Algorithmic Aspects of WQO Theory

S. Schmitz and Ph. Schnoebelen
LSV, ENS Cachan & CNRS, France

Lecture Notes

Work supported in part by ANR ReacHard.

FOREWORD

Well-quasi-orderings (wqos) (Kruskal, 1972) are a fundamental tool in logic and computer science. They provide termination arguments in a large number of decidability (or finiteness, regularity, …) results. In constraint solving, automated deduction, program analysis, and many more fields, wqo's usually appear under the guise of specific tools, like Dickson's Lemma (for tuples of integers), Higman's Lemma (for words and their subwords), Kruskal's Tree Theorem and its variants (for finite trees with embeddings), and recently the Robertson-Seymour Theorem (for graphs and their minors). What is not very well known is that wqo-based proofs have an algorithmic content.

The purpose of these notes is to provide an introduction to the complexity-theoretical aspects of wqos, to cover both upper bounds and lower bounds techniques, and provide several applications in logics (e.g. data logics, relevance logic), verification (prominently for well-structured transition systems), and rewriting. Because wqos are in such wide use, we believe this topic to be of relevance to a broad community with interests in complexity theory and decision procedures for logical theories. Our presentation is largely based on recent works that simplify previous results for upper bounds (Figueira et al., 2011; Schmitz and Schnoebelen, 2011) and lower bounds (Schnoebelen, 2010a; Haddad et al., 2012), but also contains some original material.

These lecture notes originate from an advanced course taught at the *24th European Summer School in Logic, Language and Information* (ESSLLI 2012) on August 6–10, 2012 in Opole, Poland, and also provide background material for Course 2.9.1 on the mathematical foundations of infinite transition systems taught at the *Parisian Master of Research in Computer Science* (MPRI). They follow their own logic rather than the ordering of these courses, and focus on subproblems that are treated in-depth:

- Chapter 1 presents how wqos can be used in algorithms,
- Chapter 2 proves complexity upper bounds for the use of Dickson's Lemma —this chapter is adapted chiefly from (Schmitz and Schnoebelen, 2011)—, and
- Chapter 3 details how to derive Ackermannian lower bounds on decision problems, drawing heavily on (Schnoebelen, 2010a).

Additionally, Appendix A proves many results on subrecursive hierarchies, which are typically skipped in papers and presentations, but needed for a working understanding of the results in chapters 2 and 3, and Appendix B lists known problems of enormous complexities.

CONTENTS

1

BASICS OF WQOS AND APPLICATIONS

1.1 Well Quasi Orderings

A relation $\leq$ over a set A is a *quasi ordering* (qo) iff it is reflexive and transitive. A quasi-ordering is a *partial ordering* (po) iff it also antisymmetric ($x \leq y$ and $y \leq x$ imply $x = y$). Any qo induces an equivalence relation $\equiv \overset{\text{def}}{=} \leq \cap \geq$, and gives rise to a canonical partial ordering between the equivalence classes, and to a *strict ordering* $< \overset{\text{def}}{=} \leq \smallsetminus \geq = \leq \smallsetminus \equiv$ between non-equivalent comparable elements. A qo is *linear* (aka *total*) iff any two elements are comparable ($\leq \cup \geq = A^2$). The main object of interest in this course is the following:

quasi ordering
partial ordering
strict ordering
linear ordering
total ordering

Definition 1.1 (wqo.1)**.** A *well quasi ordering* (wqo) $\leq$ over a set A is a qo such that every infinite sequence $x_0, x_1, x_2, \ldots$ over A contains an *increasing pair*: $\exists i < j$ s.t. $x_i \leq x_j$.

well quasi ordering
increasing pair

A *well partial ordering* is an antisymmetric wqo. By extension, a set along with an ordering $(A, \leq)$ is a *quasi order* (also noted *qo*) if $\leq$ is a quasi ordering over A (and similarly with po, wqo, etc.).

well partial ordering

Example 1.2 (Basic WQOs)**.** The set of nonnegative integers $(\mathbb{N}, \leq)$ is a wqo. Note that it is linear and partial. Given a set A, $(A, =)$ is always a po; it is a wqo iff A is finite.

See Exercise 1.1 for examples of qos and wqos.

1.1.1 Alternative Definitions

Definition 1.1 will be our main working definition for wqos, or rather its consequence that any sequence $x_0, x_1, \ldots$ over A with $x_i \not\leq x_j$ for all $i < j$ is necessarily finite. Nevertheless, wqos can be found under many guises, and enjoy several equivalent characterizations, e.g.

Definition 1.3 (wqo.2)**.** A qo $(A, \leq)$ is a wqo iff every infinite sequence $x_0, x_1, \ldots$ over A contains an *infinite* increasing subsequence: $\exists i_0 < i_1 < i_2 < \cdots$ s.t. $x_{i_1} \leq x_{i_1} \leq x_{i_2} \leq \cdots$.

Definition 1.4 (wqo.3)**.** A qo $(A, \leq)$ is a wqo iff

well-founded ordering

1. there are no infinite strictly decreasing sequences $x_0 > x_1 > x_2 > \cdots$ in A—i.e., $(A, \leq)$ is *well founded*—, and

antichain

2. there are no infinite sets $\{x_0, x_1, x_2, \ldots\}$ of mutually incomparable elements in A—i.e., $(A, \leq)$ has no infinite *antichains*.

The equivalence between these characterizations is quite useful; see Exercise 1.2 and the following:

Example 1.5. The qos $(\mathbb{Z}, \leq)$ and $(\mathbb{Q}, \leq)$ are not well-founded. The set of positive natural numbers $\mathbb{N}_+$ ordered by divisibility "|" has infinite antichains, e.g. the set of primes. The set of finite sequences Σ^* ordered lexicographically is not well-founded. None of these examples is wqo.

Ramsey Theorem

Regarding the equivalence of (wqo.1), (wqo.2, and (wqo.3), it is clear that (wqo.2) implies (wqo.1), which in turn implies (wqo.3). In order to prove that (wqo.3) implies (wqo.2), we use the Infinite Ramsey Theorem. Assume $(x_i)_{i\in\mathbb{N}}$ is an infinite sequence over $(A, \leq)$, which is a wqo according to (wqo.3). We consider the complete graph over $\mathbb{N}$ and color every edge $\{i, j\}$ (where $i < j$) with one of three colors. The edge is red when $x_i \leq x_j$ (up), it is blue when $x_i > x_j$ (strictly down), and it is green when $x_i \not\leq x_j \not\leq x_i$ (incomparable). The Infinite Ramsey Theorem shows that there exists an infinite subset $I \subseteq \mathbb{N}$ of indexes such that every edge $\{i, j\}$ over I has the same color. In effect, I yields an infinite subsequence $(x_i)_{i\in I}$ of $(x_i)_{i\in\mathbb{N}}$. If the subsequence has all its edges green, then we have exhibited an infinite antichain. If it has all edges blues, then we have exhibited an infinite strictly decreasing sequence. Since these are not allowed by (wqo.3), the single color for the edges of I must be red. Hence the original sequence has a infinite increasing subsequence: $(A, \leq)$ satisfies (wqo.2).

1.1.2 Upward-closed Subsets of wqos

upward-closure

upward-closed

downward-closure

downward-closed

Let $(A, \leq)$ be a quasi-ordering. The *upward-closure* $\uparrow B$ of some $B \subseteq A$ is defined as $\{x \in A \mid x \geq y \text{ for some } y \in B\}$. When $B = \uparrow B$, we say that B is *upward-closed*; the *downward-closure* $\downarrow B$ of B and the notion of *downward-closed* sets are defined symmetrically.

Definition 1.6 (wqo.4)**.** A qo $(A, \leq)$ is a wqo iff any increasing sequence $U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots$ of upward-closed subsets of A eventually stabilize, i.e., $\bigcup_{i\in\mathbb{N}} U_i$ is $U_k = U_{k+1} = U_{k+2} = \ldots$ for some k.

This characterization is sometimes expressed by saying that upward-closed sets satisfy the Ascending Chain Condition. See Exercise 1.3 for the equivalence of (wqo.4) with the other characterizations.

ascending chain condition|defpageidx

Upward- and downward-closed sets are important algorithmic tools: they are subsets of A that can be finitely represented and handled. The simplest generic representation is by minimal elements:

Lemma 1.7. *Let $(A, \leq)$ be a wqo. Any upward-closed $U \subseteq A$ can be written under the form $U = \uparrow\{x_1, \ldots, x_n\}$.*

(See Exercise 1.4 for a proof.) One can see how, using this representation, the comparisons of possibly infinite (but upward-closed) sets can be reduced to comparing finitely many elements.

The complement of a downward-closed set D is upward-closed. Hence downward-closed subsets of a wqo can be characterized by so-called *excluded minors*. That is, every downward-closed D is associated with a finite set $\{x_1, \ldots, x_n\}$ such that $x \in D$ iff $x_1 \not\leq x \wedge \cdots \wedge x_n \not\leq x$. Here the x_is are the excluded minors and D is "everything that does not have one of them as a minor."

excluded minor

1.1.3 Constructing wqos

There are several well-known ways of building new wqos out of simpler ones.

We already mention how the product $\prod_{i=} 1^m (A_i, \leq_i)$ of finitely many wqos is a wqo (see Exercise 1.2).

Lemma 1.8 (Dickson's Lemma). *Let $(A, \leq_A)$ and $(B, \leq_B)$ be two wqos. Then $(A \times B, \leq_{A \times B})$ is a wqo.*

Dickson's Lemma|defpageidx

There is a more general way of relating tuples of different lengths, which are then better understood as *finite sequences over A*. These can be well-quasi-ordered thanks to a fundamental result by G. Higman:

Lemma 1.9 (Higman's Lemma). *Let $(A, \leq)$ be a wqo. Then $(A^*, \leq_*)$ is a wqo.*

Higman's Lemma|defpageidx

See Exercise 1.7 for a proof; here the *sequence extension* A^* is the set of all finite sequences over A, and these sequences are ordered via the *subword embedding*:

sequence extension

subword embedding

$$(a_1 \cdots a_n) \leq_* (b_1 \cdots b_m) \overset{\text{def}}{\Leftrightarrow} \begin{cases} & \exists 1 \leq i_1 < i_2 < \cdots < i_n \leq m \\ \text{s.t.} & a_i \leq b_{i_1} \wedge \cdots \wedge a_n \leq b_{i_n}. \end{cases} \tag{1.1}$$

Example 1.10 (Subword ordering). We use ε to denote the empty sequence. Over $(\Sigma, =)$, where $\Sigma = \{a, b, c\}$ is a 3-letter alphabet and where different letters are incomparable, the word abb is a subword of $c\underline{ab}ca\underline{b}$, as witnessed by the underlined letters, and written $abb \leq_* cabcab$. On the other hand $bba \not\leq_* cabcab$. Over $(\mathbb{N}, \leq)$, examples are $\varepsilon \leq_* 4{\cdot}1{\cdot}3 \leq_* 1{\cdot}5{\cdot}0{\cdot}3{\cdot}3{\cdot}0{\cdot}0$ and $4{\cdot}1{\cdot}3 \not\leq_* 1{\cdot}5{\cdot}0{\cdot}3{\cdot}3{\cdot}0{\cdot}0$. Over $(\mathbb{N}^2, \leq_\times)$, one checks that $\binom{0}{1}{\cdot}\binom{2}{0}{\cdot}\binom{0}{2} \not\leq_* \binom{2}{0}{\cdot}\binom{0}{2}{\cdot}\binom{0}{2}{\cdot}\binom{2}{2}{\cdot}\binom{2}{0}{\cdot}\binom{0}{1}{\cdot}\binom{1}{0}$.

