. Otherwise, Lemma A.5 shows that $F_{h,\alpha} \leq F_{\beta+\alpha} \circ F_\gamma$ for some $\gamma < \beta + \alpha$. Observe that F_γ is in $\mathscr{F}_{<\beta+\alpha}$, and thus $\mathbf{F}_{h,\alpha} \subseteq \mathbf{F}_{\beta+\alpha}$. □

The statement of Theorem 4.2 is somewhat technical but easy to apply to concrete situations, i.e., note the following corollary.

Corollary 4.3. *Let $h\colon \mathbb{N} \to \mathbb{N}$ be a strictly increasing primitive recursive function and $\alpha \geq \omega$. Then $\mathbf{F}_{h,\alpha} = \mathbf{F}_\alpha$.*

Proof. The function h is in $\mathscr{F}_k$ for some $k < \omega$, and thus $\mathbf{F}_{h,\alpha} \subseteq \mathbf{F}_{k+1+\alpha} = \mathbf{F}_\alpha$ by Theorem 4.2. Conversely, since h is strictly increasing, $\mathbf{F}_\alpha \subseteq \mathbf{F}_{h,\alpha}$. □

4.1.3. Fundamental Sequences. Our last example of a minor variation is to change the assignment of fundamental sequences. Instead of the standard assignment of (1), we posit a monotone function $s\colon \mathbb{N} \to \mathbb{N}$ and consider the assignment

$$(\gamma + \omega^{\beta+1})(x)_s \stackrel{\text{def}}{=} \gamma + \omega^\beta \cdot s(x), \qquad (\gamma + \omega^\lambda)(x)_s \stackrel{\text{def}}{=} \gamma + \omega^{\lambda(x)_s}. \tag{19}$$

Thus, the standard assignment in (1) is obtained as the particular case $s(x) = x + 1$. As previously, this gives rise to new fast-growing functions

$$F_{0,s}(x) \stackrel{\text{def}}{=} x + 1, \quad F_{\alpha+1,s}(x) \stackrel{\text{def}}{=} F^{s(x)}_{\alpha,s}(x), \quad F_{\lambda,s}(x) \stackrel{\text{def}}{=} F_{\lambda(x)_s,s}(x) \tag{20}$$

and complexity classes

$$\mathbf{F}_{\alpha,s} \stackrel{\text{def}}{=} \bigcup_{p \in \mathscr{F}_{<\alpha}} \mathrm{DTime}(F_{\alpha,s}(p(n))). \tag{21}$$

We obtain similar results with nonstandard fundamental sequences as with relativized hierarchies (thus also yielding a statement similar to that of Corollary 4.3).

Theorem 4.4. *Let $s\colon \mathbb{N} \to \mathbb{N}$ be a strictly increasing function and α, β be two ordinals.*

(i) *If $s \in \mathscr{F}_\beta$, then $\mathbf{F}_{\alpha,s} \subseteq \mathbf{F}_{\beta+1+\alpha}$.*
(ii) *If $s \leq F_\beta$, then $\mathbf{F}_{\alpha,s} \subseteq \mathbf{F}_{\beta+\alpha}$.*

Proof. By applying Theorem 4.2 alongside Lemma A.6. □

The case where s is the identity function $\mathrm{id}(x) \stackrel{\text{def}}{=} x$ is fairly common in the literature; we obtain in this particular case the following corollary.

Corollary 4.5. *For all α, $\mathbf{F}_{\alpha,\mathrm{id}} = \mathbf{F}_\alpha$.*

Proof. By Theorem 4.4 and since $\mathrm{id} \leq F_0$, we have the inclusion $\mathbf{F}_{\alpha,id} \subseteq \mathbf{F}_\alpha$. The converse inclusion stems from $F_\alpha \leq F_{\alpha,\mathrm{id}} \circ F_0$, as can be seen by transfinite induction over α (see Lemma A.7). □

4.2. Computational Models and Reductions

In order to be used together with reductions in $\mathscr{F}_{<\alpha}$, the classes $\mathbf{F}_\alpha$ need to be closed under such functions. The main technical lemma to this end states the following.

Lemma 4.6. *Let f and f' be two functions in $\mathscr{F}_{<\alpha}$. Then there exists p in $\mathscr{F}_{<\alpha}$ such that $f \circ F_\alpha \circ f' \leq F_\alpha \circ p$.*

Proof. By Corollary A.9, we know that there exists g in $\mathscr{F}_{<\alpha}$ such that $f \circ F_\alpha \leq F_\alpha \circ g$. We can thus define $p \stackrel{\text{def}}{=} g \circ f'$, which is also in $\mathscr{F}_{<\alpha}$ since the latter is closed under composition, to obtain the statement. □

4.2.1. Computational Models. Note that because we assume that $\alpha \geq 3$, $\mathscr{F}_{<\alpha}$ contains all of the elementary functions, and thus Lemma 4.6 also entails the robustness of the $\mathbf{F}_\alpha$ classes under changes in the model of computation—e.g., RAM versus Turing machines versus Minsky machines, deterministic or nondeterministic or alternating—or the type

of resources under consideration—time or space, e.g.,

$$\mathbf{F}_\alpha = \bigcup_{p\in\mathscr{F}_{<\alpha}} \text{NTime}(F_\alpha(p(n))) = \bigcup_{p\in\mathscr{F}_{<\alpha}} \text{Space}(F_\alpha(p(n))). \tag{22}$$

4.2.2. Many-One Reductions. For a function $f:\mathbb{N}\to\mathbb{N}$ and two languages A and B, we say that A *many-one* reduces to B in time $f(n)$, written $A \leq_m^f B$, if there exists a Turing transducer T working in deterministic time $f(n)$ such that for all x, x is in A if and only if $T(x)$ is in B. For a class of functions $\mathcal{C}$, we write $A \leq_m^{\mathcal{C}} B$ if there exists f in $\mathcal{C}$ such that $A \leq_m^f B$. As could be expected given the definitions, each class $\mathbf{F}_\alpha$ is closed under many-one $\mathscr{F}_{<\alpha}$ reductions.

Theorem 4.7. *Let A and B be two languages. If $A \leq_m^{\mathscr{F}_{<\alpha}} B$ and $B \in \mathbf{F}_\alpha$, then $A \in \mathbf{F}_\alpha$.*

Proof. By definition, $A \leq_m^{\mathscr{F}_{<\alpha}} B$ means that there exists a Turing transducer T working in deterministic time $f(n)$ for some f in $\mathscr{F}_{<\alpha}$; note that this implies that the function implemented by T is also in $\mathscr{F}_{<\alpha}$ by (6). Furthermore, $B \in \mathbf{F}_\alpha$ entails the existence of a Turing machine M that accepts x if and only if x is in B and works in deterministic time $F_\alpha(p(n))$ for some p in $\mathscr{F}_{<\alpha}$. We construct $T(M)$, a Turing machine that, given an input x, first computes $T(x)$ by simulating T and then simulates M on $T(x)$ to decide acceptance; $T(M)$ works in deterministic time $f(n)+F_\alpha(p(T(n)))$, which shows that A is in $\mathbf{F}_\alpha$ by Lemma 4.6. □

4.2.3. Turing Reductions. We write similarly that $A \leq_T^f B$ if there exists a Turing machine for A working in deterministic time $f(n)$ with oracle calls to B, and $A \leq_T^{\mathcal{C}} B$ if there exists f in $\mathcal{C}$ such that $A \leq_T^f B$. It turns out that Turing reductions in $\mathscr{F}_{<\alpha}$ can be used instead of many-one reductions.

Theorem 4.8. *Let $\alpha \geq 3$ and A and B be two languages. If $A \leq_T^{\mathscr{F}_{<\alpha}} B$ and $B \in \mathbf{F}_\alpha$, then $A \in \mathbf{F}_\alpha$.*

Proof. It is a folklore result on queries in recursion theory that if $A \leq_T^f B$, then $A \leq_m^{2^f} B^{\text{tt}}$ where $2^f(n) \stackrel{\text{def}}{=} 2^{f(n)}$ and B^{tt} is the *truth table* version of the language B, which evaluates a Boolean combination of queries "$x \in B$." Indeed, we can easily simulate the oracle machine for A using a nondeterministic Turing transducer also in time $f(n)$ that guesses the answers of the B oracle and writes a conjunction of checks "$x \in B$" or "$x \notin B$" on the output, to be evaluated by a B^{tt} machine. This transducer can be determinized by exploring both outcomes of the oracle calls and handling them through disjunctions in the output; it now works in time $2^f(n)$.

Since $\alpha \geq 3$ and f is in $\mathscr{F}_{<\alpha}$, 2^f is also in $\mathscr{F}_{<\alpha}$. Furthermore, since B is in $\mathbf{F}_\alpha$, B^{tt} is also in $\mathbf{F}_\alpha$. The statement then holds by Theorem 4.7. □

5. STRICTNESS

The purpose of this section is to establish the strictness of the $(\mathbf{F}_\alpha)_\alpha$ hierarchy (Section 5.2). As a first step, we prove that the F_α functions are "elementarily" constructible (Section 5.1), which is of independent interest for combinatorial algorithms, in line with Section 3.2.3. We end this section with a remark on the case $\alpha = 2$ (Section 5.3).

5.1. Elementary Constructivity

The functions F_α are known to be *honest*, i.e., to be computable in time $\mathscr{F}_\alpha$ [Wainer 1970; Fairtlough and Wainer 1998]. However, this is not tight enough for their use in length function theorems, as in Section 3.2.3, where we want to compute their value in

time elementary in F_α itself. Formally, we call a function f *elementarily constructible* if there exists an elementary function e in $\mathrm{FEL}_{\mathrm{EM}} = \mathscr{F}^*_{<3}$ such that $f(n)$ can be computed in time $e(f(n))$ for all n.

We present the statement in the more general case of relativized fast-growing functions, defined in (14) and discussed in Section 4.1.2; since $F_0(x) = x+1$ is elementarily constructible, this yields the result that all F_α functions are elementarily constructible.

THEOREM 5.1. *Let $h\colon \mathbb{N} \to \mathbb{N}$ be an elementarily constructible strictly increasing function and α be an ordinal, then $F_{h,\alpha}$ is also elementarily constructible.*

PROOF. Assume that $h(n)$ can be computed in time $e(h(n))$ for some fixed elementary monotone function e. Proposition A.12 shows that $F_{h,\alpha}$ can be computed in time $O(f(F_{h,\alpha}(n)))$ for the elementary function $f(x) \stackrel{\text{def}}{=} x \cdot (p \circ G_{\omega^\alpha}(x)) + e(x))$, where $p \circ G_{\omega^\alpha}$ is an elementary function that takes the cost of manipulating (an encoding of) the ordinal indices into account. Lemma 4.6 then yields the result. □

5.2. Strictness

Let us introduce yet another generalization of the $(\mathbf{F}_\alpha)_\alpha$ classes, which will allow for a characterization of the $(\mathscr{F}^*_\alpha)_\alpha$ and $(\mathscr{F}^*_{<\alpha})_\alpha$ classes. For an ordinal α and a finite $c > 0$, define

$$\mathbf{F}^c_\alpha \stackrel{\text{def}}{=} \bigcup_{p \in \mathscr{F}_{<\alpha}} \mathrm{DTime}\big(F^c_\alpha(p(n))\big). \tag{23}$$

Thus, $\mathbf{F}_\alpha$ as defined in (9) corresponds to the case $c = 1$.

PROPOSITION 5.2. *For all $\alpha \geq 2$,*

$$\mathscr{F}^*_\alpha = \bigcup_c \mathbf{F}^c_\alpha.$$

PROOF. The left-to-right inclusion is immediate by definition of $\mathscr{F}^*_\alpha$ in (8). The converse inclusion stems from the fact that if p is in $\mathscr{F}_\beta$ for some $\beta < \alpha$, then there exists d such that $p \leq F^d_\alpha$ [Löb and Wainer 1970, Theorem 2.10], and hence $F^c_\alpha \circ p \leq F^{c+d}_\alpha$ by monotonicity of F_α. □

Let us prove the strictness of the $(\mathbf{F}^c_\alpha)_{c,\alpha}$ hierarchy. By Proposition 5.2, it will also prove that of $(\mathscr{F}^*_\alpha)_\alpha$ along the way (note that it is not implied by the strictness of $(\mathscr{F}_\alpha)_\alpha$, as it would be conceivable that none of the separating examples would be $\{0,1\}$-valued).

THEOREM 5.3 (STRICTNESS). *For all $c > 0$ and $2 \leq \beta < \alpha$,*

$$\mathbf{F}^c_\beta \subsetneq \mathbf{F}^{c+1}_\beta \subsetneq \mathbf{F}_\alpha.$$

PROOF OF $\mathbf{F}^{c+1}_\beta \subsetneq \mathbf{F}_\alpha$. Consider first a language L in $\mathbf{F}^{c+1}_\beta$, accepted by a Turing machine working in time $F^{c+1}_\beta \circ p$ for some p in $\mathscr{F}_{<\beta}$ that we can assume to be monotone. Since $\beta < \alpha$ and $F^{c+1}_\beta \circ p$ is in $\mathscr{F}_\beta$, there exists n_0 such that for all $n \geq n_0$, $F^{c+1}_\beta(p(n)) \leq F_\alpha(n)$, and hence for all n, $F^{c+1}_\beta(p(n)) \leq F^{c+1}_\beta(p(n+n_0)) \leq F_\alpha(n+n_0)$ by monotonicity and expansivity of F_β. Observe that the function $n \mapsto n_0 + n$ is in $\mathscr{F}_0 \subseteq \mathscr{F}_{<\alpha}$, and thus L also belongs to $\mathbf{F}_\alpha$.