It is also possible to order finite and infinite subsets of a wqo, see Exercise 1.9.

Higman's original lemma was actually more general and handled homeomorphisms between finite trees with fixed arities, but this was extended by Kruskal to finite trees with variadic labels:

Kruskal's Tree Theorem|defpageidx

Theorem 1.11 (Kruskal's Tree Theorem)**.** *The set $T(A)$ of finite trees node-labeled from a wqo $(A, \leq)$ and partially ordered by homeomorphic embeddings is a wqo.*

(See Exercise 1.10 for the definition of homeomorphic embeddings and a proof of Kruskal's Theorem.)

Finally, a further generalization of Kruskal's Tree Theorem exists for graphs:

Graph Minor Theorem|defpageidx

Theorem 1.12 (Robertson and Seymour's Graph-Minor Theorem)**.** *The set of (finite undirected) graphs node-labeled from a wqo $(A, \leq)$ and ordered by the graph-minor relation is a wqo.*

1.2 Well-Structured Transition Systems

well-structured transition system

transition system

compatibility

In the field of algorithmic verification of program correctness, wqos figure prominently in *well-structured transition systems* (WSTS). These are *transition system* $\langle S, \to \rangle$, where S is a set of states and $\to \subseteq S \times S$ is a transition relation, further endowed with a wqo $\leq \subseteq S \times S$ that satisfies a *compatibility* condition:

$$s \to s' \wedge s \leq t \text{ implies } \exists t' \geq s', \, t \to t' \, . \qquad \text{(compatibility)}$$

Put together, this defines a WSTS $\mathcal{S} = \langle S, \to, \leq \rangle$. In other words, the states of $\mathcal{S}$ are well quasi ordered in a way such that "larger" states can simulate the behaviour of "smaller" states.

Several variants of the basic WSTS notion exist (backward compatibility, strict compatibility, ...) and we shall mention some of them in exercises 1.11 to 1.13.

vector addition system with states

Example 1.13. A d-dimensional *vector addition system with states* (VASS) is a finite-state system that manipulates d counters with only increment and decrement operations. Formally, it is a tuple $\mathcal{V} = \langle Q, \delta, q_0, \mathbf{x}_0 \rangle$ where Q is a finite set of states, $\delta \subseteq Q \times \mathbb{Z}^d \times Q$ is a finite set of translations, q_0 in Q is an initial state, and $\mathbf{x}_0$ in $\mathbb{N}^d$ describes the initial counter contents.

The semantics of a VASS define a transition system $\langle Q \times \mathbb{N}^d, \to \rangle$ where a transition $\to$ holds between two configurations $(q, \mathbf{x})$ and $(q', \mathbf{x}')$ if and only if there exists a translation $(q, \mathbf{a}, q')$ in δ with $\mathbf{x}' = \mathbf{x} + \mathbf{a}$; note that this transition requires $\mathbf{x} + \mathbf{a}$ non negative.

We can check that this transition system is a WSTS for the product ordering $\leq$ over $Q \times \mathbb{N}^d$, i.e. for $(q, \mathbf{x}) \leq (q', \mathbf{x}')$ iff $q = q'$ and $\mathbf{x}(j) = \mathbf{x}'(j)$ for all $j = 1, \ldots, d$. Indeed, whenever $(q, \mathbf{x}) \to (q', \mathbf{x}')$ and $\mathbf{x} \leq \mathbf{y}$, then there exists $(q, \mathbf{a}, q')$ in δ s.t. $\mathbf{x}' = \mathbf{x} + \mathbf{a}$, and $\mathbf{y}' = \mathbf{y} + \mathbf{a} \geq \mathbf{x} + \mathbf{a} \geq \mathbf{0}$, thus $(q, \mathbf{y}) \to (q', \mathbf{y}')$.

1.2.1 Termination

A transition system $\langle S, \to\rangle$ *terminates* from some state s_0 in S, if every transition sequence $s_0 \to s_1 \to \cdots$ is finite. This gives rise to the following, generally undecidable, problem:

termination

[Term] Termination
instance: A transition system $\langle S, \to\rangle$ and a state s_0 in S.
question: Does $\langle S, \to\rangle$ terminate from s_0?

In a WSTS, non-termination can be witnessed by increasing pairs in a finite run:

Lemma 1.14. *Let $\mathcal{S} = \langle S, \to, \leq\rangle$ be a WSTS and s_0 be a state in S. Then $\mathcal{S}$ has an infinite run starting from s_0 iff $\mathcal{S}$ has a run $s_0 \to s_1 \to \cdots \to s_j$ with $s_i \leq s_j$ for some $0 \leq i < j$.*

Proof. The direct implication follows from (wqo.1) applied to the infinite run $s_0 \to s_1 \to \cdots$. The converse implication follows from repeated applications of the compatibility condition to build an infinite run: first there exists $s_{j+1} \geq s_{i+1}$ s.t. $s_j \to s_{j+1}$, and so on and so forth. □

There is therefore a simple procedure to decide [Term], pending some effectiveness conditions: in a transition system $\langle S, \to\rangle$, define the *successor set*

successor set

$$Post(s) \stackrel{\text{def}}{=} \{s' \in S \mid s \to s'\} \tag{1.2}$$

of any s in S. A transition system is *image-finite* if $Post(s)$ is finite for all s in S. It is *Post-effective* if these elements can effectively be computed from s.

image-finite

Post-effective

Proposition 1.15 (Decidability of Termination for WSTSs)**.** *Let $\mathcal{S} = \langle S, \to, \leq\rangle$ be a WSTS and s_0 be a state in S. If $\mathcal{S}$ is image-finite, Post-effective, and $\leq$ is decidable, then termination of $\mathcal{S}$ from s_0 is also decidable.*

Proof. The algorithm consists of two semi-algorithms. The first one attempts to prove termination and builds a *reachability tree* starting from s_0; if $\mathcal{S}$ terminates from s_0, then every branch of this tree will be finite, and since $\mathcal{S}$ is image-finite this tree is also finitely branching, hence finite overall by Kőnig's Lemma. The second one attempts to prove non-termination, and looks nondeterministically for a finite witness matching Lemma 1.14. □

reachability tree

1.2.2 Coverability

The second decision problem we consider on WSTSs is also of great importance, as it encodes *safety* checking: can an error situation occur in the system?

[Cover] Coverability
instance: A transition system $\langle S, \to\rangle$, a qo $(S, \leq)$, and two states s, t in S.
question: Is t *coverable* from s, i.e. is there a run $s = s_0 \to s_1 \to \cdots \to s_n \geq t$?

coverability|defpageidx

In the particular case of a WSTS over state space $Q \times A$ for some finite set of control states Q and some wqo domain $(A, \leq_A)$, the Control-state Reachability Problem asks whether some input state q can be reached, regardless of the associated data value. This immediately reduces to coverability of the finitely many minimal elements of $\{q\} \times A$ for the product ordering over $Q \times A$, i.e. $(q, x) \leq (q', x')$ iff $q = q'$ and $x \leq_A x'$.

control-state reachability|defpageidx

The decidability of [Cover] for WSTS uses a *set-saturation method*, whose termination relies on (wqo.4). This particular algorithm is called the *backward coverability algorithm*, because it essentially computes all the states s' s.t. $s' \to^* t' \geq t$. For a set of states $I \subseteq S$, define its *predecessor set*

backward coverability

predecessor set

$$Pre(I) \overset{\text{def}}{=} \{s \in S \mid \exists s' \in I,\ s \to s'\} \ . \tag{1.3}$$

The backward analysis computes the limit $Pre^*(I)$ of the sequence

$$I = I_0 \subseteq I_1 \subseteq \cdots \text{ where } I_{n+1} \overset{\text{def}}{=} I_n \cup Pre(I_n) \ . \tag{1.4}$$

There is no reason for (1.4) to converge in general, but for WSTSs, this can be solved when I is upward-closed:

Lemma 1.16. *If $I \subseteq S$ is an upward-closed set of states, then $Pre(I)$ is upward-closed.*

Proof. Assume $s \in Pre(I)$. Then $s \to t$ for some $t \in I$. By compatibility of $\mathcal{S}$, if $s' \geq s$, then $s' \to t'$ for some $t' \geq t$. Thus $t' \in I$ and $s' \in Pre(I)$. □

A corollary is that sequence (1.4) stabilizes to $Pre^*(I)$ after a finite amount of time thanks to (wqo.4). The missing ingredient is an effectiveness one: a WSTS has *effective pred-basis* if there exists an algorithm accepting any state $s \in S$ and returning $pb(s)$, a finite basis of $\uparrow Pre(\uparrow\{s\})$.[1]

effective pred-basis

Proposition 1.17 (Decidability of Coverability for WSTSs)**.** *Let $\mathcal{S} = \langle S, \to, \leq \rangle$ be a WSTS and s, t be two states in S. If $\mathcal{S}$ has effective pred-basis and decidable $\leq$, then coverability of t from s in $\mathcal{S}$ is also decidable.*

Proof. Compute a finite basis B for $Pre^*(\uparrow\{t\})$ using sequence (1.4) and calls to *pb*, and test whether $s \geq b$ for some b in B. □

Exercises 1.12 and 1.13 present variants of this algorithm for different notions of compatibility.

[1] This definition is slightly more demanding than required, in order to accommodate for weaker notions of compatibility.

```
SIMPLE $(\mathrm{a},\mathrm{b})$
$\mathrm{c} \longleftarrow 1$
**while** $\mathrm{a} > 0 \wedge \mathrm{b} > 0$
    $l : \langle \mathrm{a},\mathrm{b},\mathrm{c}\rangle \longleftarrow \langle \mathrm{a}-1,\mathrm{b},2\mathrm{c}\rangle$
  **or**
    $r : \langle \mathrm{a},\mathrm{b},\mathrm{c}\rangle \longleftarrow \langle 2\mathrm{c},\mathrm{b}-1,1\rangle$
**end**
```

Figure 1.1: SIMPLE: A nondeterministic `while` program.

1.3 Examples of Applications

Let us present three applications of wqos in three different areas: one is quite generic and is concerned with proving program termination (Section 1.3.1). The other two are more specialized: we present applications to relevance logic (Section 1.3.2) and vector addition systems (Section 1.3.3).

1.3.1 Program Termination

Bad Sequences and Termination. Recall from Definition 1.1 that one of the characterizations for $(A, \leq)$ to be a wqo is that every infinite sequence $a_0, a_1, \ldots$ over A contains an *increasing pair* $a_{i_1} \leq a_{i_2}$ for some $i_1 < i_2$. We say that (finite or infinite) sequences with an increasing pair $a_{i_1} \leq a_{i_2}$ are *good* sequences, and call *bad* a sequence where no such increasing pair can be found. Therefore every infinite sequence over the wqo A is good, i.e., bad sequences over A are finite.

good sequence

bad sequence

In order to see how bad sequences are related to termination, consider the SIMPLE program presented in Figure 2.1. We can check that every run of SIMPLE terminates, this for any choice of initial values $\langle a_0, b_0\rangle$ of a and b. Indeed, we can consider any sequence

$$\langle a_0, b_0, c_0\rangle, \ldots, \langle a_j, b_j, c_j\rangle, \ldots \tag{1.5}$$

of successive configurations of SIMPLE, project away its third component, yielding a sequence

$$\langle a_0, b_0\rangle, \ldots, \langle a_j, b_j\rangle, \ldots\ , \tag{1.6}$$

and look at any factor $\langle a_{i_1}, b_{i_1}\rangle, \ldots, \langle a_{i_2}, b_{i_2}\rangle$ inside it:

- either only the first transition l is ever fired between steps i_1 and i_2, in which case $a_{i_2} < a_{i_1}$,
- or the second transition r was fired at least once, in which case $b_{i_2} < b_{i_1}$.

Thus $\langle a_{i_1}, b_{i_1}\rangle \not\leq \langle a_{i_2}, b_{i_2}\rangle$, which means that (1.6) is a bad sequence over $(\mathbb{N}^2, \leq)$, and is therefore a finite sequence. Consequently, (1.5) is also finite, which means that SIMPLE always terminates.

well-founded relation
Noetherian relation

Ranking Functions. Program termination proofs essentially establish that the program's transition relation R is *well-founded* (aka *Noetherian*), i.e. that there does not exist an infinite sequence of program configurations $x_0 \, R \, x_1 \, R \, x_2 \, R \cdots$. In the case of the integer program simple, this relation is included in $\mathbb{Z}^3 \times \mathbb{Z}^3$ and can be easily read from the program:

$$\begin{aligned}\langle a,b,c\rangle \; R \; \langle a',b',c'\rangle \text{ iff } a>0 \wedge b>0 \wedge ((a'=a-1 \wedge b'=b \wedge c'=2c) \\ \vee (a'=2c \wedge b'=b-1 \wedge c'=1))\;.\end{aligned} \tag{1.7}$$

ranking function

The classical, "monolithic" way of proving well-foundedness is to exhibit a *ranking function* ρ from the set of program configurations $x_0, x_1, \ldots$ into a well-founded order $(O, \leq)$ such that

$$R \subseteq \{(x_i, x_j) \mid \rho(x_i) > \rho(x_j)\}\;. \tag{1.8}$$

Then R is well-founded, otherwise we could exhibit an infinite decreasing sequence in $(O, \leq)$.

This is roughly what we did in (1.6), by projecting away the third component and using $\mathbb{N}^2$ as codomain; this does not satisfy (1.8) for the product ordering $(\mathbb{N}^2, \leq)$, but it does satisfy it for the lexicographic ordering $(\mathbb{N}^2, \leq_{\text{lex}})$. Equivalently, one could define $\rho\colon \mathbb{Z}^3 \to \omega^2$ by $\rho(a,b,c) = \omega \cdot b + a$ if $a, b \geq 0$ and $\rho(a,b,c) = 0$ otherwise.

However our argument with (1.6) was rather to use bad sequences: we rather require ρ to have a wqo $(A, \leq)$ as co-domain, and check that the *transitive closure* R^+ of R verifies

$$R^+ \subseteq \{(x_i, x_j) \mid \rho(x_i) \not\leq \rho(x_j)\} \tag{1.9}$$

instead of (1.8). Again, (1.9) proves R to be well-founded, as otherwise we could exhibit an infinite bad sequence in $(A, \leq)$.

Proving termination with these methods is done in two steps: first find a ranking function, then check that it yields termination through (1.8) for well-founded orders or (1.9) for wqos. As it turns out that finding an adequate ranking function is often the hardest part, this second option might be preferable.

disjunctive termination argument

Transition Invariants. A generalization of these schemes with a simpler search for ranking functions is provided by *disjunctive termination arguments*: in order to prove that R is well-founded, one rather exhibits a finite set of well-founded relations $T_1, \ldots, T_k$ and prove that

$$R^+ \subseteq T_1 \cup \cdots \cup T_k\;. \tag{1.10}$$

Each of the T_j, $1 \leq j \leq k$, is proved well-founded through a ranking function ρ_j, but these functions might be considerably simpler than a single, monolithic ranking function for R. In the case of simple, choosing

$$T_1 = \{(\langle a,b,c\rangle, \langle a',b',c'\rangle) \mid a > 0 \wedge a' < a\} \tag{1.11}$$

$$T_2 = \{(\langle a,b,c\rangle, \langle a',b',c'\rangle) \mid b > 0 \wedge b' < b\} \tag{1.12}$$

fits, their well-foundedness being immediate by projecting on the first (resp. second) component.

Let us prove the well-foundedness of R when each of the T_j is proven well-founded thanks to a ranking function ρ_j into some wqo $(A_j, \leq_j)$ (see Exercise 1.15 for a generic proof that only requires each T_j to be well-founded). Then with a sequence

$$x_0, x_1, \ldots \tag{1.13}$$

of program configurations one can associate the sequence of tuples

$$\langle \rho_1(x_0), \ldots, \rho_k(x_0)\rangle, \langle \rho_1(x_1), \ldots, \rho_k(x_1)\rangle, \ldots \tag{1.14}$$

in $A_1 \times \cdots \times A_k$, the latter being a wqo for the product ordering by Dickson's Lemma. Since for any indices $i_1 < i_2$, $(x_{i_1}, x_{i_2}) \in R^+$ is in some T_j for some $1 \leq j \leq k$, we have $\rho_j(x_{i_1}) \not\leq_j \rho_j(x_{i_2})$ by definition of a ranking function. Therefore the sequence of tuples is bad for the product ordering and thus finite, and the program terminates.

Different strategies can be used in practice to find a disjunctive termination invariant of the form (1.10). One that works well in the example of SIMPLE is to use the structure of the program relation R: if R can be decomposed as a union $R_1 \cup \cdots \cup R_k$, then applying rank function synthesis to each R_j, thereby obtaining a well-founded overapproximation $\mathrm{wf}(R_j) \supseteq R_j$, provides an initial candidate termination argument

$$\mathrm{wf}(R_1) \cup \cdots \cup \mathrm{wf}(R_k)\ . \tag{1.15}$$

Applying this idea to SIMPLE, we see that R in (1.7) is the union of

$$R_1 = \{(\langle a,b,c\rangle, \langle a',b',c'\rangle) \mid a > 0 \wedge b > 0 \wedge a' = a - 1 \wedge b' = b \wedge c' = 2c\} \tag{1.16}$$

$$R_2 = \{(\langle a,b,c\rangle, \langle a',b',c'\rangle) \mid a > 0 \wedge b > 0 \wedge a' = 2c \wedge b' = b - 1 \wedge c' = 1\}\ , \tag{1.17}$$

which can be overapproximated by T_1 and T_2 in (1.11) and (1.12).

It remains to check that (1.10) holds. If it does not, we can iterate the previous approximation technique, computing an overapproximation $\mathrm{wf}(\mathrm{wf}(R_{j_1}) \mathbin{;} R_{j_2})$ of the composition of R_{j_1} with R_{j_2}, then $\mathrm{wf}(\mathrm{wf}(\mathrm{wf}(R_{j_1}) \mathbin{;} R_{j_2}) \mathbin{;} R_{j_3})$ etc. until their union reaches a fixpoint or proves termination.

1.3.2 Relevance Logic

Relevance logics provide different semantics of implication, where a fact B is said to follow from A, written "$A \supset B$", only if A is actually *relevant* in the deduction of B. This excludes for instance $A \supset (B \supset A)$, $(A \wedge \neg A) \supset B$, etc.

We focus here on the implicative fragment $\mathbf{R}_\supset$ of relevance logic, which can be defined through a *substructural* sequent calculus in Gentzen's style. We use upper-case letters $A, B, C, \ldots$ for formulæ and $\alpha, \beta, \gamma, \ldots$ for possibly empty sequences of formulæ; a *sequent* is an expression $\alpha \vdash A$. The rules for $\mathbf{R}_{\supset}$ are:

$$\frac{}{A \vdash A}\,(\mathsf{Ax}) \qquad \frac{\alpha \vdash A \quad \beta A \vdash B}{\alpha\beta \vdash B}\,(\mathsf{Cut})$$

$$\frac{\alpha A B \beta \vdash C}{\alpha B A \alpha \vdash C}\,(\mathsf{Ex}) \qquad \frac{\alpha A A \vdash B}{\alpha A \vdash B}\,(\mathsf{Con})$$

$$\frac{\alpha \vdash A \quad \beta B \vdash C}{\alpha\beta(A \supset B) \vdash C}\,(\supset_{\mathsf{L}}) \qquad \frac{\alpha A \vdash B}{\alpha \vdash A \supset B}\,(\supset_{\mathsf{R}})$$

exchange
contraction
weakening

where (Ex) and (Con) are the *structural rules* of *exchange* and *contraction*. Note that the *weakening* rule (W) of propositional calculus is missing: otherwise we would have for instance the undesired derivation

$$\cfrac{\cfrac{\cfrac{\cfrac{}{A \vdash A}\,(\mathsf{Ax})}{AB \vdash A}\,(\mathsf{W})}{A \vdash B \supset A}\,(\supset_{\mathsf{R}})}{\vdash A \supset (B \supset A)}\,(\supset_{\mathsf{R}})$$

There are two important simplifications possible in this system: the first one is to redefine $\alpha, \beta, \ldots$ to be *multisets* of formulæ, which renders (Ex) useless; thus juxtaposition in (Ax–$\supset_{\mathsf{R}}$) should be interpreted as multiset union.

cut elimination

The second one is *cut elimination*, i.e. any sequent derivable in $\mathbf{R}_{\supset}$ has a derivation that does not use (Cut). This can be seen by the usual arguments, where cuts are progressively applied to “smaller” formulæ, thanks to a case analysis. For instance,

$$\cfrac{\cfrac{\cfrac{\vdots}{\gamma A \vdash B}}{\gamma \vdash A \supset B}\,(\supset_{\mathsf{R}}) \qquad \cfrac{\cfrac{\vdots}{\alpha \vdash A} \quad \cfrac{\vdots}{\beta B \vdash C}}{\alpha\beta(A \supset B) \vdash C}\,(\supset_{\mathsf{L}})}{\alpha\beta\gamma \vdash C}\,(\mathsf{Cut})$$

can be rewritten into

$$\cfrac{\cfrac{\cfrac{\vdots}{\gamma A \vdash B} \quad \cfrac{\vdots}{\alpha \vdash A}}{\alpha\gamma \vdash B}\,(\mathsf{Cut}) \qquad \cfrac{\vdots}{\beta B \vdash C}}{\alpha\beta\gamma \vdash C}\,(\mathsf{Cut})$$

subformula property

A consequence of cut elimination is that $\mathbf{R}_{\supset}$ enjoys the *subformula property*:

Lemma 1.18 (Subformula Property)**.** *If $\alpha \vdash A$ is a derivable sequent in $\mathbf{R}_{\supset}$, then there is a cut-free derivation of $\alpha \vdash A$ where every formula appearing in any sequent is a subformula of some formula of αA.*

The Decision Problem we are interested in solving is whether a formula A is a theorem of $\mathbf{R}_{\supset}$; it is readily generalized to whether a sequent $\alpha \vdash A$ is derivable using (Ax–$\supset_{\mathsf{R}}$).

[RI] Relevant Implication

relevant implication|defpageidx

instance: A formula A of $\mathbf{R}_{\supset}$.
question: Can the sequent $\vdash A$ be derived in $\mathbf{R}_{\supset}$?

A natural idea to pursue for deciding [RI] is to build a proof search tree with nodes labeled by sequents, and reversing rule applications from the root $\vdash A$ until only axioms are found as leaves. An issue with this idea is that the tree grows to an unbounded size, due in particular to contractions. See Exercise 1.18 for an algorithm that builds on this idea.