The strictness of the inclusion can be shown by a straightforward diagonalization argument. Define for this the language

$$L_\alpha \stackrel{\text{def}}{=} \{\langle M\rangle \# x \mid M \text{ accepts } x \text{ in } F_\alpha(|x|) \text{ steps}\}, \tag{24}$$

where $\langle M \rangle$ denotes a description of the Turing machine M and # is a separator. Then, by Theorem 5.1, L_α belongs to $\mathbf{F}_\alpha$, thanks to a Turing machine that first computes F_α in time $F_\alpha \circ e$ for some elementary function e, and then simulates M in time elementary in $F_\alpha \circ e$. Assume now for the sake of contradiction that L_α belongs to $\mathbf{F}_\beta^{c+1}$, i.e., that there exists some c and some Turing machine K that accepts L_α in time F_β^{c+1}. Again, since $\beta < \alpha$ and $F_\beta^{c+1} \circ F_1$ is in $\mathscr{F}_\beta$, there exists n_0 such that for all $n \geq n_0$, $F_\beta^{c+1}(2n+1) \leq F_\alpha(n)$. We exhibit a new Turing machine N,

(1) which takes as input the description $\langle M \rangle$ of a Turing machine and simulates K on $\langle M \rangle \# \langle M \rangle$ but accepts if and only if K rejects, and
(2) we ensure that a description $\langle N \rangle$ of N has size $n \geq n_0$.

Feeding this description $\langle N \rangle$ to N, it runs in time $F_\beta^{c+1}(2n+1) \leq F_\alpha(n)$, and we obtain a contradiction whether it accepts or not:

—if N accepts, then K rejects $\langle N \rangle \# \langle N \rangle$, which is therefore not in L_α, and thus N does not accept $\langle N \rangle$ in at most $F_\alpha(n)$ steps, which is absurd;
—if N rejects, then K accepts $\langle N \rangle \# \langle N \rangle$, which is therefore in L_α, and thus N accepts $\langle N \rangle$ in at most $F_\alpha(n)$ steps, which is absurd. □

Proof of $\mathbf{F}_\beta^c \subsetneq \mathbf{F}_\beta^{c+1}$. Similar to the previous proof; picking F_β^{c+1} as the time bound instead of F_α in (24) suffices to establish strictness. □

By Proposition 5.2, a first consequence of Theorem 5.3 is that

$$\mathscr{F}_\beta^* \subsetneq \mathbf{F}_\alpha \tag{25}$$

for all $2 \leq \beta < \alpha$. Another consequence is that $(\mathbf{F}_\alpha)_\alpha$ "catches up" with $(\mathscr{F}_\alpha^*)_\alpha$ at every limit ordinal.

Corollary 5.4. *Let λ be a limit ordinal, then*

$$\mathscr{F}_{<\lambda}^* = \bigcup_{\beta<\lambda} \mathbf{F}_\beta \subsetneq \mathbf{F}_\lambda.$$

Proof. The equality $\mathscr{F}_{<\lambda}^* = \bigcup_{\beta<\lambda} \mathbf{F}_\beta$ and the inclusion $\mathscr{F}_{<\lambda}^* \subseteq \mathbf{F}_\lambda$ can be checked by considering a problem in some $\mathscr{F}_\beta^*$ for $\beta < \lambda$: it is in $\mathbf{F}_\beta^c$ for some $c > 0$ by Proposition 5.2, and hence in $\mathbf{F}_{\beta+1}$ with $\beta + 1 < \lambda$ by Theorem 5.3, and therefore in $\mathbf{F}_\lambda$ again by Theorem 5.3. Regarding the strictness of the inclusion, assume for the sake of contradiction that $\mathbf{F}_\lambda \subseteq \bigcup_{\beta<\lambda} \mathbf{F}_\beta$: this would entail $\mathbf{F}_\lambda \subseteq \mathbf{F}_\beta$ for some $\beta < \lambda$, violating Theorem 5.3. □

Corollary 5.4 yields another characterization of the primitive-recursive and multiply-recursive problems as

$$\mathrm{PR} = \bigcup_k \mathbf{F}_k, \qquad \mathrm{MR} = \bigcup_k \mathbf{F}_{\omega^k}. \tag{26}$$

Note that strictness implies that there are no "$\mathscr{F}_\alpha^*$-complete" problems under $\mathscr{F}_{<\alpha}$ reductions, since by Proposition 5.2 such a problem would necessarily belong to some $\mathbf{F}_\alpha^c$ level, which would in turn entail the collapse of the $(\mathbf{F}_\alpha^c)_c$ hierarchy at the $\mathbf{F}_\alpha^c$ level and contradict Theorem 5.3.

Similarly, fix a limit ordinal λ and some reduction class $\mathscr{F}_\alpha$ for some $\alpha < \lambda$: there cannot be any meaningful "$\mathscr{F}_{<\lambda}^*$-complete" problem under $\mathscr{F}_\alpha$ reductions, as such a problem would be in $\mathscr{F}_\beta^*$ for some $\alpha < \beta < \lambda$, hence contradicting the strictness of the $(\mathscr{F}_\beta^*)_{\beta<\alpha}$ hierarchy; in particular, there are no "PR-complete" nor "MR-complete" problems.

5.3. The Case $\alpha = 2$

This case is a bit particular. We did not consider it in the rest of the article (nor the other cases for $\alpha < 2$) because it does not share the usual characteristics of the $(\mathbf{F}_\alpha)_\alpha$: i.e., the model of computation and the kind of resources become important, as

$$\mathbf{F}_2 \stackrel{\text{def}}{=} \bigcup_{p \in \mathscr{F}_1} \text{DTime}(F_2(p(n))) \tag{27}$$

would a priori be different if we were to define it through NTime or DSpace computations; the following results are artifacts of this particular choice of a definition.

5.3.1. Recursion Schemes. To define $\mathbf{F}_2$ fully, we need the original definition of the extended Grzegorczyk hierarchy $(\mathscr{F}_\alpha)_\alpha$ by Löb and Wainer [1970]—the characterization in (5) is only correct for $\alpha \geq 2$. This definition is based on the closure of a set of initial functions under the operations of *substitution* and *limited primitive recursion*. More precisely, the set of initial functions at level α comprises the constant *zero function* 0, the *sum function* $+\colon x_1, x_2 \mapsto x_1 + x_2$, the *projections* $\pi_i^n\colon x_1, \dots, x_n \mapsto x_i$ for all $0 < i \leq n$, and the fast-growing function F_α. New functions are added to form the class $\mathscr{F}_\alpha$ through two operations:

—*substitution* if $h_0, h_1, \dots, h_p$ belong to the class, then so does f if

$$f(x_1, \dots, x_n) = h_0(h_1(x_1, \dots, x_n), \dots, h_p(x_1, \dots, x_n));$$

—*limited primitive recursion* if h_0, h_1, and g belong to the class, then so does f if

$$\begin{aligned} f(0, x_1, \dots, x_n) &= h_0(x_1, \dots, x_n), \\ f(y+1, x_1, \dots, x_n) &= h_1(y, x_1, \dots, x_n, f(y, x_1, \dots, x_n)), \\ f(y, x_1, \dots, x_n) &\leq g(\max\{y, x_1, \dots, x_n\}). \end{aligned}$$

Observe that primitive recursion is defined by ignoring the last *limitedness* condition in the previous definition. See the survey by Clote [1999] on the relationships between machine-defined and recursion-defined complexity classes.

5.3.2. Linear Exponential Time. Let us focus for now on $\mathscr{F}_1$, which is the class of reductions used in $\mathbf{F}_2$. First note that the successor function $\text{succ}(x) = x + 1 = x + F_1(0)$ belongs to $\mathscr{F}_1$.

Call a function f *linear* if there exists a constant c such that $f(x_1, \dots, x_n) \leq c \cdot \max_i x_i$ for all $x_1, \dots, x_n$. Observe that for all c, the function $f_c(x) \stackrel{\text{def}}{=} c \cdot x$ is in $\mathscr{F}_1$ since $f_c(0) = 0$, $f_c(x+1) = \text{succ}^c(0) + f_c(x)$, and $f_c(x) \leq F_1^c(x)$; thus, any linear function is bounded above by a function in $\mathscr{F}_1$. Conversely, if f is in $\mathscr{F}_1$, then it is linear: this is true of the initial functions and preserved by the two operations of substitution and limited primitive recursion.[3]

This entails that $\mathbf{F}_2$ matches a well-known complexity class, since furthermore $F_2(n) = 2^{n+1+\log(n+1)} - 1$ is in $2^{O(n)}$: $\mathbf{F}_2$ is the *weak* (aka *linear*) exponential-time complexity class:

$$\mathbf{F}_2 = \text{E} \stackrel{\text{def}}{=} \text{DTime}\big(2^{O(n)}\big). \tag{28}$$

6. A SHORT CATALOGUE

Our introduction of the fast-growing complexity classes is motivated by *already known* decidability problems, arising for instance in logic, verification, or database theory, for

[3]Thus, $\mathscr{F}_1 \subsetneq \mathscr{E}^2$: the latter additionally contains the function $x, y \mapsto (x+1)\cdot(y+1)$ as an initial function and is equal to FLinSpace [Ritchie 1963; Clote 1999, Theorem 3.36].

which no precise classification could be provided in the existing hierarchies. By listing some of these problems, we hope to initiate the exploration of this mostly uncharted area of complexity and to foster the use of reductions from known problems rather than proofs from Turing machines. The following catalogue of complete problems does not attempt to be exhaustive—i.e., Friedman [1999] presents many problems "of enormous complexity."

Because examples for TOWER are well known and abound in the literature, starting with a 1975 survey by Meyer [1975a],[4] we rather focus on the nonprimitive-recursive levels, i.e., the $\mathbf{F}_\alpha$ for $\alpha \geq \omega$. Interestingly, for their upper bound, all of these examples rely on the existence of some WQO (of *maximal order type* ω^α; see de Jongh and Parikh [1977]) and on a matching length function theorem.

6.1. F_ω-Complete Problems

Here we gather some of the decision problems known to be ACK-complete at the time of this writing. The common trait of all of these problems is their reliance on Dickson's lemma over $\mathbb{N}^d$ for some d for decidability, and on the associated length function theorems [McAloon 1984; Clote 1986; Figueira et al. 2011; Abriola et al. 2015] for ACK upper bounds.

6.1.1. Vector Addition Systems. Vector addition systems (VAS, and equivalently Petri nets) provided the first known Ackermannian decision problem: FCP.

A d-dimensional VAS is a pair $\langle \bar{\mathsf{v}}_0, \bar{\mathsf{A}} \rangle$, where $\bar{\mathsf{v}}_0$ is an initial configuration in $\mathbb{N}^d$ and $\bar{\mathsf{A}}$ is a finite set of transitions in $\mathbb{Z}^d$. A transition $\bar{\mathsf{u}}$ in $\bar{\mathsf{A}}$ can be applied to a configuration $\bar{\mathsf{v}}$ in $\mathbb{N}^d$ if $\bar{\mathsf{v}}' = \bar{\mathsf{v}} + \bar{\mathsf{u}}$ is in $\mathbb{N}^d$; the resulting configuration is then $\bar{\mathsf{v}}'$. The complexity of decision problems for VAS usually varies from EXPSPACE-complete [Lipton 1976; Rackoff 1978; Blockelet and Schmitz 2011] to $\mathbf{F}_\omega$-complete [Mayr and Meyer 1981; Jančar 2001] to undecidable [Hack 1976; Jančar 1995], via a key problem, whose exact complexity is unknown: VAS Reachability [Mayr 1981; Kosaraju 1982; Lambert 1992; Leroux 2011; Leroux and Schmitz 2015].

Finite containment problem (FCP).

Instance: Two VAS $\mathcal{V}_1$ and $\mathcal{V}_2$ known to have finite sets Reach($\mathcal{V}_1$) and Reach($\mathcal{V}_2$) of reachable configurations.
Question: Is Reach($\mathcal{V}_1$) included in Reach($\mathcal{V}_2$)?
Lower bound: Mayr and Meyer [1981], from an F_ω-bounded version of Hilbert's Tenth Problem. A simpler reduction is given by Jančar [2001] from F_ω-MM, the halting problem of F_ω-bounded Minsky machines.
Upper bound: Originally McAloon [1984] and Clote [1986], or more generally using length function theorems for Dickson's lemma [Figueira et al. 2011; Abriola et al. 2015].
Comment: Testing whether the set of reachable configurations of a VAS is finite is EXPSPACE-complete [Lipton 1976; Rackoff 1978]. FCP has been generalized by Jančar [2001] to a large range of behavioral relations between two VAS. Without the finiteness condition, these questions are undecidable [Hack 1976; Jančar 1995, 2001].