We reduce here [RI] to a WSTS coverability problem. Given A, we want to construct a WSTS $\mathcal{S} = \langle S, \rightarrow, \leq \rangle$, a target state t of S, and an initial state s in S s.t. t can be covered in $\mathcal{S}$ from s if and only if A is a theorem of $\mathbf{R}_{\supset}$.

Write $\text{Sub}(A)$ for its finite set of subformulæ. Then, by the Subformula Property, any sequent $\alpha \vdash B$ that derives A in a cut-free proof can be seen as an element of $\text{Seq}(A) \overset{\text{def}}{=} \mathbb{N}^{\text{Sub}(A)} \times \text{Sub}(A)$; we let

$$S \overset{\text{def}}{=} \mathcal{P}_f(\text{Seq}(A)) \tag{1.18}$$

be the set of finite subsets of $\text{Seq}(A)$.

Given a finite set s' of sequents, we say that

$$s' \rightarrow s' \cup \{\alpha \vdash B\} \tag{1.19}$$

if some rule among (Ax–$\supset_{\mathsf{R}}$) ((Cut) excepted) can employ some premise(s) in s' to derive the sequent $\alpha \vdash B$.

For a multiset α, define its *multiset support* $\sigma(\alpha)$ as its underlying set of elements $\sigma(\alpha) = \{B \mid \alpha(B) > 0\}$. We define the *contraction* qo $\ll$ over sequents by $\alpha \vdash B \ll \alpha' \vdash B'$ iff $\alpha \vdash B$ can be obtained from $\alpha' \vdash B'$ by some finite, possibly null, number of contractions. Over $\text{Seq}(A)$, this is equivalent to having $\alpha \leq \alpha'$ (for the product ordering over $\mathbb{N}^{\text{Sub}(A)}$), $\sigma(\alpha) = \sigma(\alpha')$, and $B = B'$: $\ll$ over $\text{Seq}(A)$ is thus defined as a product ordering between the three wqos $(\mathbb{N}^{\text{Sub}(A)}, \leq)$, $(\mathcal{P}(\text{Sub}(A)), =)$, and $(\text{Sub}(A), =)$, and therefore by Dickson's Lemma:

multiset support

contraction ordering

Lemma 1.19 (Kripke's Lemma). *The qo* $(\text{Seq}(A), \ll)$ *is a wqo.*

Then, by Exercise 1.9, the qo $(S, \leq)$, where $\leq$ is *Hoare's ordering* applied to $\ll$, is a wqo, and we easily see that $\mathcal{S} = \langle S, \rightarrow, \leq \rangle$ is a WSTS with effective pred-basis and a decidable ordering (see Exercise 1.17), thus the coverability problem for

$$s \overset{\text{def}}{=} \{B \vdash B \mid B \in \text{Sub}(A)\} \qquad\qquad t \overset{\text{def}}{=} \{\vdash A\} \tag{1.20}$$

is decidable by Proposition 1.17.

It remains to check that coverability of $\langle \mathcal{S}, s, t\rangle$ is indeed equivalent to derivability of $\vdash A$. Clearly, if $s = s_0 \to s_1 \to \cdots \to s_n$, then any sequent appearing in any s_i along this run is derivable in $\mathbf{R}_\supset$, and if $t \leq s_n$—which is equivalent to the existence of a sequent $\alpha \vdash B$ in s_n, s.t. $\vdash A \ll \alpha \vdash B$, which by definition of $\ll$ is equivalent to $\sigma(\alpha) = \emptyset$ and $A = B$, i.e. to $\vdash A$ being in s_n—, then A is indeed a theorem of $\mathbf{R}_\supset$. Conversely, if $\vdash A$ is derivable by a cut-free proof in $\mathbf{R}_\supset$, then we can reconstruct a run in $\mathcal{S}$ by a breadth-first visit starting from the leaves of the proof tree, which starts from the set $s_0 \subseteq s$ of leaves of the proof tree, applies $\to$ along the rules (Ax–$\supset_\mathrm{R}$) of the proof tree, and ends at the root of the proof tree with a set s' of sequents that includes $\vdash A$. Finally, by compatibility of $\mathcal{S}$, since $s_0 \leq s$, there exists a run $s \to \cdots \to s''$ such that $t = \{\vdash A\} \subseteq s' \leq s''$, proving that t is indeed coverable from s in $\mathcal{S}$.

1.3.3 Karp & Miller Trees

vector addition system|defpageidx

Vector Addition Systems (VAS) are systems where d counters evolve by non-deterministically applying d-dimensional translations from a fixed set, i.e. they are single-state VASSs. They can be seen as an abstract presentation of Petri nets, and are thus widely used to model concurrent systems, reactive systems with resources, etc. They also provide an example of systems for which WSTS algorithms work especially well.

Formally, a d-dimensional VAS is a pair $\mathcal{V} = \langle \mathbf{x}_0, \mathbf{A}\rangle$ where $\mathbf{x}_0$ is an initial configuration in $\mathbb{N}^d$ and $\mathbf{A}$ is a finite set of translations in $\mathbb{Z}^d$. A translation $\mathbf{a}$ in $\mathbf{A}$ can be applied to a configuration $\mathbf{x}$ in $\mathbb{N}^d$ if $\mathbf{x}' = \mathbf{x} + \mathbf{a}$ is in $\mathbb{N}^d$, i.e. non-negative. The resulting configuration is then $\mathbf{x}'$, and we write $\mathbf{x} \xrightarrow{\mathbf{a}}_\mathcal{V} \mathbf{x}'$. A d-dimensional VAS $\mathcal{V}$ clearly defines a WSTS $\langle \mathbb{N}^d, \to, \leq\rangle$ where $\to \stackrel{\text{def}}{=} \bigcup_{\mathbf{a}\in\mathbf{A}} \xrightarrow{\mathbf{a}}_\mathcal{V}$ and $\leq$ is the product ordering over $\mathbb{N}^d$. A configuration $\mathbf{x}$ is reachable, denoted $\mathbf{x} \in \mathrm{Reach}(\mathcal{V})$, if there exists a sequence

$$\mathbf{x}_0 \xrightarrow{\mathbf{a}_1} \mathbf{x}_1 \xrightarrow{\mathbf{a}_2} \mathbf{x}_2 \xrightarrow{\mathbf{a}_3} \cdots \xrightarrow{\mathbf{a}_n} \mathbf{x}_n = \mathbf{x}\,. \tag{1.21}$$

That reachability is decidable for VASs is a major result of computer science but we are concerned here with computing a *covering* of the reachability set.

Coverings. In order to define what is a “covering”, we consider the completion $\mathbb{N}_\omega \stackrel{\text{def}}{=} \mathbb{N} \cup \{\omega\}$ of $\mathbb{N}$ and equip it with the obvious ordering. Tuples $\mathbf{y} \in \mathbb{N}_\omega^d$, called *$\omega$-markings*, are ordered with the product ordering. Note that $\mathbb{N}_\omega$ is a wqo, and thus $\mathbb{N}_\omega^d$ as well by Dickson’s Lemma.

While ω-markings are not proper configurations, it is convenient to extend the notion of steps and write $\mathbf{y} \xrightarrow{\mathbf{a}} \mathbf{y}'$ when $\mathbf{y}' = \mathbf{y} + \mathbf{a}$ (assuming $n + \omega = \omega + n = \omega$ for all n).

covering

Definition 1.20 (Covering)**.** Let $\mathcal{V}$ be a d-dimensional VAS. A set $C \subseteq \mathbb{N}_\omega^d$ of ω-markings is a *covering* for $\mathcal{V}$ if

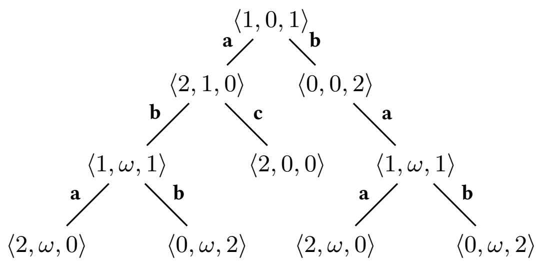

Figure 1.2: A Karp & Miller tree constructed for the VAS $\langle\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}, \langle 1, 0, 1\rangle\rangle$ with translations $\mathbf{a} = \langle 1, 1, -1\rangle$, $\mathbf{b} = \langle -1, 0, 1\rangle$, and $\mathbf{c} = \langle 0, -1, 0\rangle$.

1. for any $\mathbf{x} \in \text{Reach}(\mathcal{V})$, C contains some $\mathbf{y}$ with $\mathbf{x} \leq \mathbf{y}$, and

2. any $\mathbf{y} \in C$ is in the *adherence* of the reachability set, i.e. $\mathbf{y} = \lim_{i=1,2,\ldots} \mathbf{x}_i$ for some infinite sequence of configurations $\mathbf{x}_1, \mathbf{x}_2, \ldots$ in $\text{Reach}(\mathcal{V})$.

Hence a covering is a rather precise approximation of the reachability set (precisely, the adherence of its downward-closure). A fundamental result is that *finite* coverings always exist and are computable. This entails several decidability results, e.g. whether a counter value remains bounded throughout all the possible runs.

The Karp & Miller Tree constructs a particular covering of $\mathcal{V}$. Formally, this tree has nodes labeled with ω-markings in $\mathbb{N}_\omega^d$ and edges labeled with translations in $\mathbf{A}$. The root s_0 is labeled with $\mathbf{x}_0$ and the tree is grown in the following way: Karp & Miller tree

Assume a node s of the tree is labeled with some $\mathbf{y}$ and let $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_n$ be the sequence of labels on the path from the root s_0 to s, with $\mathbf{x}_0 = \mathbf{y}_0$ and $\mathbf{y}_n = \mathbf{y}$. For any translation $\mathbf{a} \in \mathbf{A}$ such that there is a step $\mathbf{y} \xrightarrow{\mathbf{a}} \mathbf{y}'$, we consider whether to grow the tree by adding a child node s' to s with a $\mathbf{a}$-labeled edge from s to s':

1. If $\mathbf{y}' \leq \mathbf{y}_i$ for one of the $\mathbf{y}_i$'s on the path from s_0 to s, we do not add s' (the branch ends).

2. Otherwise, if $\mathbf{y}' > \mathbf{y}_i$ for some $i = 0, \ldots, n$, we build $\mathbf{y}''$ from $\mathbf{y}'$ by setting, for all $j = 1, \ldots, d$,

$$\mathbf{y}''(j) \overset{\text{def}}{=} \begin{cases} \omega & \text{if } \mathbf{y}'(j) > \mathbf{y}_i(j) \\ \mathbf{y}'(j) & \text{otherwise.} \end{cases} \tag{1.22}$$

   Formally, $\mathbf{y}''$ can be thought as "$\mathbf{y}_i + \omega \cdot (\mathbf{y}' - \mathbf{y}_i)$." We add s', the edge from s to s', and we label s' with $\mathbf{y}''$.

3. Otherwise, $\mathbf{y}'$ is not comparable with any $\mathbf{y}_i$: we simply add the edge and label s' with $\mathbf{y}'$.

See Figure 1.2 for an example of tree constructed by this procedure.

Theorem 1.21. *The above algorithm terminates and the set of labels in the Karp & Miller tree is a covering for $\mathcal{V}$.*

Proof of termination. First observe that the tree is finitely branching (a node has at most $|\mathbf{A}|$ children), thus by Kőnig's Lemma the tree can only be infinite by having an infinite branch. Assume, for the sake of contradiction, that there is such an infinite branch labeled by some $\mathbf{y}_0, \mathbf{y}_1, \ldots$ By (wqo.2) applied to $\mathbb{N}^d_\omega$, we can exhibit an infinite subsequence $\mathbf{y}_{i_0} \leq \mathbf{y}_{i_1} \leq \cdots$ with $i_0 < i_1 < \cdots$. Any successive pair $\mathbf{y}_{i_k} \leq \mathbf{y}_{i_{k+1}}$ requires $\mathbf{y}_{i_{k+1}}$ to be inserted at step 2 of the algorithm, hence $\mathbf{y}_{i_{k+1}}$ has more ω-components than $\mathbf{y}_{i_k}$. Finally, since an ω-marking has at most d ω-components, this extracted sequence is of length at most $d+1$ and cannot be infinite. □

We leave the second part of the proof as Exercise 1.20.

Exercises

Exercise 1.1 (Examples of qos)**.** Among the following quasi orders, which ones are partial orders? Are they total? Well-founded? Wqo?

(1) the natural numbers $(\mathbb{N}, \leq)$, the integers $(\mathbb{Z}, \leq)$, the positive reals $(\mathbb{R}_+, \leq)$;

(2) the natural numbers $(\mathbb{N}, |)$ where $a \mid b$ means that a divides b;

prefix ordering

lexicographic ordering

(3) given a linearly ordered finite alphabet Σ, the set of finite sequences Σ^* with *prefix ordering* $\leq_{\text{pref}}$ or *lexicographic ordering* $\leq_{\text{lex}}$;

(4) $(\mathcal{P}(\mathbb{N}), \subseteq)$ the subsets of $\mathbb{N}$ ordered with inclusion;

Smyth's ordering

(5) $(\mathcal{P}(\mathbb{N}), \sqsubseteq_S)$ where we use *Smyth's ordering*: $U \sqsubseteq_S V \overset{\text{def}}{\Leftrightarrow} \forall m \in V, \exists n \in U, n \leq m$;

(6) $(\mathcal{P}_f(\mathbb{N}), \subseteq)$ and $(\mathcal{P}_f(\mathbb{N}), \sqsubseteq_S)$ where we restrict to finite subsets.

Exercise 1.2 (Generalized Dickson's Lemma)**.** If $(A_i, \leq_i)_{i=1,\ldots,m}$ are m quasi-orderings, their *product* is $\prod_{=1}^m (A_i, \leq_i) = (\mathbf{A}, \leq_\times)$ given by $\mathbf{A} = A_1 \times \cdots \times A_m$, and

$$\langle x_1, \ldots, x_m \rangle \leq_\times \langle x'_1, \ldots, x'_m \rangle \overset{\text{def}}{\Leftrightarrow} x_1 \leq_1 x'_1 \wedge \cdots \wedge x_m \leq_m x'_m .$$

(1) Show that $\prod_{=1}^m (A_i, \leq_i)$ is well-founded when each $(A_i, \leq_i)$ is.

(2) Show that $\prod_{=1}^m (A_i, \leq_i)$ is a wqo when each $(A_i, \leq_i)$ is.

monomial

(3) Show that the set of *monomials* $x_1^{a_1} \cdot x_2^{a_2} \cdots x_d^{a_d}$ over the set of variables $\{x_1, \ldots, x_d\}$ where the a_i's are natural numbers is well quasi ordered by the divisibility ordering.

Exercise 1.3 (Ascending Chain Condition)**.** Show that (wqo.4) is equivalent with the other definition(s) of wqos.

Exercise 1.4 (Finite Basis Property)**.**

(1) Prove Lemma 1.7: any upward-closed subset U of a wqo $(A, \leq)$ can be written under the form $U = \uparrow\{x_1, \ldots, x_n\}$.

(2) (wqo.5) Prove that a qo $(A, \leq)$ is a wqo iff every non-empty subset U of A contains at least one, and at most finitely many (up to equivalence), minimal elements.

Exercise 1.5 (Linear WQOs)**.**

(1) Prove that a linear ordering is a wqo iff it is well-founded.

(2) (wqo.6) Prove that a qo is a wqo iff all its linearizations are well-founded, where a *linearization* of $(A, \leq)$ is any linear qo $(A, \preceq)$ with same support and such that $x \leq y$ implies $x \preceq y$.

linearization

Exercise 1.6 ($\mathbb{Z}^k, \leq_{\text{sparse}}$)**.** We consider the *sparser-than ordering*. Assume $\mathbf{a} = (a_1, \ldots, a_k)$ and $\mathbf{b} = (b_1, \ldots, b_k)$ are two tuples in $\mathbb{Z}^k$, then

sparser-than ordering

$$\mathbf{a} \leq_{\text{sparse}} \mathbf{b} \overset{\text{def}}{\Leftrightarrow} \forall i, j \in \{1, \ldots, k\} : \big(a_i \leq a_j \text{ iff } b_i \leq b_j\big) \text{ and } \big(|a_i - a_j| \leq |b_i - b_j|\big) .$$

Show that $(\mathbb{Z}^k, \leq_{\text{sparse}})$ is a wqo.

Exercise 1.7 (Higman's Lemma)**.** Recall that for a qo $(A, \leq)$, the set A^* of finite sequences ("words") over A can be ordered by the subword embedding $\leq_*$ defined with (1.1). We shall prove Higman's Lemma: $(A^*, \leq_*)$ is wqo iff $(A, \leq)$ is.

⋆

(1) Show that $(A^*, \leq_*)$ is well-founded if $(A, \leq)$ is.

(2) Assume, by way of contradiction, that $(A, \leq)$ is wqo but $(A^*, \leq_*)$ is not. Then there exist some infinite bad sequences over A^*, i.e., sequences of the form $w_0, w_1, w_2, \ldots$ where $w_i \not\leq_* w_j$ for all $i, j \in \mathbb{N}$ s.t. $i < j$.

Consider all words that can start such an infinite bad sequence, pick a shortest one among them, and call it v_0. Consider now all infinite bad sequences that start with v_0 and, among all words that can appear after the initial v_0, pick a shortest one and call it v_1. Repeat the process and at stage k pick v_k as one among the shortest words that can appear after $v_0, \ldots, v_{k-1}$ inside an infinite bad sequence. Show that this process can be continued forever and that is generates an *infinite* sequence $S = v_0, v_1, \ldots$

(3) Show that S itself is a bad sequence.

(4) We now write every v_i under the form $v_i = a_i u_i$ where $a_i \in A$ is the first "letter" of v_i and u_i is the first strict suffix (this is possible since an infinite bad sequence cannot contain the empty word). We now pick an infinite increasing sequence $a_{k_0} \leq a_{k_1} \leq a_{k_2} \leq \cdots$ from $(a_i)_{i \in \mathbb{N}}$ (possible since A is wqo) and we write S' for the sequence $u_{k_0}, u_{k_1}, \ldots$ of corresponding suffixes. Show that if S' is good—i.e., contains an increasing pair—, then S is good too.

(5) We deduce that S' must be an infinite bad sequence over A^*. Use this to derive a contradiction (hint: recall the definition of v_{i_0}).

At this point we conclude that our assumption "A is wqo but A^* is not" was contradictory, proving Higman's Lemma.

Exercise 1.8 (Higman's Lemma for ω-sequences?)**.** Let $(A, \leq)$ be a wqo. For two infinite words $v = (x_i)_{i\in\mathbb{N}}$ and $w = (y_i)_{i\in\mathbb{N}}$ in A^ω, we let

$$v \leq_\omega w \overset{\text{def}}{\Leftrightarrow} \begin{cases} & \text{there are some indexes } n_0 < n_1 < n_2 < \cdots \\ \text{s.t.} & x_i \leq y_{n_i} \text{ for all } i \in \mathbb{N}. \end{cases}$$

Show that $(A^\omega, \leq_\omega)$ is a wqo when $(A, \leq)$ is a *linear* wqo.[2]

Exercise 1.9 (Ordering Powersets)**.** Recall from Exercise 1.1 the definition of *Smyth's ordering* on the powerset $\mathcal{P}(A)$: if $(A, \leq)$ is a qo and $U, V \subseteq A$ we let:

Hoare ordering

Egli-Milner ordering

$$U \sqsubseteq_{\mathrm{S}} V \overset{\text{def}}{\Leftrightarrow} \forall m \in V, \exists n \in U, n \leq m\ . \tag{$*$}$$

There also exists the (more natural) *Hoare ordering* (also called *Egli-Milner ordering*):

$$U \sqsubseteq_{\mathrm{H}} V \overset{\text{def}}{\Leftrightarrow} \forall n \in U, \exists m \in V, n \leq m\ . \tag{$\dagger$}$$

(1) What are the equivalences generated by $\sqsubseteq_{\mathrm{S}}$ and by $\sqsubseteq_{\mathrm{H}}$?

(2) Express $\sqsubseteq_{\mathrm{S}}$ in terms of $\sqsubseteq_{\mathrm{H}}$ (and reciprocally), using set-theoretic operations like upward-closure, intersection, etc.

(3) Prove the following characterization of wqo's:

$$\text{A qo } (A, \leq) \text{ is wqo if, and only if, } (\mathcal{P}(A), \sqsubseteq_{\mathrm{H}}) \text{ is well-founded.} \tag{wqo.7}$$

(4) Further show that $(\mathcal{P}_f(A), \sqsubseteq_{\mathrm{H}})$ is wqo iff $(A, \leq)$ is wqo—recall that $\mathcal{P}_f(A)$ only contains the *finite* subsets of A.

Exercise 1.10 (Kruskal's Tree Theorem)**.** For a qo $(A, \leq)$, we write $T(A)$ for the set of finite trees node-labeled by A. Formally, $T(A) = \{t, u, v, \ldots\}$ is the smallest set such that if $a \in A$, $m \in \mathbb{N}$ and $t_1, \ldots, t_m \in T(A)$ then the tree with root labeled by a and subtrees $t_1, \ldots, t_m$, denoted $a(t_1, \ldots, t_m)$, is in $T(A)$. We order $T(A)$ with $\leq_T$, the homeomorphic embedding that extends $\leq$. The definition of $u \leq_T t$ is by induction on the structure of t, with

$$a(u_1, \ldots, u_m) \leq_T b(t_1, \ldots, t_k) \overset{\text{def}}{\Leftrightarrow} \begin{cases} a \leq b \text{ and } (u_1, \ldots, u_m) \leq_{T,*} (t_1, \ldots, t_k) \\ \text{or } \exists i \in \{1, \ldots, k\} : a(u_1, \ldots, u_m) \leq_T t_i\ . \end{cases} \tag{$\ddagger$}$$

Here $\leq_{T,*}$ denotes the sequence extension of $\leq_T$.

(1) We now assume that $(A, \leq)$ is a wqo and prove that $(T(A), \leq_T)$ is a wqo too. For this we assume, by way of contradiction, that $(T(A), \leq_T)$ is not wqo. We proceed as in the proof of Higman's Lemma (Exercise 1.7) and construct a "minimal infinite bad

[2]This does not extend to arbitrary wqos, see (Rado, 1954) or (Jančar, 1999) for a characterization of the qos A with $(A^\omega, \leq_\omega)$ wqo.

sequence" $S = t_0, t_1, t_2, \dots$ where t_0 is a smallest tree that can be used to start an infinite bad sequence, and at stage k, t_k is a smallest tree that can continue an infinite bad sequence starting with $t_0, \dots, t_{k-1}$. By construction S is infinite and is bad.