An arguably simpler problem on VAS has recently been shown to be ACK-complete by Hofman and Totzke [2014]. A *labeled vector addition system with states* (VASS) $\mathcal{V} = \langle Q, \Sigma, d, T, q_0, \bar{\mathsf{v}}_0 \rangle$ is a VAS extended with a finite set Q of control states that includes a distinguished initial state q_0. The transitions in T of such systems are

[4]Of course, Meyer does not explicitly state TOWER-completeness, but it follows immediately from the lower and upper bounds that he provides.

furthermore labeled with symbols from a finite alphabet Σ: transitions are then defined as quadruples $q \xrightarrow{a,\bar{u}} q'$ for a in Σ and $\bar{u}$ in $\mathbb{Z}^d$. Such a system defines an infinite labeled transition system $\langle Q \times \mathbb{N}^d, \to, (q_0, \bar{v}_0)\rangle$, where $(q, \bar{v}) \xrightarrow{a} (q', \bar{v} + \bar{u})$ if $q \xrightarrow{a,\bar{u}} q'$ is in T and $\bar{v} + \bar{u} \geq \bar{0}$. The *set of traces* of $\mathcal{V}$ is the set of finite sequences $L(\mathcal{V}) \stackrel{\text{def}}{=} \{a_1 \cdots a_n \in \Sigma^* \mid \exists (q, \bar{v}) \in Q \times \mathbb{N}^d . (q_0, \bar{v}_0) \xrightarrow{a_1 \cdots a_n} (q, \bar{v})\}$.

One-dimensional VASS universality (1VASSU).

Instance: A one-dimensional labeled VASS $\mathcal{V} = \langle Q, \Sigma, 1, T, q_0, \bar{x}_0\rangle$.
Question: Does $L(\mathcal{V}) = \Sigma^*$, i.e., is every finite sequence over Σ a trace of $\mathcal{V}$?
Lower bound: Hofman and Totzke [2014], by reduction from reachability in gainy counter machines (see LCM).
Upper bound: Hofman and Totzke [2014], using length function theorems for Dickson's lemma.
Comment: One-dimensional VASS are also called *one counter nets* in the literature. More generally, the *inclusion* problem $L \subseteq L(\mathcal{V})$ for some rational language L is still Ack-complete.

6.1.2. Unreliable Counter Machines. An LCM is syntactically a Minsky machine, but its operational semantics are different: its counter values can decrease nondeterministically at any moment during execution. See Section 3.2 for details.

Lossy counter machines reachability (LCM).

Instance: An LCM M and a configuration σ.
Question: Is σ reachable in M with lossy semantics?
Lower bound: Schnoebelen [2010], by a direct reduction from F_ω-bounded Minsky machines. The first proofs were given independently by Urquhart [1999] and Schnoebelen [2002].
Upper bound: Length function theorem for Dickson's lemma.
Comment: Completeness also holds for terminating LCMs—meaning that every computation starting from the initial configuration terminates—for coverability in Reset or Transfer Petri nets, and for reachability in *gainy* counter machines, where counter values can increase nondeterministically.

6.1.3. Relevance Logics. Relevance logics provide different semantics of implication, where a fact B is said to follow from A, written "$A \to B$," only if A is actually *relevant* in the deduction of B. For instance, this excludes $A \to (B \to A)$, $(A \wedge \neg A) \to B$, and so forth—see Dunn and Restall [2002] for more details. Although the full logic $\mathbf{R}$ is undecidable [Urquhart 1984], its conjunctive-implicative fragment $\mathbf{R}_{\to,\wedge}$ is decidable and Ack-complete.

Conjunctive relevant implication (CRI).

Instance: A formula A of $\mathbf{R}_{\to,\wedge}$.
Question: Is A a theorem of $\mathbf{R}_{\to,\wedge}$?
Lower bound: Urquhart [1999], from a variant of LCM: the emptiness problem of *alternating expansive counter systems*, for which he proved $\mathbf{F}_\omega$-hardness directly from F_ω-MM the halting problem in F_ω-bounded Minsky machines.
Upper bound: Urquhart [1999], using length function theorem for Dickson's lemma.
Comment: Hardness also holds for any intermediate logic between $\mathbf{R}_{\to,\wedge}$ and $\mathbf{T}_{\to,\wedge}$, which might include some undecidable fragments. The related *contractive propositional linear logic* LLC and its additive-multiplicative fragment MALLC are also Ack-complete [Lazić and Schmitz 2015].

6.1.4. Data Logics and Register Automata. Data logics and register automata are concerned with structures like words or trees with an additional equivalence relation over the positions. The motivation for this stems in particular from XML processing, where the equivalence stands for elements sharing the same *datum* from some infinite data domain $\mathbb{D}$. Enormous complexities often arise in this context, both for automata models (register automata and their variants, when extended with alternation or histories) and for logics (which include logics with *freeze* operators and XPath fragments)—the two views being tightly interconnected.

Emptiness of alternating 1-register automata (A1RA).

Instance: An A1RA $\mathcal{A}$.
Question: Is the data language $L(\mathcal{A})$ empty?
Lower bound: Demri and Lazić [2009], from reachability in gainy counter machines LCM.
Upper bound: Demri and Lazić [2009], by reducing to reachability in gainy counter machines LCM.
Comment: There exist many variants of the A1RA model, and hardness also holds for the corresponding data logics (e.g., Jurdzinski and Lazic [2011], Demri and Lazic [2009], Figueira and Segoufin [2009], Tan [2010], Figueira [2012] and Tzevelekos and Grigore [2013]). See A1TA for the case of linearly ordered data and $\mathsf{LTL}^{\downarrow}_{[k]}$ for data logics using multiple attributes with a hierarchical policy.

6.1.5. Metric Temporal Logic. Metric temporal logic (MTL) allows reasoning on *timed words* over $\Sigma \times \mathbb{R}$, where Σ is a finite alphabet and the real values are nondecreasing *timestamps* on events [Koymans 1990]. When considering infinite timed words, one usually focuses on *non-Zeno* words, where the timestamps are increasing and unbounded. MTL is an extension of linear temporal logic, where temporal modalities are decorated with real intervals constraining satisfaction; for instance, a timed word w satisfies the formula $\mathsf{F}_{[3,\infty)}\varphi$ at position i, written $w, i \models \mathsf{F}_{[3,\infty)}\varphi$, only if φ holds at some position $j > i$ of w with timestamp $\tau_j - \tau_i \geq 3$. The *safety* fragment of MTL restricts the intervals decorating "until" modalities to be right bounded.

Satisfiability of safety metric temporal logic (SMTL).

Instance: A safety MTL formula φ.
Question: Does there exist an infinite non-Zeno timed word w subject to $w, 0 \models \varphi$?
Lower bound: Lazić et al. [2013], by a direct reduction from F_ω-bounded Turing machines.
Upper bound: Lazić et al. [2013], by resorting to length function theorems for Dickson's Lemma.
Comment: The complexity bounds are established through reductions to and from the *fair termination* problem for insertion channel systems, which Lazić et al. [2013] show to be Ack-complete (see LCST).

6.1.6. Ground Term Rewriting. A *ground term rewrite system with state* (sGTRS) maintains a finite ordered labeled tree along with a control state from some finite set. Although most questions about ground term rewrite systems are decidable [Dauchet and Tison 1990], the addition of a finite set of control states yields a Turing-powerful formalism. Formally, a sGTRS $\langle Q, \Sigma, R\rangle$ over a ranked alphabet Σ and a finite set of states Q is defined by a finite set of rules $R \subseteq (Q \times T(\Sigma))^2$ of the form $(q, t) \to (q', t')$ acting over pairs of states and trees, which rewrite a configuration $(q, C[t])$ into $(q', C[t'])$ in any context C.

Hague [2014] adds *age* labels in $\mathbb{N}$ to every node of the current tree. In the initial configuration, every tree node has age zero, and at each rewrite step $(q, C[t]) \to (q', C[t'])$, in the resulting configuration the nodes in t' have age zero, and the nodes in C see their age increment by one if $q \neq q'$ or remain with the same age as in $(q, C[t])$ if $q = q'$. A *senescent* sGTRS with *lifespan* k in $\mathbb{N}$ restricts rewrites to only occur in subtrees of age at most k, i.e., when matching $C[t]$ the age of the root of t is $\leq k$.

State reachability in senescent ground term rewrite systems (SGTRS).

Instance: A senescent sGTRS $\langle Q, \Sigma, R\rangle$ with lifespan k, two states q_0 and q_f in Q, and an initial tree t_0 in $T(\Sigma)$.
Question: Does there exist a tree t in $T(\Sigma)$ such that (q_f, t) is reachable from (q_0, t_0)?
Lower bound: Hague [2014], from coverability in reset Petri nets (see LCM).
Upper bound: Hague [2014], by reducing to coverability in reset Petri nets (see LCM).

6.1.7. Interval Temporal Logics. Interval temporal logics provide a formal framework for reasoning about temporal intervals. Halpern and Shoham [1991] define a logic with modalities expressing the basic relationships that can hold between two temporal intervals, $\langle B\rangle$ for "begun by," $\langle E\rangle$ for "ended by," and their inverses $\langle \bar{B}\rangle$ and $\langle \bar{E}\rangle$. This logic, and even small fragments of it, has an undecidable satisfiability problem, thus prompting the search for decidable restrictions and variants. Montanari et al. [2010] show that the logic with relations $A\bar{A}B\bar{B}$—where $\langle A\rangle$ expresses that the two intervals "meet," i.e., share an endpoint—has an $\mathbf{F}_\omega$-complete satisfiability problem over finite linear orders, as follows.

Finite linear satisfiability of $A\bar{A}B\bar{B}$ (ITL).

Instance: An $A\bar{A}B\bar{B}$ formula φ.
Question: Does there exist an interval structure $\mathcal{S}$ over some finite linear order and an interval I of $\mathcal{S}$ s.t. $\mathcal{S}, I \models \varphi$?
Lower bound: Montanari et al. [2010], from reachability in lossy counter systems (LCM).
Upper bound: Montanari et al. [2010], by reducing to reachability in lossy counter systems (LCM).
Comment: Hardness already holds for the fragments $\bar{A}B$ and $\bar{A}\bar{B}$ [Bresolin et al. 2012].

6.2. $\mathbf{F}_{\omega^\omega}$-Complete Problems

The following problems are known to be complete for HAck. In most cases, they have been proven decidable thanks to Higman's lemma over some finite alphabet, and the complexity upper bounds stem from the length function theorems of Weiermann [1994], Cichoń and Tahhan Bittar [1998], and Schmitz and Schnoebelen [2011].

6.2.1. Lossy Channel Systems. Lossy channel systems (LCS) are finite labeled transition systems $\langle Q, M, \delta, q_0\rangle$ where transitions in $\delta \subseteq Q \times \{?, !\} \times M \times Q$ read and write on an unbounded channel. This would lead to a Turing-complete model of computation, but the operational semantics of LCS are "lossy": the channel loses symbols in an uncontrolled manner. Formally, the configurations of an LCS are pairs (q, x), where q in Q holds the current state and x in M^* holds the current contents of the channel. A read $(q, ?m, q')$ in δ updates this configuration into (q, x') if there exists some x'' subject to $x' \leq_* x''$ and $mx'' \leq_* x$—where $\leq_*$ denotes subword embedding—whereas a write transition $(q, !m, q')$ updates it into (q', x') with $x' \leq_* xm$; the initial configuration is (q_0, ε), with empty initial channel contents.

Due to the unboundedness of the channel, there might be infinitely many configurations reachable through transitions. Nonetheless, many problems are decidable [Abdulla and Jonsson 1996; Cécé et al. 1996] using Higman's lemma and what would later become known as the theory of *well-structured transition systems* (WSTS) [Finkel 1987; Abdulla et al. 2000; Finkel and Schnoebelen 2001]. LCS are also the primary source of problems hard for $\mathbf{F}_{\omega^\omega}$.

LCS reachability (LCS).

Instance: An LCS and a configuration (q, x) in $Q \times M^*$.
Question: Is (q, x) reachable from the initial configuration?
Lower bound: Chambart and Schnoebelen [2008b], by a direct reduction from F_{ω^ω}-MM the halting problem in F_{ω^ω}-bounded Minsky machines.
Upper bound: Chambart and Schnoebelen [2008b], using the length function theorem of Cichoń and Tahhan Bittar [1998], or more generally using length function theorems for Higman's lemma [Weiermann 1994; Schmitz and Schnoebelen 2011].
Comment: Hardness holds already for the (semantically defined) class of terminating systems, and for reachability in *insertion channel systems*, where symbols are nondeterministically inserted in the channel at arbitrary positions instead of being lost. The bounds are refined and parameterized in function of the size of the alphabet M in Karandikar and Schmitz [2013].

There are many interesting applications of this question; let us mention one in particular: Atig et al. [2010] show how concurrent finite programs communicating through *weak* shared memory—i.e., prone to reorderings of read or writes, modeling the actual behavior of microprocessors, their instruction pipelines, and cache levels—have an $\mathbf{F}_{\omega^\omega}$-complete control-state reachability problem, through reductions to and from LCS.

LCS termination (LCST).