Let us now write every t_i in S under the form $t_i = a_i(u_{i,1}, \dots, u_{i,m_i})$ and collect all the immediate subtrees in $U \overset{\text{def}}{=} \{t_{i,j} \mid i \in \mathbb{N} \wedge 1 \leq j \leq m_i\}$. Show that $(U, \leq_T)$ is wqo.

(2) Derive a contradiction (hint: use Higman's Lemma on U).

At this point we conclude that our assumptions "A is wqo but $T(A)$ is not" was contradictory, proving Kruskal's Theorem.

Well Structured Transition Systems

Exercise 1.11 (Transitive Compatibility)**.** We relax in this exercise (compatibility) to a weaker notion of compatibility, but show that [Term] remains decidable in this setting. Consider the following replacement for (compatibility):

transitive compatibility

$$s \to s' \wedge s \geq t \text{ implies } s' \geq t \vee \exists t' \leq s', t \to^+ t' , \tag{tc}$$

where $\to^+$ is the transitive closure of $\to$.

Show that, if $\mathcal{S} = \langle S, \to, \leq \rangle$ is a WSTS for (tc), which is image-finite and *Post*-effective and has decidable $\leq$, then one can decide whether $\mathcal{S}$ terminates from some state s_0 in S.

Exercise 1.12 (Reflexive Transitive Compatibility)**.** Let us relax (compatibility) to:

reflexive transitive compatibility

$$s \to s' \wedge s \geq t \text{ implies } s' \geq t \vee \exists t' \leq s', t \to^* t' , \tag{rtc}$$

where $\to^*$ is the reflexive transitive closure of $\to$. We assume throughout this exercise that $\mathcal{S} = \langle S, \to, \leq \rangle$ is a WSTS under (rtc).

(1) Show that, if I is upward-closed, then $\mathit{Pre}^*(I)$ is also upward-closed. Does Lemma 1.16 still hold?

(2) Let K_0 be a finite basis of I. Lift *pb* to operate on finite sets. The sequence

$$K_0 \subseteq K_1 \subseteq \cdots \text{ where } K_{n+1} \overset{\text{def}}{=} K_n \cup \mathit{pb}(K_n) \tag{§}$$

converges by (wqo.4) after finitely many steps to some finite set K. Show that $\uparrow K = \uparrow \bigcup_{i \in \mathbb{N}} K_i$.

(3) Show that $\uparrow K = \mathit{Pre}^*(I)$.

(4) Conclude that [Cover] is decidable for WSTS with (rtc), effective pred-basis, and decidable $\leq$.

Exercise 1.13 (Downward WSTSs)**.** Let $\langle S, \to \rangle$ be a transition system and $(S, \leq)$ be a wqo. The definition of compatibility is also known as "upward-compatibility", by contrast with its dual *reflexive downward compatibility* :

reflexive downward compatibility

$$s \to s' \wedge s \geq t \text{ implies } s' \geq t \vee \exists t' \leq s',\, t \to t' \;. \tag{rdc}$$

downward WSTS

that defines a *downward WSTS* $\mathcal{S} = \langle S, \to, \leq \rangle$.

Show that the following problem is decidable for image-finite, *Post*-effective downward WSTSs with decidable $\leq$:

[SCover] Sub-Coverability
instance: A transition system $\langle S, \to \rangle$, a qo $(S, \leq)$, and two states s, t in S.
question: Is there a run $s = s_0 \to s_1 \to \cdots \to s_n \leq t$?

Program Termination

Exercise 1.14. Show that the weaker condition

$$R \subseteq T_1 \cup \cdots \cup T_k \tag{¶}$$

with each T_j is well-founded does not imply R well-founded.

Exercise 1.15 (Disjunctive Termination Arguments)**.** Assume that a binary relation R verifies (1.10) on page 8, where each T_j is well-founded. Prove using the Infinite Ramsey Theorem that R is well-founded.

Relevance Logic

Exercise 1.16 (Cut Elimination & Subformula Property)**.** Prove Lemma 1.18.

Exercise 1.17 (A WSTS for Relevant Implication)**.** Prove that $\mathcal{S}$ defined by equations (1.18) and (1.19) is a WSTS with effective pred-basis and decidable ordering.

⋆ **Exercise 1.18** (Proof Search for Relevant Implication)**.** The purpose of this exercise is to find an alternative algorithm for [RI]. The key idea in this algorithm is to remove (Con) from $\mathbf{R}_\supset$ and apply contractions only when needed, i.e. modify the rules ($\supset_\mathsf{L}$) and ($\supset_\mathsf{R}$) to contract their conclusion, but only inassomuch as could not be obtained by first contracting their premises. Doing so we define an alternative proof system $\mathbf{R}'_\supset$ that includes the unmodified (Ax) and ($\supset_\mathsf{R}$), and a modified version of ($\supset_\mathsf{L}$):

$$\frac{\alpha \vdash A \quad \beta B \vdash C}{\gamma \vdash C}\;(\supset'_\mathsf{L})$$

where $\gamma \vdash C \ll \alpha\beta(A \supset B) \vdash C$ is such that, for all formulæ D, $\gamma(D) \geq \alpha(D) + \beta(D) - 1$.

(1) Show how any derivation of a sequent $\alpha \vdash B$ in $\mathbf{R}_\supset \cup \mathbf{R}'_\supset$ can be transformed into a derivation in $\mathbf{R}'_\supset$ of no larger height.

(2) Deduce that $\mathbf{R}'_\supset$ and $\mathbf{R}_\supset$ derive the same sequents.

(3) Deduce that, if $\alpha \vdash B \ll \alpha' \vdash B'$ and $\alpha' \vdash B'$ has a derivation of height n in $\mathbf{R}'_\supset$, then $\alpha \vdash B$ has a derivation of height at most n in $\mathbf{R}'_\supset$.

(4) We work now in the modified calculus $\mathbf{R}'_{\supset}$. We say that a derivation in $\mathbf{R}'_{\supset}$ is *irredundant* if, by following any branch starting from the root to the leaves, we never first meet $\alpha \vdash B$ and later $\alpha' \vdash B'$ with $\alpha \vdash B \ll \alpha' \vdash B'$. Show that [RI] is decidable by proof search using Kőnig's Lemma and Kripke's Lemma.

Karp & Miller Trees

Exercise 1.19. Show that $\mathbb{N}_\omega$ is a wqo.

Exercise 1.20 (Covering)**.** The aim of this exercise is to complete the proof of Theorem 1.21 and show that the set of labels $C \subseteq \mathbb{N}_\omega^d$ of the Karp & Miller tree T forms a covering according to Definition 1.20. ⋆⋆

(1) Let $\text{neg}(\mathbf{a})$ be the vector in $\mathbb{N}^d$ defined by

$$\text{neg}(\mathbf{a})(j) = \begin{cases} -\mathbf{a}(j) & \text{if } \mathbf{a}(j) \leq 0 \\ 0 & \text{otherwise} \end{cases} \tag{$\|$}$$

for $\mathbf{a}$ in $\mathbb{Z}^d$ and j in $\{1, \ldots, d\}$. The *threshold* $\Theta(u)$ of a transition sequence u in $\mathbf{A}^*$ is the minimal configuration $\mathbf{x}$ in $\mathbb{N}^d$ s.t. u is enabled from $\mathbf{x}$, i.e. there exists $\mathbf{x}'$ s.t. $\mathbf{x} \xrightarrow{u}_{\mathcal{V}} \mathbf{x}'$. Show how to compute $\Theta(u)$. Show that $\Theta(uv) \leq \Theta(u) + \Theta(v)$ for all u, v in $\mathbf{A}^*$. threshold

(2) In order to prove that C satisfies Definition 1.20.1, we will prove a stronger statement. For an ω-marking $\mathbf{y}$ in $\mathbb{N}_\omega^d$, first define

$$\Omega(\mathbf{y}) \stackrel{\text{def}}{=} \{j = 1, \ldots, d \mid \mathbf{y}(j) = \omega\} \tag{$**$}$$

the set of ω-components of $\mathbf{y}$, and

$$\overline{\Omega}(\mathbf{y}) \stackrel{\text{def}}{=} \{1, \ldots, d\} \smallsetminus \Omega(\mathbf{y}) \tag{$\dagger\dagger$}$$

its set of finite components. We introduce for this question a variant of the construction found in the main text, which results in a *Karp & Miller graph* G instead of a tree: in step 1 we rather add an edge $s \xrightarrow{\mathbf{a}}_G s_i$. Observe that this does not change C nor the termination of the algorithm. Karp & Miller graph

Show that, if $\mathbf{x}_0 \xrightarrow{u}_{\mathcal{V}} \mathbf{x}$ for some translation sequence u in $\mathbf{A}^*$, then there exists a node s in G labeled by $\mathbf{y}$ such that $\mathbf{x}(j) = \mathbf{y}(j)$ for all j in $\overline{\Omega}(\mathbf{y})$ and $s_0 \xrightarrow{u}_G s$ is a path in the graph.

(3) Let us prove that C satisfies Definition 1.20.2. The idea is that we can find reachable configurations of $\mathcal{V}$ that agree with $\mathbf{y}$ on its finite components, and that can be made arbitrarily high on its ω-components. For this, we focus on the graph nodes where new ω values are introduced by step 2, which we call *ω-nodes*. ω-node

Prove that, if $s_0 \xrightarrow{u}_T s$ labeled $\mathbf{y}$ for some u in $\mathbf{A}^*$ in the tree and $\mathbf{z}$ in $\mathbb{N}^{\Omega(\mathbf{y})}$ is a partial configuration on the components of $\Omega(\mathbf{y})$, then there are

- n in $\mathbb{N}$,

- a decomposition $u = u_1 u_2 \cdots u_{n+1}$ with each u_i in $\mathbf{A}^*$ where the nodes s_i reached by $s_0 \xrightarrow{u_1 \cdots u_i}_T s_i$ for $i \leq n$ are ω-nodes,
- sequences w_1, ..., w_n in $\mathbf{A}^+$,
- numbers k_1, ..., k_n in $\mathbb{N}$,

such that $\mathbf{x}_0 \xrightarrow{u_1 w_1^{k_1} u_2 \cdots u_n w_n^{k_n} u_{n+1}}_{\mathcal{V}} \mathbf{x}$ with $\mathbf{x}(j) = \mathbf{y}(j)$ for all j in $\overline{\Omega}(\mathbf{y})$ and $\mathbf{x}(j) \geq \mathbf{z}(j)$ for all j in $\Omega(\mathbf{y})$. Conclude.

Bibliographic Notes

Well Quasi Orders are "a frequently discovered concept", to quote the title of a survey by Kruskal (1972). Nevertheless, much of the theory appears in Higman (1952), although Dickson's Lemma already appeared (in a rather different form) in (Dickson, 1913). The reader will find more information in the survey of Milner (1985), which also covers *better quasi orders* (bqo), which allow to handle the problematic powerset constructions of Exercise 1.9—see (Marcone, 1994) for a good reference, and (Rado, 1954) or (Jančar, 1999) for a characterization of the wqos for which $(\mathcal{P}(A), \sqsubseteq_S)$ and/or $(A^\omega, \leq_\omega)$ is also a wqo. See Lovász (2006) for an exposition of Robertson and Seymour's Graph-Minor Theorem, its underlying ideas, and its consequences in graph theory.

better quasi order

Well Structured Transition Systems have been developed in different directions by Finkel (1987, 1990) and Abdulla et al. (1996), before a unifying theory finally emerged in the works of Abdulla et al. (2000) and Finkel and Schnoebelen (2001)—the latter being our main source for this chapter and exercises 1.11 to 1.13. More recent developments are concerned with the algorithmics of downward-closed sets (Finkel and Goubault-Larrecq, 2009, 2012) and of games (Abdulla et al., 2008; Bertrand and Schnoebelen, 2012).