Instance: An LCS.
Question: Is every sequence of transitions from the initial configuration finite?
Lower bound: Chambart and Schnoebelen [2008b], by a reduction from terminating instances of LCS.
Upper bound: Length function theorems for Higman's lemma.
Comment: Unlike Reachability, Termination is sensitive to switching from lossy semantics to insertion semantics: it becomes NL-complete in general [Cécé et al. 1996], Tower-complete when the channel system is equipped with *channel tests* [Bouyer et al. 2012], and Ack-complete when one asks for *fair* nontermination, where the channel contents are read infinitely often [Lazić et al. 2013].

6.2.2. Embedding Problems. Embedding problems were introduced by Chambart and Schnoebelen [2007], motivated by decidability problems in various classes of channel systems mixing lossy and reliable channels. These problems are centered on the subword embedding relation $\leq_*$ and referred to as Post Embedding Problems. There is a wealth of variants and applications (e.g., see Chambart and Schnoebelen [2008a], Karandikar and Schnoebelen [2012], and Karandikar and Schmitz [2013]).

Here we give a slightly different viewpoint, taken from Barcelo et al. [2013] and Karandikar and Schmitz [2013], that uses regular relations (i.e., definable by synchronous finite transducers) and rational relations (i.e., definable by finite transducers).

Rational embedding problem (RatEP).

Instance: A rational relation R included in $\Sigma^* \times \Sigma^*$.
Question: Is $R \cap \leq_*$ nonempty?
Lower bound: Chambart and Schnoebelen [2007], from reachability in lossy channel systems (LCS).
Upper bound: Length function theorems for Higman's lemma.
Comment: Chambart and Schnoebelen [2007] refer to this problem as the Regular Post Embedding Problem, not to be mistaken with GEP. An equivalent presentation uses a rational language L included in Σ^* and two homomorphisms $u, v\colon \Sigma^* \to \Sigma^*$, and asks whether there exists w in L subject to $u(w) \leq_* v(w)$. The bounds are refined and parameterized in function of the size of the alphabet Σ in Karandikar and Schmitz [2013].

Generalized embedding problem (GEP).

Instance: A regular relation R included in $(\Sigma^*)^m$ and a subset I of $\{1, \ldots, m\}^2$.
Question: Does there exist $(w_1, \ldots, w_m)$ in R subject to all (i, j) in I, $w_i \leq_* w_j$?
Lower bound: Barceló et al. [2013], from RatEP.
Upper bound: Length function theorems for Higman's lemma.
Comment: The Regular Embedding Problem (RegEP) corresponds to the case where $m = 2$ and $I = \{(1, 2)\}$, and is already $\mathbf{F}_{\omega^\omega}$-hard (see Karandikar and Schmitz [2013] for refined bounds). Barceló et al. [2013] use GEP to show the $\mathbf{F}_{\omega^\omega}$-hardness of querying graph databases using particular extended conjunctive regular path queries.

6.2.3. Timed Automata. Timed automata, invented by Alur and Dill [1994], are finite automata able to recognize timed words. They are extended with *clocks* that evolve synchronously through time, and can be reset and compared against some time interval by the transitions of the automaton. The model can be extended with alternation, which is then called an *ATA*. Satisfiability problems for MTL reduce to emptiness problems for ATAs. Using WSTS techniques, Ouaknine and Worrell [2007] and Lasota and Walukiewicz [2008] prove that in the case of a single clock, emptiness of ATAs is decidable. Note that the *safety* fragment of MTL has an Ack-complete satisfiability problem (see SMTL).

Emptiness of alternating 1-clock timed automata (A1TA).

Instance: An A1TA $\mathcal{A}$.
Question: Is the timed language $L(\mathcal{A})$ empty?
Lower bound: Lasota and Walukiewicz [2008], from reachability in insertion channel systems (LCS).
Upper bound: Length function theorems for Higman's lemma.
Comment: Hardness already holds for universality of nondeterministic 1-clock timed automata.

Finite satisfiability of metric temporal logic (fMTL).

Instance: An MTL formula φ.
Question: Does there exist a finite timed word w subject to $w, 0 \models \varphi$?
Lower bound: Ouaknine and Worrell [2007], from reachability in insertion channel systems (LCS).
Upper bound: Length function theorems for Higman's lemma.
Comment: Satisfiability for infinite timed words is undecidable [Ouaknine and Worrell 2006].

Note that recent work on data automata over linearly ordered domains has uncovered some strong ties with timed automata [Figueira et al. 2015; Figueira 2012].

6.2.4. Unordered Data Nets. Unordered data nets are a generalization of Petri nets where each token carries some datum from some infinite data domain, which can be tested for equality against the data of other tokens when firing the transitions of the system. This is a restriction over the more general *data nets* [Lazić et al. 2013], where the data domain is deemed to be densely linearly ordered (see ENC). Like general data nets, unordered data nets allow so-called whole-place operations, endowing them with generalized reset capabilities; the exact complexity of coverability for *unordered Petri data nets*, where such operations are not available, is unknown at the moment (TOWER-hardness is shown by Lazić et al. [2008]).

Unordered data nets coverability (UDN).

Instance: An unordered data net $\mathcal{N}$ and a place p of the net.
Question: Is there a reachable marking with a least one token in p?
Lower bound: Rosa-Velardo [2014], by a direct reduction from the halting problem in F_{ω^ω}-bounded Minsky machines.
Upper bound: Rosa-Velardo [2014], by proving a length function theorem for $\mathbb{M}_{\text{fin}}(\mathbb{N}^d)$ the set of finite multisets of vectors of naturals, ordered by multiset embedding.

This is the only instance in this list of a HACK-complete problem that does not explicitly rely on Higman's lemma.

6.3. $\mathbf{F}_{\omega^{\omega^\omega}}$-Complete Problems

Currently, the known $\mathbf{F}_{\omega^{\omega^\omega}}$-complete problems are all related to extensions of Petri nets called *enriched nets*, which include timed-arc Petri nets [Abdulla and Nylén 2001], ordered data nets and ordered Petri data nets [Lazić et al. 2008], and constrained multiset rewriting systems [Abdulla and Delzanno 2006]. Reductions between the different classes of enriched nets were shown by Abdulla et al. [2011] and Bonnet et al. [2010]. Defining these families of nets here would take too much space (see the referenced papers for details). These models share one characteristic: they define well-structured transition systems over finite sequences of vectors of natural numbers, which have an $\omega^{\omega^{\omega^\omega}}$ maximal order type.

Enriched net coverability (ENC).

Instance: An enriched net $\mathcal{N}$ and a place p of the net.
Question: Is there a reachable marking with a least one token in p?
Lower bound: Haddad et al. [2012], by a direct reduction from the halting problem in $F_{\omega^{\omega^\omega}}$-bounded Minsky machines.
Upper bound: Haddad et al. [2012], using length function theorems for finite sequences of vectors of natural numbers and Higman's lemma [Schmitz and Schnoebelen 2011].

6.4. $\mathbf{F}_{\varepsilon_0}$-Complete Problems

Problems complete for $\mathbf{F}_{\varepsilon_0}$ are untractable in a distinctive sense: although there exists a Turing machine able to answer on every instance, the termination proof of this Turing machine implies a totality proof for a function akin to F_{ε_0}: however, the latter is known to be independent of Peano arithmetic (e.g., Fartlough and Wainer [1998]).

6.4.1. Priority Channel Systems. Priority channel systems are defined similarly to LCS (compare to Section 6.2.1), but the message alphabet M is linearly ordered to

represent message *priorities*. Rather than message losses, the unreliable behaviors are now *message supersedings*, i.e., applications of the rewrite rules $ab \to b$ for $b \geq a$ in M on the channel contents.

PCS reachability (PCS).

Instance: A PCS and a configuration (q, x) in $Q \times M^*$.
Question: Is (q, x) reachable from the initial configuration?
Lower bound: Haase et al. [2014], by a direct reduction from the halting problem in F_{ε_0}-bounded Turing machines.
Upper bound: Haase et al. [2014], using length function theorems for nested applications of Higman's lemma [Schmitz and Schnoebelen 2011].

6.4.2. Nested Counter Systems and Hierarchical Multiattributed Data Logics. Finite data words may generally carry several data values from some infinite data domain in addition to a label from some finite alphabet. The satisfiability of data logics over such data words becomes undecidable, even for the restricted logics discussed in Section 6.1.4. However, decidability can be recovered when the logic is restricted by a hierarchical discipline on its attributes $\{0, \ldots, k\}$, where attribute i can only be tested for equality on two positions of the word if all attributes $0, \ldots, i-1$ are also simultaneously tested.

Satisfiability of freeze LTL with ordered attributes ($\mathsf{LTL}^{\downarrow}_{[k]}$).

Instance: A formula φ of freeze LTL with one register and k hierarchical attributes.
Question: Does there exist a k-attributed finite data word w subject to $w \models \varphi$?
Lower bound: Decker and Thoma [2015], by a direct reduction from F_{ε_0}-bounded Minsky machine.
Upper bound: Decker and Thoma [2015], by a reduction to reachability in priority channel systems (PCS).
Comment: The complexity bounds are established through the coverability problem for a class of nested counter systems [Decker and Thoma 2015].

7. CONCLUDING REMARKS

The classical complexity hierarchies are limited to elementary problems despite a growing number of natural problems that require much larger computational resources. In this article, we propose a definition for fast-growing complexity classes $(\mathbf{F}_\alpha)_\alpha$, which provide accurate enough notations for many nonelementary decision problems: they allow to express some important landmarks, like Tower $= \mathbf{F}_3$, Ack $= \mathbf{F}_\omega$, or HAck $= \mathbf{F}_{\omega^\omega}$, and are close enough to the extended Grzegorczyck hierarchy so that complexity statements in terms of $\mathscr{F}_\alpha$ can often be refined as statements in terms of $\mathbf{F}_\alpha$. These definitions allow one to employ the familiar vocabulary of complexity theory, reductions, and completeness instead of the more ad hoc notions used thus far. This will hopefully foster the reuse of "canonical problems" in establishing high complexity results rather than proofs from first principles, i.e., resource-bounded Turing machines.

A pattern emerges in the list of known $\mathbf{F}_\alpha$-complete problems, allowing one to answer a natural concern already expressed by Clote [1986]: "What do complexity classes for such rapidly growing functions really mean?" Indeed, beyond the intellectual satisfaction one might find in establishing a problem as complete for some class, being $\mathbf{F}_\alpha$-complete brings additional information on the problem itself: that it relies in some essential way on the ordinal ω^α being well ordered. All problems in Section 6 match this pattern, as their decision algorithms rely on well-quasi-orders with maximal order type ω^α for their termination, for which length function theorems then allow one to derive $\mathbf{F}_\alpha$ bounds.

Finally, we remark that currently there are no known natural problems of "intermediate" complexity, i.e., between ELEM and ACK, or between the latter and HACK. Parametric versions of LCM or LCS seem like good candidates for this, but so far the best lower and upper bounds do not quite match (e.g., see Karandikar and Schmitz [2013]). It would be interesting to find examples that exercise the intermediate levels of the $(\mathbf{F}_\alpha)_\alpha$ hierarchy.

APPENDIX

A. SUBRECURSIVE HIERARCHIES

This section presents the technical background and proofs missing from the main text.

A.1. Hardy Functions

Let $h\colon \mathbb{N} \to \mathbb{N}$ be a strictly increasing function. The *Hardy functions* $(h^\alpha)_{\alpha<\varepsilon_0}$ controlled by h are defined inductively by

$$h^0(x) \stackrel{\text{def}}{=} x, \qquad h^{\alpha+1}(x) \stackrel{\text{def}}{=} h^\alpha(h(x)), \qquad h^\lambda(x) \stackrel{\text{def}}{=} h^{\lambda(x)}(x). \tag{29}$$

A definition related to fundamental sequences is that of the *predecessor* at x of an ordinal greater than 0, which recursively considers the xth element in the fundamental sequence of limit ordinals until a successor ordinal is found:

$$P_x(\alpha+1) \stackrel{\text{def}}{=} \alpha, \qquad P_x(\lambda) \stackrel{\text{def}}{=} P_x(\lambda(x)). \tag{30}$$

Using predecessors, the definition of the Hardy functions becomes even simpler: for $\alpha > 0$,

$$h^\alpha(x) \stackrel{\text{def}}{=} h^{P_x(\alpha)}(h(x)). \tag{31}$$

For instance, observe that $h^k(x)$ 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 diagonalization to handle limit ordinals. The usual Hardy functions H^α are then obtained by fixing $H(x) \stackrel{\text{def}}{=} \mathrm{succ}(x) = x+1$.

The Hardy functions enjoy a number of properties (see Fairtlough and Wainer [1992] and Cichoń and Tahhan Bittar [1998]). They are *expansive*, and they are *monotonic* with respect to both the base function h and to the argument x: for all $g \le h$, $x \le y$, and α,

$$x \le h^\alpha(x), \qquad g^\alpha(x) \le h^\alpha(x), \qquad h^\alpha(x) \le h^\alpha(y). \tag{32}$$

As often with subrecursive functions, what the Hardy functions lack is monotonicity in the ordinal index (see Section A.2).

By transfinite induction on ordinals, we also find several identities:

$$h^{\omega^\alpha \cdot c} = F^c_{h,\alpha}, \tag{33}$$

$$h^{\alpha+\beta} = h^\alpha \circ h^\beta. \tag{34}$$

Note that (33) entails the expansiveness and monotonicity of the fast-growing functions.

Equation (34) is extremely valuable: it shows that —up to some extent—the *composition* of Hardy functions can be internalized in the ordinal index. However, here we run into a limitation of considering "set-theoretic" ordinal indices: informally, (34) is implicitly restricted to ordinals $\alpha + \beta$ "in CNF." Formally, it requires

$\alpha + \beta = \alpha \oplus \beta$, where "$\oplus$" denotes the natural sum operation. For instance, it fails in $H^1(H^\omega(x)) = H^1(H^x(x+1)) = 2x+2 > 2x+1 = H^\omega(x)$, although $1+\omega = \omega$. We will discuss this point further in Section A.6.

Remark A.1. Thanks to (33), the definitions of the $(\mathscr{F}_{<\alpha})_\alpha$ and $(\mathbf{F}_\alpha)_\alpha$ classes can be restated purely in terms of Hardy functions. Indeed,

$$\mathscr{F}_{<\alpha} = \bigcup_{\beta<\alpha, c<\omega} \text{FDTime}\big(F^c_\beta(n)\big) = \bigcup_{\beta<\alpha, c<\omega} \text{FDTime}(H^{\omega^\beta\cdot c}(n)) = \bigcup_{\gamma<\omega^\alpha} \text{FDTime}(H^\gamma(n)),$$

$$\mathbf{F}_\alpha = \bigcup_{p\in\mathscr{F}_{<\alpha}} \text{DTime}(H^{\omega^\alpha}(p(n))).$$

A.2. Monotonicity

One of the issues of most subrecursive hierarchies of functions is that they are not monotone in the ordinal index: $\beta < \alpha$ does not necessarily imply $H^\beta \leq H^\alpha$, i.e., $H^{x+2}(x) = 2x+2 > 2x+1 = H^\omega(x)$. However, what is true is that they are *eventually* monotone: if $\beta < \alpha$, then there exists n_0 such that for all $x \geq n_0$, $H^\beta(x) \leq H^\alpha(x)$. This result (and others) can be proven using a *pointwise ordering*: for all x, define the $\prec_x$ relation as the transitive closure of

$$\alpha \prec_x \alpha+1, \qquad\qquad \lambda(x) \prec_x \lambda. \tag{35}$$

The relation "$\beta \prec_x \alpha$" is also noted "$\beta \in \alpha[x]$" in Schwichtenberg and Wainer [2012, pp. 158–163]), where the results of this section are proven.

The $\prec_x$ relations form a strict hierarchy of refinements of the ordinal ordering $<$:

$$\prec_0 \subsetneq \prec_1 \subsetneq \cdots \subsetneq \prec_x \subsetneq \cdots \subsetneq <. \tag{36}$$

We are going to use two main properties of the pointwise ordering:

$$x < y \qquad \text{implies} \qquad \lambda(x) \prec_y \lambda(y), \tag{37}$$

$$\beta \prec_x \alpha \qquad \text{implies} \qquad H^\beta(x) \leq H^\alpha(x). \tag{38}$$

For a first application, define the *norm* of an ordinal term as the maximal coefficient that appears in its normal form: if $\alpha = \omega^{\alpha_1}\cdot c_1 + \cdots + \omega^{\alpha_m}\cdot c_m$ with $\alpha_1 > \cdots > \alpha_m$ and $c_1, \ldots, c_m > 0$, then $N\alpha \stackrel{\text{def}}{=} \max\{c_1, \ldots, c_m, N\alpha_1, \ldots, N\alpha_m\}$. Then $\beta < \alpha$ implies $\beta \prec_{N\beta} \alpha$ [Schwichtenberg and Wainer 2012, p. 158]. Together with (38), this entails that for all $x \geq N\beta$, $H^\beta(x) \leq H^\alpha(x)$.

A.3. Ackermann Functions

In this section, we prove some basic properties of the Ackermann hierarchy of functions $(A_\alpha)_\alpha$ defined in Section 4.1.1. Its definition is less uniform than the fast-growing and Hardy functions, leading to slightly more involved proofs.

Lemma A.2. *For all $\alpha > 0$, $A_\alpha(0) \leq 1$.*

Proof. By transfinite induction over α. For $\alpha = 1$, $A_1(0) = 0 \leq 1$. For a successor ordinal $\alpha+1$, $A_{\alpha+1}(0) = 1$. For a limit ordinal λ, $A_\lambda(0) = A_{\lambda(0)}(0) \leq 1$ by induction hypothesis. □

As usual with subrecursive hierarchies, the main issue with the Ackermann functions is to prove various monotonicity properties in the argument and in the index.

LEMMA A.3. *For all $\alpha, \beta > 0$ and x, y:*

(i) if $\alpha > 1$, A_α is strictly expansive: $A_\alpha(x) > x$,
(ii) A_α is strictly monotone in its argument: if $y > x$, $A_\alpha(y) > A_\alpha(x)$,
(iii) $(A_\alpha)_\alpha$ is pointwise monotone in its index: if $\alpha \succ_x \beta$, $A_\alpha(x) \geq A_\beta(x)$.

PROOF. Let us first consider the case $\alpha = 1$: A_1 is strictly monotone, proving (ii). Regarding (i) for $\alpha = 2$, $A_2(x) = 2^x > x$ for all x.

We prove now the three statements by simultaneous transfinite induction over α. Assume that they hold for all $\beta < \alpha$ (and thus for all $\beta \prec_x \alpha$ for all x).

For (i),

—If α is a successor ordinal $\beta+1$, then $A_{\beta+1}(x) \geq A_\beta(x) > x$ by induction hypothesis (iii) and (i) on $\beta \prec_x \alpha$.
—If α is a limit ordinal λ, then $A_\lambda(x) = A_{\lambda(x)}(x) > x$ by induction hypothesis (i) on $\lambda(x) \prec_x \alpha$.

For (ii), it suffices to prove the result for $y = x + 1$:

—If α is a successor ordinal $\beta + 1$, then $A_\alpha(x + 1) = A_\beta(A_\alpha(x)) > A_\alpha(x)$ by induction hypothesis (i) on $\beta \prec_x \alpha$.
—If α is a limit ordinal λ, then $A_\lambda(x + 1) = A_{\lambda(x+1)}(x + 1) \geq A_{\lambda(x)}(x + 1)$ by induction hypothesis (iii) on $\lambda(x) \prec_{x+1} \lambda(x + 1)$ (recall Equation (37)), hence the result by induction hypothesis (ii) on $\lambda_x \prec_x \alpha$.

For (iii), it suffices to prove the result for $\alpha = \beta + 1$ and $\beta = \alpha(x)$ and rely on transitivity:

—If $\alpha = \beta + 1$, then we show (iii) by induction over x: the base case $x = 0$ stems from $A_\alpha(0) = A^0_\beta(1) = 1 \geq A_\beta(0)$ by Lemma A.2; the induction step $x + 1$ stems from $A_\alpha(x + 1) = A_\beta(A_\alpha(x)) \geq A_\beta(x + 1)$ using the induction hypothesis on x and (ii) on $\beta \prec_{A_\alpha(x)} \alpha$.
—If $\beta = \alpha(x)$, then $A_\alpha(x) = A_\beta(x)$ by definition. □

Our main interest in the Ackermann functions is their relation with the fast-growing ones.

LEMMA A.4. *For all $\alpha > 0$ and all x, $A_\alpha(x) \leq F_\alpha(x) \leq A_\alpha(6x + 5)$.*

PROOF. We only prove the second inequality, as the first one can be deduced from the various monotonicity properties of F_α and A_α. The case $x = 0$ is settled for all $\alpha > 0$ by checking that $F_\alpha(0) = 1 \leq 10 = A_1(5) \leq A_\alpha(5)$, since $1 \preceq_x \alpha$ for all $\alpha > 0$ and we can therefore apply Lemma A.3.(iii). Assume now that $x > 0$; we prove the statement by transfinite induction over $\alpha > 0$:

—For the base case $\alpha = 1$, $F_1(x) = 2x + 1 \leq 12x + 10 = A_1(6x + 5)$.
—For the successor case $\alpha + 1$, $A_{\alpha+1}(6x+5) = A^{5(x+1)}_\alpha(A^x_\alpha(1)) \geq A^{5(x+1)}_\alpha(x)$ by Lemma A.3. We show by induction over j that $A^{5j}_\alpha(x) \geq F^j_\alpha(x)$. This holds for the base case $j = 0$, and for the induction step, $A^5_\alpha(A^{5j}_\alpha(x)) \geq A^5_\alpha(F^j_\alpha(x))$ by induction hypothesis on j and Lemma A.3.(ii). Furthermore, for all $y > 0$, $A_\alpha(A^4_\alpha(y)) \geq A_\alpha(A^4_1(y)) = A_\alpha(16y) \geq A_\alpha(6y+5) \geq F_\alpha(y)$ by induction hypothesis on α, which shows that $A^5_\alpha(F^j_\alpha(x)) \geq F^{j+1}_\alpha(x)$ when choosing $y = F^j_\alpha(x) > 0$. Then $A^{5(x+1)}_\alpha(x) \geq F^{x+1}_\alpha(x) = F_{\alpha+1}(x)$, thus completing the proof in the successor case.
—For the limit case λ, $A_\lambda(6x + 5) = A_{\lambda(6x\,+\,5)}(6x + 5) \geq A_{\lambda(x)}(6x + 5) \geq F_{\lambda(x)}(x) = F_\lambda(x)$, using successively Lemma A.3.(iii) on $\lambda(x) \prec_{6x+5} \lambda(6x + 5)$ and the induction hypothesis on $\lambda(x) < \lambda$. □

A.4. Relativized Functions

We prove here the missing lemma from the proof of Theorem 4.2:

LEMMA A.5. *Let $h\colon \mathbb{N} \to \mathbb{N}$ be a function, α, β be two ordinals, and x_0 be a natural number. If for all $x \geq x_0$, $h(x) \leq F_\beta(x)$, then there exists an ordinal γ such that*

(i) *for all $x \geq x_0$, $F_{h,\alpha}(x) \leq F_{\beta+\alpha}(F_\gamma(x))$, and*
(ii) *$\gamma < \beta + \alpha$ whenever $\beta + \alpha > 0$.*

PROOF. Let us first fix some notations: write $\alpha = \omega^{\alpha_1} + \cdots + \omega^{\alpha_m}$ with $\alpha_1 \geq \cdots \geq \alpha_m$ and $\beta = \omega^{\beta_1} + \cdots + \omega^{\beta_n}$ with $\beta_1 \geq \cdots \geq \beta_n$, and let i be the maximal index in $\{1, \ldots, n\}$ such that $\beta_i \geq \alpha_1$, or set $i = 0$ if this does not occur. Define $\beta' \stackrel{\text{def}}{=} \omega^{\beta_1} + \cdots + \omega^{\beta_i}$ and $\gamma \stackrel{\text{def}}{=} \omega^{\beta_{i+1}} + \cdots + \omega^{\beta_n}$ (thus, $\beta' = 0$ if $i = 0$); then $\beta = \beta' + \gamma$ and $\beta + \alpha = \beta' + \alpha$. Note that this implies $\gamma < \omega^{\alpha_1} \leq \alpha \leq \beta + \alpha$, unless $\alpha = 0$ and then $\gamma = 0$, thus fulfilling (ii).

We first prove by transfinite induction over α that

$$F_{\beta'+\alpha} \circ F_\gamma \geq F_\gamma \circ F_{F_\beta,\alpha}. \tag{39}$$

PROOF OF (39). For the base case $\alpha = 0$, then $\gamma = 0$ and $\beta' = \beta$, and indeed

$$\begin{aligned} F_\beta(F_0(x)) &= F_\beta(x+1) \\ &\geq F_\beta(x) + 1 && \text{by monotonicity of } F_\beta \\ &= F_0(F_\beta(x)) \\ &= F_0(F_{F_\beta,0}(x)). \end{aligned}$$

For the successor case $\alpha + 1$ and assuming it holds for α, let us first show by induction over j that for all y,

$$F^j_{\beta'+\alpha}(F_\gamma(y)) \geq F_\gamma\big(F^j_{F_\beta,\alpha}(y)\big). \tag{40}$$

This immediately holds for the base case $j = 0$, and for the induction step,

$$\begin{aligned} F_{\beta'+\alpha}\big(F^j_{\beta'+\alpha}(F_\gamma(y))\big) &\geq F_{\beta'+\alpha}\big(F_\gamma(F^j_{F_\beta,\alpha}(y))\big) && \text{by induction hypothesis (40) on } j \\ &\geq F_\gamma\big(F_{F_\beta,\alpha}(F^j_{F_\beta,\alpha}(y))\big) && \text{by induction hypothesis (39) on } \alpha < \alpha + 1. \end{aligned}$$

This yields the desired inequality:

$$\begin{aligned} F_{\beta'+\alpha+1}(F_\gamma(x)) &= F^{F_\gamma(x)+1}_{\beta'+\alpha}(F_\gamma(x)) \\ &\geq F^{x+1}_{\beta'+\alpha}(F_\gamma(x)) \\ &\geq F_\gamma\big(F^{x+1}_{F_\beta,\alpha}(x)\big) \\ &= F_\gamma(F_{F_\beta,\alpha+1}(x)) \end{aligned}$$

using (40) with $j = x + 1$ and $y = x$.