Program Termination. Proving termination thanks to a ranking function into a well-founded ordering can be traced back at least to Turing (1949). The presentation in these notes rather follows Cook et al. (2011) and emphasizes the interest of transition invariants; see Podelski and Rybalchenko (2004) and the discussion of related work by Blass and Gurevich (2008).

Relevance Logic. The reader will find a good general exposition on relevance logic in the chapter of Dunn and Restall (2002), and in particular a discussion of decidability issues in their Section 4, from which Exercise 1.18 is taken (credited to Kripke, 1959). Both the approach in the exercise and that of the main text scale to larger fragments like the conjunctive implicative fragment $\mathbf{R}_{\supset,\wedge}$, but Urquhart (1984) proved the undecidability of the full relevance logic $\mathbf{R}$ and its variants the *entailment logic* $\mathbf{E}$ and *ticket logic* $\mathbf{T}$. This is still an active area of research: although Urquhart (1999) proved $\mathbf{R}_{\supset,\wedge}$ to be Ackermannian (see [CRI] on page 89), the complexity of [RI] is still unknown; similarily the decidability of implicative ticket logic $\mathbf{T}_{\supset}$ was only recently proven by Padovani (2012), and its complexity is also unknown.

Karp & Miller Trees and vector addition systems were first defined by Karp and Miller (1969). Coverability trees are used in a large number of algorithms and decision procedures on VAS, although their worst-case size can be Ackermannian in the size of the input VAS (Cardoza et al., 1976). Quite a few of these problems, including termination and coverability, can actually be solved in ExpSpace instead (Rackoff, 1978; Blockelet and Schmitz, 2011), but finite equivalences are an exception (Mayr and Meyer, 1981; Jančar, 2001); see [FCP] on page 88. The notion of *covering* can be generalized to complete WSTS, but they are in general not finite as in the VAS case (Finkel and Goubault-Larrecq, 2012).

2

COMPLEXITY UPPER BOUNDS

As seen in Chapter 1, many algorithms rely on well quasi orderings to prove the termination. Although it is true that the classical proofs of Dickson's Lemma, Higman's Lemma, and other wqos, are infinistic in nature, the way they are typically applied in algorithms lends itself to constructive proofs, from which complexity upper bounds can be extracted and applied to evaluate algorithmic complexities.

We present in this chapter how one can derive complexity upper bounds for these algorithms as a side-product of the use of Dickson's Lemma over tuples of integers. The techniques are however quite generic and also apply to more complex wqos; see the Bibliographic Notes at the end of the chapter.

Bad Sequences and Termination. Recall from Definition 1.1 that one of the characterizations for $(A, \leq)$ to be a wqo is that every infinite sequence $a_0, a_1, \ldots$ over A contains an *increasing pair* $a_{i_1} \leq a_{i_2}$ for some $i_1 < i_2$. We say that (finite or infinite) sequences with an increasing pair $a_{i_1} \leq a_{i_2}$ are *good* sequences, and call *bad* a sequence where no such increasing pair can be found. Therefore every infinite sequence over the wqo A is good, i.e., bad sequences over A are finite.

```
SIMPLE $(\mathrm{a}, \mathrm{b})$
$\mathrm{c} \longleftarrow 1$
**while** $\mathrm{a} > 0 \wedge \mathrm{b} > 0$
    $l : \langle \mathrm{a}, \mathrm{b}, \mathrm{c} \rangle \longleftarrow \langle \mathrm{a} - 1, \mathrm{b}, 2\mathrm{c} \rangle$
  **or**
    $r : \langle \mathrm{a}, \mathrm{b}, \mathrm{c} \rangle \longleftarrow \langle 2\mathrm{c}, \mathrm{b} - 1, 1 \rangle$
**end**
```

Figure 2.1: SIMPLE: A simple `while` program, repeated from Figure 1.1.

Recall the SIMPLE program from Figure 1.1 on page 7, repeated here in Figure 2.1. We argued on page 7 that, in any run, the sequence of values taken by a

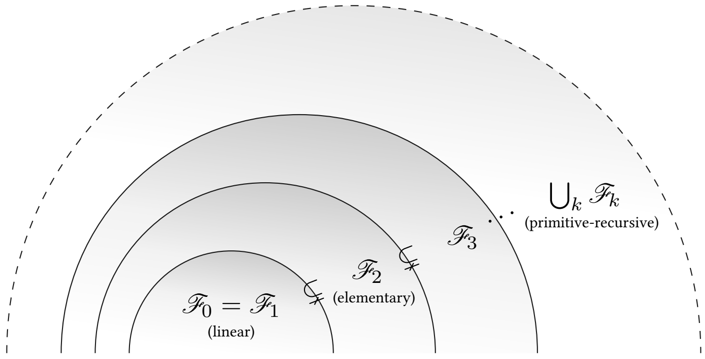

Figure 2.2: The Grzegorczyk hierarchy of primitive-recursive functions.

---

and b

$$\langle a_0, b_0\rangle, \ldots, \langle a_j, b_j\rangle, \ldots\ , \tag{2.1}$$

is a bad sequence over $(\mathbb{N}^2, \leq)$, and by Dickson's Lemma, it is finite, which means that SIMPLE always terminates.

In this chapter, we are going to see that the very fact that we applied Dickson's Lemma yields more that just the termination of SIMPLE: it also yields an upper bound on the number of times its main loop can be unrolled as a function of its initial input $\langle a_0, b_0\rangle$, i.e. a bound on the length of the bad sequence (2.1). Better, the upper bounds we will prove are highly *generic*, in that we only need to find out the complexity of the operations (i.e. only linear operations in SIMPLE) and the dimension we are working with (i.e. in dimension 2 in (2.1)), to provide an upper bound.

A LOWER BOUND. Before we investigate these upper bounds, let us have a look at how long SIMPLE can run: for instance, for $\langle a_0, b_0\rangle = \langle 2, 3\rangle$, we find the following run

$$\langle 2, 3, 2^0\rangle \xrightarrow{l} \langle 1, 3, 2^1\rangle \xrightarrow{r} \langle 2^2, 2, 2^0\rangle \xrightarrow{l^{2^2-1}r} \langle 2^{2^2}, 1, 1\rangle \xrightarrow{l^{2^{2^2}-1}r} \langle 0, 1, 2^{2^{2^2}}\rangle\ ,$$

of length

$$2 + 2^2 + 2^{2^2}\ , \tag{2.2}$$

which is non-elementary in the size of the initial values. This is instructive: linear operations and dimension 2 constitute the simplest case we care about, and the complexities we find are already beyond the elementary hierarchies, where the distinctions time vs. space resources, or deterministic vs. nondeterministic computations, become irrelevant. Hierarchies for non-elementary complexities are maybe not so well-known, so we will introduce one such hierarchy, the *Grzegorczyk hierarchy* $(\mathscr{F}_k)_{k\in\mathbb{N}}$ of classes of functions (see Figure 2.2).

As we will see, in the case of SIMPLE, we can show there exists a function bounding the length of all runs and residing in $\mathscr{F}_3$, which is the lowest level to contain non-elementary functions. Chapter 3 will be devoted to further matching complexity lower bounds for decision problems on monotonic counter systems.

Outline. The upcoming Section 2.1 surveys all the notions (controlled sequences, polynomial normed wqos, and the Grzegorczyk hierarchy) needed in order to state the Length Function Theorem, and later apply it to several algorithms in Section 2.2. The proof of the theorem is delayed until Section 2.3, which ends with the definition of a *bounding function* M on the length of controlled bad sequences, and Section 2.4 that classifies this function inside the Grzegorczyk hierarchy.

2.1 The Length of Controlled Bad Sequences

As seen with the example of Simple, wqo-based termination arguments rely on the finiteness of bad sequences. In order to further provide a complexity analysis, our goal is thus to bound the length of bad sequences.

2.1.1 Controlled Sequences

Our first issue with our program is that one can construct arbitrarily long bad sequences, even when starting from a fixed first element. Consider $\mathbb{N}^2$ and fix $\mathbf{x}_0 = \langle 0, 1\rangle$. Then the following

$$\langle 0,1\rangle, \langle L,0\rangle, \langle L-1,0\rangle, \langle L-2,0\rangle, \ldots, \langle 2,0\rangle, \langle 1,0\rangle \tag{2.3}$$

is a bad sequence of length $L+1$. What makes such examples possible is the "uncontrolled" jump from an element like $\mathbf{x}_0$ to an *arbitrarily* large next element, here $\mathbf{x}_1 = \langle L, 0\rangle$. Indeed, when one only considers bad sequences displaying some controlled behaviour (in essence, bad sequences of bounded complexity, as with the linear operations of Simple), upper bounds on their lengths certainly exist.

Norms and Controls. In order to control the growth of the values in a sequence $a_0, a_1, a_2, \ldots$ over some wqo $(A, \leq)$, we introduce two main ingredients:

1. the first is a *norm* $|.|_A \colon A \to \mathbb{N}$ on the elements to represent their size. We always assume $A_{<n} \overset{\text{def}}{=} \{a \in A \mid |a|_A < n\}$ to be *finite* for every n; we call the resulting structure $(A, \leq, |.|_A)$ a *normed wqo* (nwqo). For instance, for $\mathbb{N}^2$ we will use the *infinite norm* $|\langle m,n\rangle|_{\mathbb{N}^2} \overset{\text{def}}{=} \max(m,n)$; normed wqo

2. the second is a *control function* $g\colon \mathbb{N} \to \mathbb{N}$ used to bound the growth of elements as we iterate through the sequence. We always assume g to be *strictly increasing*: $g(x+1) \geq 1 + g(x)$ for all x. control function

controlled sequence

Mixing these together, we say that a sequence $a_0, a_1, a_2, \dots$ over A is (g,n)-*controlled* for some initial norm $n \in \mathbb{N}$ $\overset{\text{def}}{\Leftrightarrow}$

$$\forall i = 0,1,2,\dots :\ |a_i|_A < g^i(n) \overset{\text{def}}{=} \overbrace{g(g(\cdots g}^{i \text{ times}}(n)))\,. \tag{2.4}$$

In particular, $|a_0|_A < n$, hence the name "initial norm" for n. For instance, the bad sequence (2.1) over $\mathbb{N}^2$ extracted from the runs of SIMPLE is (g,n)-controlled for $g(x) = 2x$ and $n = \max(a_0, b_0) + 1$. Observe that the empty sequence is always a controlled sequence.

Definition 2.1 (Basic nwqos)**.** We note $[k]$ the nwqo $(\{0,\dots,k-1\}, \leq, |.|_{[k]})$ defined over the initial segment of the natural numbers, where $|j|_{[k]} \overset{\text{def}}{=} j$ for all $0 \leq j < k$, and Γ_k the generic k-elements nwqo $(\{a_0,\dots,a_{k-1}\}, =, |.|_{\Gamma_k})$ where $|a_j|_{\Gamma_k} \overset{\text{def}}{=} 0$ for all $0 \leq j < k$.

LENGTH FUNCTION. The outcome of these definitions is that, unlike in the uncontrolled case, *there is* a longest (g,n)-controlled bad sequence over any nwqo $(A, \leq_A, |.|_A)$: indeed, we can organize such sequences in a tree by sharing common prefixes; this tree has

- finite branching degree, bounded by the cardinal of $A_{<g^i(n)}$ for a node at depth i, and
- finite depth thanks to the wqo property.

length function

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

Remark 2.2 (Monotonicity of L). It is easy to see that $L_{g,A}(n)$ is monotone in the initial norm n (because g is increasing), but also in the choice of the control function: if $h(x) \geq g(x)$ for all x, then a (g,n)-controlled bad sequence is also an (h,n)-controlled one, thus $L_{g,A}(n) \leq L_{h,A}(n)$.