For the limit case λ,

$$\begin{aligned} F_{\beta'+\lambda}(F_\gamma(x)) &= F_{\beta'+\lambda(F_\gamma(x))}(F_\gamma(x)) \\ &\leq F_{\beta'+\lambda(x)}(F_\gamma(x)) && \text{since } \lambda(x) \prec_{F_\gamma(x)} \lambda(F_\gamma(x)) \\ &\leq F_\gamma\big(F_{F_\beta,\lambda(x)}(x)\big) && \text{by induction hypothesis (39) on } \lambda(x) < \lambda \\ &= F_\gamma(F_{F_\beta,\lambda}(x)). \quad \square \end{aligned}$$

Returning to the main proof, a simple induction over α shows that for all $x \geq x_0$,

$$F_{h,\alpha}(x) \leq F_{F_\beta,\alpha}(x). \tag{41}$$

We then conclude for (i) that for all $x \geq x_0$,

$$\begin{aligned} F_{h,\alpha}(x) &\leq F_{F_{\beta},\alpha}(x) && \text{by (41)} \\ &\leq F_{\gamma}(F_{F_{\beta},\alpha}(x)) && \text{by expansivity of } F_{\gamma} \\ &\leq F_{\beta'+\alpha}(F_{\gamma}(x)) && \text{by (39).} \quad \square \end{aligned}$$

A.5. Nonstandard Assignment of Fundamental Sequences

Here we show the omitted details of the proof of Theorem 4.4.

LEMMA A.6. *Let s:$\mathbb{N} \to \mathbb{N}$ be a monotone function and α be an ordinal.*

—If s is strictly expansive, then $F_{\alpha,s} \leq F_{s,\alpha} \circ s$, and
—otherwise $F_{\alpha,s} \leq F_{\alpha}$.

PROOF. For the first point, let us show that

$$s(F_{\alpha,s}(x)) \leq F_{s,\alpha}(s(x)) \tag{42}$$

for all monotone s with $s(x) > x$, all α and all x, which entails the lemma since s is expansive. We proceed by transfinite induction over α. For the base case, $F_{s,0}(s(x)) = s(s(x)) \geq s(x+1) = s(F_{0,s}(x))$ since s is monotone and strictly expansive. For the successor case, $F_{s,\alpha+1}(s(x)) = F_{s,\alpha}^{s(x)+1}(s(x)) \geq s(F_{\alpha,s}^{s(x)}(x)) = s(F_{\alpha+1,s}(x))$, where the middle inequality stems from the fact that $F_{s,\alpha}^{j}(s(x)) \geq s(F_{\alpha,s}^{j}(x))$, as can be seen by induction on j using the induction hypothesis on $\alpha < \alpha + 1$. For the limit case, observe that $\lambda(x)_s \prec_{s(x)} \lambda(s(x))$, thus $F_{s,\lambda}(s(x)) = F_{s,\lambda(s(x))}(s(x)) \geq F_{s,\lambda(x)_s}(s(x)) \geq s(F_{\lambda(x)_s,s}(x)) = s(F_{\lambda,s}(x))$ using the induction hypothesis on $\lambda(x)_s < \lambda$.

The second point is straightforward by induction over α. $\square$

LEMMA A.7. *For all α, $F_0 \circ F_{\alpha} \leq F_{\alpha,\mathrm{id}} \circ F_0$.*

PROOF. By induction over α. For the zero case, $F_0(F_0(x)) = x + 2 = F_{0,\mathrm{id}}(F_0(x))$. For the successor case, we can check that $F_{\alpha,\mathrm{id}}^{j}(x+1) \geq F_{\alpha}^{j}(x) + 1$ for all j using the induction hypothesis on α, thus $F_{\alpha,\mathrm{id}}(x+1) = F_{\alpha,\mathrm{id}}^{x+1}(x+1) \geq F_{\alpha}^{x+1}(x) + 1 = F_{\alpha+1}(x) + 1$. For the limit case, note that $\lambda(x) \prec_{x+1} \lambda(x+1)$, thus $F_{\lambda,\mathrm{id}}(x+1) = F_{\lambda_{x+1},\mathrm{id}}(x+1) \geq F_{\lambda(x+1)}(x) + 1 \geq F_{\lambda(x)}(x) + 1 = F_{\lambda}(x) + 1$. $\square$

A.6. Composing Hardy Functions

The purpose of this section is to provide the technical details for the proof of Lemma 4.6.

The *natural sum* $\alpha \oplus \beta$ of two ordinals written as $\alpha = \omega^{\alpha_1} + \cdots + \omega^{\alpha_m}$ with $\alpha_1 \geq \cdots \geq \alpha_m$ and $\beta = \omega^{\beta_1} + \cdots \omega^{\beta_n}$ with $\beta_1 \geq \cdots \geq \beta_n$ can be defined as the ordinal $\omega^{\gamma_1} + \cdots + \omega^{\gamma_{m+n}}$ where the γ_i's range over $\{\alpha_j \mid 1 \leq j \leq m\} \cup \{\beta_k \mid 1 \leq k \leq n\}$ in nonincreasing order. For instance, $\omega^2 + \omega^{\omega} = \omega^{\omega}$ but $\omega^2 \oplus \omega^{\omega} = \omega^{\omega} + \omega^2$.

LEMMA A.8. *For all ordinals α and β, and all functions h,*

$$h^{\alpha} \circ h^{\beta} \leq h^{\alpha \oplus \beta}.$$

PROOF. Write $\alpha = \omega^{\alpha_1} + \cdots + \omega^{\alpha_m}$ with $\alpha_1 \geq \cdots \geq \alpha_m$ and $\beta = \omega^{\beta_1} + \cdots + \omega^{\beta_n}$ with $\beta_1 \geq \cdots \geq \beta_n$, then $\alpha \oplus \beta = \omega^{\gamma_1} + \cdots + \omega^{\gamma_{m+n}}$. We prove the lemma by transfinite induction over β: it holds immediately for the base case since $\alpha \oplus 0 = \alpha$ and for the successor case since $\alpha \oplus (\beta + 1) = (\alpha \oplus \beta) + 1$. For the limit case, let i be the last index of β_n among the γ_j in the CNF of $\alpha \oplus \beta$. If $i = m + n$, then $\alpha \oplus (\beta(x)) = (\alpha \oplus \beta)(x)$ and the statement

holds. Otherwise, define $\gamma \stackrel{\text{def}}{=} \omega^{\gamma_1} + \cdots + \omega^{\gamma_i}$ and $\gamma' \stackrel{\text{def}}{=} \omega^{\gamma_{i+1}} + \cdots + \omega^{\gamma_{m+n}}$. For all x,

$$\begin{aligned}
h^{\alpha\oplus\beta} &= h^{\gamma}(h^{\gamma'}(x)) && \text{by (34)}\\
&= h^{\gamma(h^{\gamma'}(x))}(h^{\gamma'}(x)) && \text{since } \gamma \text{ is a limit ordinal}\\
&\geq h^{\gamma(x)}(h^{\gamma'}(x)) && \text{since } \gamma(x) \prec_{[h^{\gamma'}(x)]} \gamma(h^{\gamma'}(x))\\
&= h^{\alpha\oplus(\beta(x))}(x) && \text{by (34)}\\
&\geq h^{\alpha}(h^{\beta(x)}(x)) && \text{by induction hypothesis on } \beta(x) < \beta\\
&= h^{\alpha}(h^{\beta}(x)). \quad \square
\end{aligned}$$

Corollary A.9. *Let α be an ordinal and f a function in $\mathscr{F}_{<\alpha}$. Then there exists g in $\mathscr{F}_{<\alpha}$ such that $f \circ F_\alpha \leq F_\alpha \circ g$.*

Proof. As f is in some $\mathscr{F}_\beta$ for $\beta < \alpha$, $f \leq F_\beta^c$ for some finite c by Löb and Wainer [1970, Theorem 2.10], thus $f \leq H^{\omega^\beta \cdot c}$ by (33), and we let $g \stackrel{\text{def}}{=} H^{\omega^\beta \cdot c}$, which indeed belongs to $\mathscr{F}_\beta \subseteq \mathscr{F}_{<\alpha}$. Still by (33), $F_\alpha = H^{\omega^\alpha}$. Observe that $\omega^\beta \cdot c < \omega^\alpha$, hence $(\omega^\beta \cdot c) \oplus \omega^\alpha = \omega^\alpha + \omega^\beta \cdot c$. By (34), $H^{\omega^\alpha + \omega^\beta \cdot c} = H^{\omega^\alpha} \circ H^{\omega^\beta \cdot c}$. Applying (33) and Proposition A.8, we obtain that $f \circ F_\alpha \leq g \circ F_\alpha \leq F_\alpha \circ g$. $\square$

A.7. Computing Hardy Functions

In this section, we explain how to compute Hardy functions, thus providing the background material for the proof of Theorem 5.1. This type of results is pretty standard—see for instance Wainer [1970], Fartlough and Wainer [1998], or Schichtenberg and Wainer [2012, pp. 159–160]—but the particular way that we employ is closer in spirit to the viewpoint employed in Haddad et al. [2012], Karandikar and Schmitz [2013], and Haase et al. [2014].

A.7.1. Hardy Computations. Using (31), let us call a *Hardy computation* for $h^\alpha(n)$ a sequence of pairs $\langle \alpha_0, n_0 \rangle, \langle \alpha_1, n_1 \rangle, \ldots, \langle \alpha_\ell, n_\ell \rangle$, where $\alpha_0 = \alpha$, $n_0 = n$, $\alpha_\ell = 0$, and at each step $0 < i \leq \ell$, $\alpha_i = P_{n_{i-1}}(\alpha_{i-1})$ and $n_i = h(n_{i-1})$. An invariant of this computation is that $h^{\alpha_i}(n_i) = h^\alpha(n)$ at all steps $0 \leq i \leq \ell$, hence $n_\ell = h^\alpha(n)$. Since h is increasing, the n_i values increase throughout this computation, whereas the α_i values decrease, and termination is guaranteed.

Our plan is to implement the Hardy computation of $h^\alpha(n)$ using a Turing machine, which essentially needs to implement the ℓ steps $\langle \alpha_i, n_i \rangle \to \langle P_{n_{i-1}}(\alpha_{i-1}), h(n_{i-1}) \rangle$. We assume h to be an elementarily constructible expansive function such that $h(n)$ can be computed in $e(h(n))$ for some fixed monotone elementary function e. Then, the complexity of a single step will depend mainly on $h(n_{i-1}) \leq h^\ell(n)$ and on the complexity of updating α_i.

A.7.2. Cichoń Functions. To measure the length ℓ of a Hardy computation for $h^\alpha(n)$, we define a family $(h_\alpha)_\alpha$ of functions $\mathbb{N} \to \mathbb{N}$ by induction on the ordinal index:

$$h_0(x) \stackrel{\text{def}}{=} 0, \qquad h_{\alpha+1}(x) \stackrel{\text{def}}{=} 1 + h_\alpha(h(x)), \qquad h_\lambda(x) \stackrel{\text{def}}{=} h_{\lambda(x)}(x). \tag{43}$$

This family is also known as the *length hierarchy* and was defined by Cichoń and Tahhan Bittar [1998]. It satisfies several interesting identities:

$$h^\alpha(x) = h^{h_\alpha(x)}(x), \qquad h^\alpha(x) \geq h_\alpha(x) + x. \tag{44}$$

Its main interest here is that it measures the length of Hardy computations: $\ell = h_\alpha(n) \leq h^\alpha(n)$ by the preceding equations, which in turn implies $h^\ell(n) = h^\alpha(n)$.

A.7.3. Encoding Ordinal Terms. It remains to bound the complexity of computing $\alpha_i = P_{n_{i-1}}(\alpha_{i-1})$. Assuming some reasonable string encoding of the terms denoting the α_i (e.g., Haase et al. [2014]), we will consider that each α_i can be computed in time $p(|\alpha_i|)$ a monotone polynomial function of the size $|\alpha_i|$ of its term representation, and we will rather concentrate on bounding this size. We define it by induction on the term denoting α_i:

$$|0| \stackrel{\text{def}}{=} 0, \qquad |\omega^\alpha| \stackrel{\text{def}}{=} 1 + |\alpha|, \qquad |\alpha + \alpha'| \stackrel{\text{def}}{=} |\alpha| + |\alpha'|. \tag{45}$$

Let us also recall the definition of the *slow-growing hierarchy* $(G_\alpha)_\alpha$:

$$G_0(x) \stackrel{\text{def}}{=} 0, \qquad G_{\alpha+1}(x) \stackrel{\text{def}}{=} 1 + G_\alpha(x), \qquad G_\lambda(x) \stackrel{\text{def}}{=} G_{\lambda(x)}(x). \tag{46}$$

The slow-growing function satisfy several natural identities:

$$G_\alpha(x) = 1 + G_{P_x(\alpha)}(x), \tag{47}$$

$$G_\alpha(x+1) > G_\alpha(x), \tag{48}$$

$$\text{if } \beta \prec_x \alpha \text{ then } G_\beta(x) \leq G_\alpha(x). \tag{49}$$

Furthermore,

$$G_{\alpha+\alpha'}(x) = G_\alpha(x) + G_{\alpha'}(x)\,, \qquad G_{\omega^\alpha}(x) = (x+1)^{G_\alpha(x)}. \tag{50}$$

Hence, $G_\alpha(x)$ is the elementary function that results from substituting $x+1$ for every occurrence of ω in the CNF of α [Schwichtenberg and Wainer 2012, p. 159].

LEMMA A.10. *Let $x > 0$. Then $|\alpha| \leq G_\alpha(x)$.*

PROOF. By induction over the term denoting α: $|0| = 0 = G_0(x)$, $|\omega^\alpha| = 1 + |\alpha| \leq (x+1)^{|\alpha|} \leq (x+1)^{G_\alpha(x)} = G_{\omega^\alpha}(x)$, and $|\alpha + \alpha'| = |\alpha| + |\alpha'| \leq G_\alpha(x) + G_{\alpha'}(x) = G_{\alpha+\alpha'}(x)$. □

LEMMA A.11. *If $\langle \alpha_0, n_0 \rangle, \ldots, \langle \alpha_\ell, n_\ell \rangle$ is a Hardy computation for $h^\alpha(n)$ with $n > 0$, then for all $0 \leq i \leq \ell$, $|\alpha_i| \leq G_\alpha(n_\ell)$.*

PROOF. We distinguish two cases. If $i = 0$, then $|\alpha_0| = |\alpha| \leq G_\alpha(n)$ by Lemma A.10 since $n > 0$, and hence $|\alpha_0| \leq G_\alpha(n_\ell)$ since $n_\ell \geq n$ by (48). If $i > 0$, then

$$\begin{aligned}
|\alpha_i| &= |P_{n_{i-1}}(\alpha_{i-1})| \\
&\leq G_{P_{n_{i-1}}(\alpha_{i-1})}(n_{i-1}) && \text{by Lemma A.10 since } n_{i-1} \geq n > 0 \\
&< G_{\alpha_{i-1}}(n_{i-1}) && \text{by (47)} \\
&\leq G_\alpha(n_{i-1}) && \text{since } \alpha_{i-1} \prec_{n_{i-1}} \alpha \text{ by (49)} \\
&\leq G_\alpha(n_\ell) && \text{since } n_{i-1} \leq n_\ell \text{ by (48).}
\end{aligned}$$ □

The restriction to $n > 0$ in Lemma A.11 is not a big issue: either $h(0) = 0$ and then $h^\alpha(0) = 0$ or $h(0) > 0$ and then $h^{\gamma+\omega^\beta}(0) = h^\gamma(h(0))$, and we can proceed from γ instead of $\gamma + \omega^\beta$ as initial ordinal of our computation.

A.7.4. Wrapping Up. To conclude, each of the $\ell \leq h^\alpha(n)$ steps of a Hardy computation for $h^\alpha(n)$ needs to compute

—α_i, in time $p(G_\alpha(h^\alpha(n)))$ since $|\alpha_i| \leq G_\alpha(h^\alpha(n))$ and p was assumed monotone, and
—n_i, in time $e(h^\alpha(n))$ since $h(n_{i-1}) \leq h^\alpha(n)$ and e was assumed to be monotone.

This yields the following statement.

PROPOSITION A.12. *The Hardy function h^α can be computed in time*

$$O\big(h^\alpha(n) \cdot \big(p(G_\alpha(h^\alpha(n))) + e(h^\alpha(n))\big)\big).$$

REFERENCES

Parosh A. Abdulla, Karlis Čerāns, Bengt Jonsson, and Yih-Kuen Tsay. 2000. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation* 160, 1–2, 109–127. DOI:http://dx.doi.org/10.1006/inco.1999.2843

Parosh A. Abdulla and Giorgio Delzanno. 2006. On the coverability problem for constrained multiset rewriting. In *Proceedings of the 2006 AVIS Conference*.

Parosh A. Abdulla, Giorgio Delzanno, and Laurent Van Begin. 2011. A classification of the expressive power of well-structured transition systems. *Information and Computation* 209, 3, 248–279. DOI:http://dx.doi.org/10.1016/j.ic.2010.11.003

Parosh A. Abdulla and Bengt Jonsson. 1996. Verifying programs with unreliable channels. *Information and Computation* 127, 2, 91–101. DOI:http://dx.doi.org/10.1006/inco.1996.0053

Parosh A. Abdulla and Aletta Nylén. 2001. Timed Petri nets and BQOs. In *Applications and Theory of Petri Nets 2001*. Lecture Notes in Computer Science, Vol. 2075. Springer, 53–70. DOI:http://dx.doi.org/10.1007/3-540-45740-2_5

Sergio Abriola, Santiago Figueira, and Gabriel Senno. 2015. Linearizing well-quasi orders and bounding the length of bad sequences. *Theoretical Computer Science* 603, 3–22. DOI:http://dx.doi.org/10.1016/j.tcs.2015.07.012

Rajeev Alur and David L. Dill. 1994. A theory of timed automata. *Theoretical Computer Science* 126, 2, 183–235. DOI:http://dx.doi.org/10.1016/0304-3975(94)90010-8

Mohamed Faouzi Atig, Ahmed Bouajjani, Sebastian Burckhardt, and Madanlal Musuvathi. 2010. On the verification problem for weak memory models. In *Proceedings of the 2010 POPL Conference*. ACM, New York, NY, 7–18. DOI:http://dx.doi.org/10.1145/1706299.1706303

Pablo Barceló, Diego Figueira, and Leonid Libkin. 2013. Graph logics with rational relations. *Logical Methods in Computer Science* 9, 3, Article No. 1. DOI:http://dx.doi.org/10.2168/LMCS-9(3:1)2013

Arnold Beckmann. 2001. Exact bounds for lengths of reductions in typed λ-calculus. *Journal of Symbolic Logic* 66, 3, 1277–1285. DOI:http://dx.doi.org/10.2307/2695106

Michel Blockelet and Sylvain Schmitz. 2011. Model-checking coverability graphs of vector addition systems. In *Mathematical Foundations of Computer Science*. Lecture Notes in Computer Science, 6907. Springer, 108–119. DOI:http://dx.doi.org/10.1007/978-3-642-22993-0_13

Rémi Bonnet, Alain Finkel, Serge Haddad, and Fernando Rosa-Velardo. 2010. *Comparing Petri Data Nets and Timed Petri Nets*. Research Report LSV-10-23. LSV, ENS Cachan. http://lsv.fr/Publis/rrpublis?onlykey=rr-lsv-10-23

Patricia Bouyer, Nicolas Markey, Joël O. Ouaknine, Philippe Schnoebelen, and James B. Worrell. 2012. On termination and invariance for faulty channel machines. *Formal Aspects of Computing* 24, 4–6, 595–607. DOI:http://dx.doi.org/10.1007/s00165-012-0234-7

Davide Bresolin, Dario Della Monica, Angelo Montanari, Pietro Sala, and Guido Sciavicco. 2012. Interval temporal logics over finite linear orders: The complete picture. In *Proceedings of the 2012 ECAI Conference*. 199–204. DOI:http://dx.doi.org/10.3233/978-1-61499-098-7-199

Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. 1996. Unreliable channels are easier to verify than perfect channels. *Information and Computation* 124, 1, 20–31. DOI:http://dx.doi.org/10.1006/inco.1996.0003

Pierre Chambart and Philippe Schnoebelen. 2007. Post embedding problem is not primitive recursive, with applications to channel systems. In *FSTTCS 2007: Foundations of Software Technology and Theoretic Computer Science*. Lecture Notes in Computer Science, Vol. 4855. Springer, 265–276. DOI:http://dx.doi.org/10.1007/978-3-540-77050-3_22

Pierre Chambart and Philippe Schnoebelen. 2008a. The ω-regular post embedding problem. In *Foundations of Software Science and Computational Structures*. Lecture Notes in Computer Science, Vol. 4962. Springer, 97–111. DOI:http://dx.doi.org/10.1007/978-3-540-78499-9_8

Pierre Chambart and Philippe Schnoebelen. 2008b. The ordinal recursive complexity of lossy channel systems. In *Proceedings of the 2008 LICS Conference*. IEEE, Los Alamitos, CA, 205–216. DOI:http://dx.doi.org/10.1109/LICS.2008.47

E. Adam Cichoń and Elias Tahhan Bittar. 1998. Ordinal recursive bounds for Higman's theorem. *Theoretical Computer Science* 201, 1–2, 63–84. DOI:http://dx.doi.org/10.1016/S0304-3975(97)00009-1

Peter Clote. 1986. On the finite containment problem for Petri nets. *Theoretical Computer Science* 43, 99–105. DOI:http://dx.doi.org/10.1016/0304-3975(86)90169-6

Peter Clote. 1999. Computation models and function algebras. In *Handbook of Computability Theory*, E. R. Griffor (Ed.). Studies in Logic and the Foundations of Mathematics, Vol. 140. Elsevier, 589–681. DOI:http://dx.doi.org/10.1016/S0049-237X(99)80033-0

M. Dauchet and S. Tison. 1990. The theory of ground rewrite systems is decidable. In *Proceedings of the 1990 LICS Conference*. IEEE, Los Alamitos, CA, 242–248. DOI:http://dx.doi.org/10.1109/LICS.1990.113750

Dick H. J. de Jongh and Rohit Parikh. 1977. Well-partial orderings and hierarchies. *Indagationes Mathematicae* 39, 3, 195–207. http://dx:doi:org/10:1016/1385-7258(77)90067-1

Normann Decker and Daniel Thoma. 2015. On Freeze LTL with ordered attributes. Preprint. Available at http://arxiv.org/abs/1504.06355

Stéphane Demri and Ranko Lazić. 2009. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic* 10, 3, Article No. 16. DOI:http://dx.doi.org/10.1145/1507244.1507246

J. Michael Dunn and Greg Restall. 2002. Relevance logic. In *Handbook of Philosophical Logic*, D. M. Gabbay and F. Guenthner (Eds.), Vol. 6. Kluwer, 1–128. DOI:http://dx.doi.org/10.1007/978-94-017-0460-1_1

Jacob Elgaard, Nils Klarlund, and Anders Møller. 1998. MONA 1.x: New techniques for WS1S and WS2S. In *Computer Aided Verification*. Lecture Notes in Computer Science, Vol. 1427. Springer, 516–520. DOI:http://dx.doi.org/10.1007/BFb0028773

Patrick C. Fischer, Albert R. Meyer, and Arnold L. Rosenberg. 1968. Counter machines and counter languages. *Mathematical Systems Theory* 2, 3, 265–283. DOI:http://dx:doi:org/10:1007/BF01694011

Matthew V. H. Fairtlough and Stanley S. Wainer. 1992. Ordinal complexity of recursive definitions. *Information and Computation* 99, 2, 123–153. DOI:http://dx.doi.org/10.1016/0890-5401(92)90027-D

Matthew V. H. Fairtlough and Stanley S. Wainer. 1998. Hierarchies of provably recursive functions. In *Handbook of Proof Theory*, S. Buss (Ed.). Studies in Logic and the Foundations of Mathematics, Vol. 137. Elsevier, 149–207. DOI:http://dx.doi.org/10.1016/S0049-237X(98)80018-9

Solomon Feferman. 1962. Classification of recursive functions by means of hierarchies. *Transactions of the American Mathematical Society* 104, 101–122. DOI:http://dx.doi.org/10.1090/S0002-9947-1962-0142453-3

Diego Figueira. 2012. Alternating register automata on finite words and trees. *Logical Methods in Computer Science* 8, 1, Article No. 22. DOI:http://dx.doi.org/10.2168/LMCS-8(1:22)2012

Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. 2011. Ackermannian and primitive-recursive bounds with Dickson's lemma. In *Proceedings of the 2011 LICS Conference*. IEEE, Los Alamitos, CA, 269–278. DOI:http://dx.doi.org/10.1109/LICS.2011.39

Diego Figueira, Piotr Hofman, and Sławomir Lasota. 2015. Relating timed and register automata. *Mathematical Structures in Computer Science.* To appear. DOI:http://dx.doi.org/10.1017/S0960129514000322

Diego Figueira and Luc Segoufin. 2009. Future-looking logics on data words and trees. In *Mathematical Foundations of Computer Science 2009*. Lecture Notes in Computer Science, Vol. 5734. Springer, 331–343. DOI:http://dx:doi:org/10:1007/978-3-642-03816-729

Alain Finkel. 1987. A generalization of the procedure of Karp and Miller to well structured transition systems. In *Automata, Languages and Programming*. Lecture Notes in Computer Science, Vol. 267. Springer, 499–508. DOI:http://dx.doi.org/10.1007/3-540-18088-5_43

Alain Finkel and Philippe Schnoebelen. 2001. Well-structured transition systems everywhere! *Theoretical Computer Science* 256, 1–2, 63–92. DOI:http://dx.doi.org/10.1016/S0304-3975(00)00102-X

Harvey M. Friedman. 1999. Some decision problems of enormous complexity. In *Proceedings of the 1999 LICS Conference*. IEEE, Los Alamitos, CA, 2–13. DOI:http://dx.doi.org/10.1109/LICS.1999.782577

Andrzej Grzegorczyk. 1953. Some classes of recursive functions. *Rozprawy Matematyczne* 4, 1–46. http://matwbn.icm.edu.pl/ksiazki/rm/rm04/rm0401.pdf.

Christoph Haase, Sylvain Schmitz, and Philippe Schnoebelen. 2014. The power of priority channel systems. *Logical Methods in Computer Science* 10, 4, Article No. 4. DOI:http://dx.doi.org/10.2168/LMCS-10(4:4)2014

Michel Hack. 1976. The equality problem for vector addition systems is undecidable. *Theoretical Computer Science* 2, 1, 77–95. DOI:http://dx.doi.org/10.1016/0304-3975(76)90008-6

Serge Haddad, Sylvain Schmitz, and Philippe Schnoebelen. 2012. The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In *Proceedings of the 2012 LICS Conference*. IEEE, Los Alamitos, CA, 355–364. DOI:http://dx.doi.org/10.1109/LICS.2012.46

Matthew Hague. 2014. Senescent ground tree rewriting systems. In *Proceedings of the 2014 CSL-LICS Conference*. ACM, New York, NY, Article No. 48. DOI:http://dx.doi.org/10.1145/2603088.2603112

Joseph Y. Halpern and Yoav Shoham. 1991. A propositional modal logic of time intervals. *Journal of the ACM* 38, 4, 935–962. DOI:http://dx.doi.org/10.1145/115234.115351

Piotr Hofman and Patrick Totzke. 2014. Trace inclusion for one-counter nets revisited. In *Reachability Problems*. Lecture Notes in Computer Science, Vol. 8762. Springer, 151–162. DOI:http://dx.doi.org/10.1007/978-3-319-11439-2_12

Petr Jančar. 1995. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science* 148, 2, 281–301. DOI:http://dx.doi.org/10.1016/0304-3975(95)00037-W

Petr Jančar. 2001. Nonprimitive recursive complexity and undecidability for Petri net equivalences. *Theoretical Computer Science* 256, 1–2, 23–30. DOI:http://dx.doi.org/10.1016/S0304-3975(00)00100-6

Marcin Jurdziński and Ranko Lazić. 2011. Alternating automata on data trees and XPath satisfiability. *ACM Transactions on Computational Logic* 12, 3, Article No. 19. DOI:http://dx:doi:org/10:1145/1929954:1929956

Prateek Karandikar and Sylvain Schmitz. 2013. The parametric ordinal-recursive complexity of Post embedding problems. In *Foundations of Software Science and Computation Structures*. Lecture Notes in Computer Science, Vol. 7794. Springer, 273–288. DOI:http://dx.doi.org/10.1007/978-3-642-37075-5_18

Prateek Karandikar and Philippe Schnoebelen. 2012. Cutting through regular post embedding problems. In *Computer Science—Theory and Applications*. Lecture Notes in Computer Science, Vol. 7353. Springer, 229–240. DOI:http://dx.doi.org/10.1007/978-3-642-30642-6_22

S. Rao Kosaraju. 1982. Decidability of reachability in vector addition systems. In *Proceedings of the 1982 STOC Conference*. ACM, New York, NY, 267–281. DOI:http://dx.doi.org/10.1145/800070.802201

Ron Koymans. 1990. Specifying real-time properties with metric temporal logic. *Real-Time Systems* 2, 4, 255–299. DOI:http://dx.doi.org/10.1007/BF01995674

Joseph B. Kruskal. 1972. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A* 13, 3, 297–305. DOI:http://dx.doi.org/10:1016/0097-3165(72)90063-5

Jean-Luc Lambert. 1992. A structure to decide reachability in Petri nets. *Theoretical Computer Science* 99, 1, 79–104. DOI:http://dx.doi.org/10.1016/0304-3975(92)90173-D

Sławomir Lasota and Igor Walukiewicz. 2008. Alternating timed automata. *ACM Transactions on Computational Logic* 9, 2, Article No. 10. DOI:http://dx.doi.org/10.1145/1342991.1342994

Ranko Lazić, Tom Newcomb, Joël O. Ouaknine, Andrew W. Roscoe, and James B. Worrell. 2008. Nets with tokens which carry data. *Fundamenta Informaticae* 88, 3, 251–274.

Ranko Lazić, Joël O. Ouaknine, and James B. Worrell. 2013. Zeno, Hercules and the Hydra: Downward rational termination is Ackermannian. In *Mathematical Foundations of Computer Science*. Lecture Notes in Computer Science, Vol. 8087. Springer, 643–654. DOI:http://dx.doi.org/10.1007/978-3-642-40313-2_57

Ranko Lazić and Sylvain Schmitz. 2015. Non-elementary complexities for branching VASS, MELL, and extensions. *ACM Transactions on Computational Logic* 16, 3, Article No. 20. DOI:http://dx.doi.org/10.1145/2733375

Jérôme Leroux. 2011. Vector addition system reachability problem: A short self-contained proof. In *Proceedings of the 2011 POPL Conference*. ACM, New York, NY, 307–316. DOI:http://dx.doi.org/10.1145/1926385.1926421

Jérôme Leroux and Sylvain Schmitz. 2015. Demystifying reachability in vector addition systems. In *Proceedings of the 2015 LICS Conference*. IEEE, Los Alamitos, CA, 56–67. DOI:http://dx.doi.org/10.1109/LICS.2015.16

Richard J. Lipton. 1976. *The Reachability Problem Requires Exponential Space*. Technical Report 62. Department of Computer Science, Yale University, New Haven, CT. http://www.cs.yale.edu/publications/techreports/tr63.pdf.

Martin H. Löb and Stanley S. Wainer. 1970. Hierarchies of number theoretic functions, I. *Archive for Mathematical Logic* 13, 39–51. DOI:http://dx.doi.org/10.1007/BF01967649

Ernst W. Mayr. 1981. An algorithm for the general Petri net reachability problem. In *Proceedings of the 1981 STOC Conference*. ACM, Los Alamitos, CA, 238–246. DOI:http://dx.doi.org/10.1145/800076.802477

Ernst W. Mayr and Albert R. Meyer. 1981. The complexity of the finite containment problem for Petri nets. *Journal of the ACM* 28, 3, 561–576. DOI:http://dx.doi.org/10.1145/322261.322271

Kenneth McAloon. 1984. Petri nets and large finite sets. *Theoretical Computer Science* 32, 1–2, 173–183. DOI:http://dx.doi.org/10.1016/0304-3975(84)90029-X

Albert R. Meyer. 1975a. The inherent computational complexity of theories of ordered sets. In *Proceedings of the International Congress of Mathematicians*. 477–482. http://www.mathunion.org/ICM/ICM1974.2/Main/icm1974.2.0477.0482.ocr.pdf.

Albert R. Meyer. 1975b. Weak monadic second order theory of successor is not elementary-recursive. In *Logic Colloquium*. Lecture Notes in Mathematics, Vol. 453. Springer, 132–154. DOI:http://dx.doi.org/10.1007/BFb0064872

Albert R. Meyer and Dennis M. Ritchie. 1967. The complexity of loop programs. In *Proceedings of the 1967 ACM Conference*. ACM, New York, NY, 465–469. DOI:http://dx.doi.org/10.1145/800196.806014

Angelo Montanari, Gabriele Puppis, and Pietro Sala. 2010. Maximal decidable fragments of Halpern and Shoham's modal logic of intervals. In *Automata, Languages and Programming*. Lecture Notes in Computer Science, Vol. 6199. Springer, 345–356. DOI:http://dx.doi.org/10.1007/978-3-642-14162-1_29

Piergiorgio Odifreddi. 1999. *Classical Recursion Theory, Vol. II*. Studies in Logic and the Foundations of Mathematics, Vol. 143. Elsevier. DOI:http://dx:doi:org/10:1016/S0049-237X(99)80040-8

Eran Omri and Andreas Weiermann. 2009. Classifying the phase transition threshold for Ackermannian functions. *Annals of Pure and Applied Logic* 158, 3, 156–162. DOI:http://dx:doi:org/10:1016/j:apal:2007:02:004

Joël O. Ouaknine and James B. Worrell. 2006. On metric temporal logic and faulty Turing machines. In *Foundations of Software Science and Computation Structures*. Lecture Notes in Computer Science, Vol. 3921. Springer, 217–230. DOI:http://dx.doi.org/10.1007/11690634_15

Joël O. Ouaknine and James B. Worrell. 2007. On the decidability and complexity of metric temporal logic over finite words. *Logical Methods in Computer Science* 3, 1, Article No. 8. DOI:http://dx.doi.org/10.2168/LMCS-3(1:8)2007

Charles Rackoff. 1978. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science* 6, 2, 223–231. DOI:http://dx.doi.org/10.1016/0304-3975(78)90036-1

Robert W. Ritchie. 1963. Classes of predictably computable functions. *Transactions of the American Mathematical Society* 106, 1, 139–173. DOI:http://dx.doi.org/10.1090/S0002-9947-1963-0158822-2

Robert W. Ritchie. 1965. Classes of recursive functions based on Ackermann's function. *Pacific Journal of Mathematics* 15, 3, 1027–1044. DOI:http://dx.doi.org/10.2140/pjm.1965.15.1027

Fernando Rosa-Velardo. 2014. *Ordinal Recursive Complexity of Unordered Data Nets*. Technical Report TR-4-14. Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid. https://federwin.sip.ucm.es/sic/investigacion/publicaciones/pdfs/TR-04-14.pdf.

Harvey E. Rose. 1984. *Subrecursion: Functions and Hierarchies*. Oxford Logic Guides, Vol. 9. Clarendon Press.

Sylvain Schmitz and Philippe Schnoebelen. 2011. Multiply-recursive upper bounds with Higman's lemma. In *Automata, Languages and Programming*. Lecture Notes in Computer Science, Vol. 6756. Springer, 441–452. DOI:http://dx.doi.org/10.1007/978-3-642-22012-8_35

Sylvain Schmitz and Philippe Schnoebelen. 2012. Algorithmic Aspects of WQO Theory. Retrieved December 9, 2016, from http://cel.archives-ouvertes.fr/cel-00727025.

Sylvain Schmitz and Philippe Schnoebelen. 2013. The power of well-structured systems. In *CONCUR 2013—Concurrency Theory*. Lecture Notes in Computer Science, Vol. 8052. Springer, 5–24. DOI:http://dx.doi.org/10.1007/978-3-642-40184-8_2

Philippe Schnoebelen. 2002. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters* 83, 5, 251–261. DOI:http://dx.doi.org/10.1016/S0020-0190(01)00337-4

Philippe Schnoebelen. 2010. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In *Mathematical Foundations of Computer Science*. Lecture Notes in Computer Science, Vol. 6281. Springer, 616–628. DOI:http://dx.doi.org/10.1007/978-3-642-15155-2_54

Helmut Schwichtenberg. 1982. Complexity of normalization in the pure typed lambda-calculus. *Studies in Logic and the Foundations of Mathematics* 110, 453–457. DOI:http://dx.doi.org/10.1016/S0049-237X(09)70143-0

Helmut Schwichtenberg and Stanley S. Wainer. 2012. *Proofs and Computation*. Cambridge University Press.

Richard Statman. 1979. The typed λ-calculus is not elementary recursive. *Theoretical Computer Science* 9, 1, 73–81. DOI:http://dx.doi.org/10.1016/0304-3975(79)90007-0

Larry J. Stockmeyer and Albert R. Meyer. 1973. Word problems requiring exponential time. In *Proceedings of the 1973 STOC Conference*. ACM, New York, NY, 1–9. DOI:http://dx.doi.org/10.1145/800125.804029

Tony Tan. 2010. On pebble automata for data languages with decidable emptiness problem. *Journal of Computer and System Sciences* 76, 8, 778–791. DOI:http://dx:doi:org/10:1016/j:jcss:2010:03:004

Nikos Tzevelekos and Radu Grigore. 2013. History-register automata. In *Foundations of Software Science and Computation Structures*. Lecture Notes in Computer Science, Vol. 7794. Springer, 273–288. DOI:http://dx:doi:org/10:1007/978-3-642-37075-52

Alasdair Urquhart. 1984. The undecidability of entailment and relevant implication. *Journal of Symbolic Logic* 49, 4, 1059–1073. DOI:http://dx.doi.org/10.2307/2274261

Alasdair Urquhart. 1999. The complexity of decision procedures in relevance logic II. *Journal of Symbolic Logic* 64, 4, 1774–1802. DOI:http://dx.doi.org/10.2307/2586811

Sergei Vorobyov. 2004. The most nonelementary theory. *Information and Computation* 190, 2, 196–219. DOI: http://dx:doi:org/10:1016/j:ic:2004:02:002

Stanley S. Wainer. 1970. A classification of the ordinal recursive functions. *Archive for Mathematical Logic* 13, 3, 136–153. DOI: http://dx.doi.org/10.1007/BF01973619

Andreas Weiermann. 1994. Complexity bounds for some finite forms of Kruskal's theorem. *Journal of Symbolic Computation* 18, 5, 463–488. DOI: http://dx.doi.org/10.1006/jsco.1994.1059

Received September 2014; revised June 2015; accepted October 2015