2.1.2 POLYNOMIAL NWQOS

Before we go any further in our investigation of the length function, let us first restrict the scope of our analysis.

nwqo isomorphism

ISOMORPHIMS. For one thing, we will work up to isomorphism: we write $A \equiv B$ when the two nwqo's A and B are *isomorphic* structures. For all practical purposes, isomorphic nwqos can be identified. Let us stress that, in particular, norm functions must be preserved by nwqo isomorphisms. Obviously, the length functions $L_{g,A}$ and $L_{g,B}$ are the same for isomorphic nwqos.

Example 2.3 (Isomorphisms)**.** On the positive side, $[0] \equiv \Gamma_0$ and also $[1] \equiv \Gamma_1$ since $|a_0|_{\Gamma_1} = 0 = |0|_{[1]}$.

However, $[2] \not\equiv \Gamma_2$: not only these two have non-isomorphic orderings, but they also have different norm functions. This can be witnessed by their associated length functions: one can see for instance that "a_1, a_0" is a $(g, 1)$-controlled bad sequence over Γ_2, but that the longest $(g, 1)$-controlled bad sequence over $[2]$ is the sequence "0" of length 1.

Polynomial nwqos. We are now ready to define the class of normed wqos we are interested in. We will need the *empty nwqo* $\Gamma_0 = \emptyset$, and a *singleton nwqo* Γ_1 containing a single element with norm 0, and using equality as ordering as in Example 2.3. The exact element found in this singleton is usually irrelevant; it could be for instance a letter in an alphabet, or a state in a finite state set.

empty nwqo

singleton nwqo

The *disjoint sum* of two nwqos $(A_1, \leq_{A_1}, |.|_{A_1})$ and $(A_2, \leq_{A_2}, |.|_{A_2})$ is the nwqo $(A_1 + A_2, \leq_{A_1+A_2}, |.|_{A_1+A_2})$ defined by

disjoint sum

$$A_1 + A_2 \overset{\text{def}}{=} \{\langle i, a\rangle \mid i \in \{1, 2\} \text{ and } a \in A_i\}\ , \tag{2.5}$$

$$\langle i, a\rangle \leq_{A_1+A_2} \langle j, b\rangle \overset{\text{def}}{\Leftrightarrow} i = j \text{ and } a \leq_{A_i} b\ , \tag{2.6}$$

$$|\langle i, a\rangle|_{A_1+A_2} \overset{\text{def}}{=} |a|_{A_i}\ . \tag{2.7}$$

We write $A \cdot k$ for $\overbrace{A + \cdots + A}^{k \text{ times}}$; then, any finite nwqo Γ_k can be defined as a k-ary disjoint sum $\Gamma_k \overset{\text{def}}{=} \Gamma_1 \cdot k$.

The *cartesian product* of two nwqos $(A_1, \leq_{A_1}, |.|_{A_1})$ and $(A_2, \leq_{A_2}, |.|_{A_2})$ is the nwqo $(A_1 \times A_2, \leq_{A_1\times A_2}, |.|_{A_1\times A_2})$ defined by

cartesian product

$$A_1 \times A_2 \overset{\text{def}}{=} \{\langle a_1, a_2\rangle \mid a_1 \in A_1, a_2 \in A_2\}\ , \tag{2.8}$$

$$\langle a_1, a_2\rangle \leq_{A_1\times A_2} \langle b_1, b_2\rangle \overset{\text{def}}{\Leftrightarrow} a_1 \leq_{A_1} b_1 \text{ and } a_2 \leq_{A_2} b_2\ , \tag{2.9}$$

$$|\langle a_1, a_2\rangle|_{A_1\times A_2} \overset{\text{def}}{=} \max_{i\in\{1,2\}} |a_i|_{A_i}\ . \tag{2.10}$$

The fact that $A_1 \times A_2$ is indeed a wqo is known as Dickson's Lemma. We note the d-fold cartesian product of a nwqo A with itself $A^d \overset{\text{def}}{=} \underbrace{A \times \cdots \times A}_{d \text{ times}}$; in particular $A^0 \equiv \Gamma_1$ is a singleton set containing only the empty tuple, of size 0 by (2.10).

Last, as we will be working on natural numbers, we also need the *naturals nwqo* $\mathbb{N}$ along with its usual ordering and the norm $|k|_{\mathbb{N}} \overset{\text{def}}{=} k$ for all k in $\mathbb{N}$.

naturals nwqo

Definition 2.4. The set of *polynomial nwqos* is the smallest set of nwqos containing Γ_0, Γ_1, and $\mathbb{N}$ and closed under the $+$ and $\times$ operations.

polynomial nwqo

Example 2.5 (VASS Configurations)**.** One can see that the set of configurations *Conf* of a d-dimensional VASS over a set of states Q with $|Q| = p$, along with its ordering, is isomorphic to the polynomial nwqo $\mathbb{N}^d \times \Gamma_p$.

Remark 2.6 (nwqo Semiring). Observe that the definitions are such that all the expected identities of $+$ and $\times$ hold: the class of *all nwqos* when considered up to isomorphism forms a *commutative semiring*: Γ_0 is neutral for $+$ and absorbing for $\times$:

$$\Gamma_0 + A \equiv A + \Gamma_0 \equiv A \qquad\qquad \Gamma_0 \times A \equiv A \times \Gamma_0 \equiv \Gamma_0\,, \tag{2.11}$$

Γ_1 is neutral for $\times$:

$$\Gamma_1 \times A \equiv A \times \Gamma_1 \equiv A\,, \tag{2.12}$$

$+$ is associative and commutative:

$$A + (B + C) \equiv (A + B) + C \qquad\qquad A + B \equiv B + A\,, \tag{2.13}$$

$\times$ is associative and commutative:

$$A \times (B \times C) \equiv (A \times B) \times C \qquad\qquad A \times B \equiv B \times A\,, \tag{2.14}$$

and $\times$ distributes over $+$:

$$(A + B) \times C \equiv (A \times C) + (B \times C)\,. \tag{2.15}$$

Remark 2.7 (Normal Form for Polynomial nwqos). An easy consequence of the identities from Remark 2.6 for polynomial nwqos is that any polynomial nwqo A can be put in a *polynomial normal form* (PNF)

polynomial normal form

$$A \equiv \mathbb{N}^{d_1} + \cdots + \mathbb{N}^{d_m} \tag{2.16}$$

for $m, d_1, \cdots, d_m \geq 0$. In particular, we denote the PNF of Γ_0 by "0." In Section 2.3.3 and later sections we will deal exclusively with nwqos in PNF; since $A \equiv A'$ implies $L_{g,A} = L_{g,A'}$ this will be at no loss of generality.

2.1.3 Subrecursive Functions

We already witnessed with simple that the complexity of some programs implementable as monotone counter systems can be quite high—more than a tower of exponentials $2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\}^{b \text{ times}}$ for $\text{simple}(2, b)$ in Equation (2.2) on page 24, which is a non-elementary function of b. However there is a vast space of functions that are non-elementary but recursive—and even primitive recursive, which will be enough for our considerations.

The Grzegorczyk Hierarchy $(\mathscr{F}_k)_{k<\omega}$ is a hierarchy of classes of primitive-recursive functions f with argument(s) and images in $\mathbb{N}$. Their union is exactly the set of primitive-recursive functions:

$$\bigcup_{k<\omega} \mathscr{F}_k = \text{FPR}\,. \tag{2.17}$$

The lower levels correspond to reasonable classes, $\mathscr{F}_0 = \mathscr{F}_1$ being the class of linear functions, and $\mathscr{F}_2$ that of elementary functions. Starting at level 1, the hierarchy is strict in that $\mathscr{F}_k \subsetneq \mathscr{F}_{k+1}$ for $k > 0$ (see Figure 2.2 on page 24).

fast-growing function

At the heart of each $\mathscr{F}_k$ lies the kth *fast-growing function* $F_k\colon \mathbb{N} \to \mathbb{N}$, which is defined for finite k by

$$F_0(x) \overset{\text{def}}{=} x + 1\,, \qquad F_{k+1}(x) \overset{\text{def}}{=} F_k^x(x) = \overbrace{F_k(F_k(\cdots F_k}^{x \text{ times}}(x)))\,. \tag{2.18}$$

This hierarchy of functions continues with ordinal indices, e.g.

$$F_\omega(x) \overset{\text{def}}{=} F_x(x)\,. \tag{2.19}$$

Observe that

$$F_1(x) = 2x\,, \qquad F_2(x) = 2^x x\,, \tag{2.20}$$

$$F_3(x) > \left.2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\right\}{\scriptstyle x \text{ times}} \qquad \text{etc.} \tag{2.21}$$

For $k \geq 2$, each level of the Grzegorczyk hierarchy can be characterized as

$$\mathscr{F}_k = \{f \mid \exists i,\ f \text{ is computed in time/space } \leq F_k^i\}\,, \tag{2.22}$$

the choice between deterministic and nondeterministic or between time-bounded and space-bounded computations being irrelevant because F_2 is already a function of exponential growth.

On the one hand, because the fast-growing functions are *honest*, i.e. can be computed in time elementary in their result, $F_k \in \mathscr{F}_k$ for all k. On the other hand, every function f in $\mathscr{F}_k$ is eventually bounded by F_{k+1}, i.e. there exists a rank x_f s.t. for all $x_1, \ldots, x_n$, if $\max_i x_i \geq x_f$, then $f(x_1, \ldots, x_n) \leq F_{k+1}(\max_i x_i)$. However, for all $k > 0$, honest function

$$F_{k+1} \notin \mathscr{F}_k\,. \tag{2.23}$$

In particular, F_ω is (akin to) the diagonal *Ackermann function*: it is not primitive-recursive and eventually bounds every primitive recursive function. Ackermann function

We delay more formal details on $(\mathscr{F}_k)_k$ until Section 2.4 on page 38 and Exercise 2.3 on page 46 and turn instead to the main theorem of the chapter.

2.1.4 Upper Bounds for Dickson's Lemma

Theorem 2.8 (Length Function Theorem). *Let g be a control function bounded by some function in $\mathscr{F}_\gamma$ for some $\gamma \geq 1$ and $d, p \geq 0$. Then $L_{g,\mathbb{N}^d \times \Gamma_p}$ is bounded by a function in $\mathscr{F}_{\gamma+d}$.* Length Function Theorem

The Length Function Theorem is especially tailored to give upper bounds for VASS configurations (recall Example 2.5 on page 27), but can also be used for VASS extensions. For instance, the runs of SIMPLE can be described by bad sequences in $\mathbb{N}^2$, of form described by Equation (2.1) on page 24. As these sequences are controlled by the linear function $g(x) = 2x$ in $\mathscr{F}_1$, the Length Function Theorem with $p = \gamma = 1$ entails the existence of a bounding function in $\mathscr{F}_3$ on the length of any run of SIMPLE, which matches the non-elementary length of the example run we provided in (2.2).

2.2 Applications

Besides providing complexity upper bounds for various problems, the results presented in this chapter also yield new "combinatorial" algorithms: we can now employ an algorithm that looks for a witness of *bounded size*. We apply this technique in this section to the two WSTS algorithms presented in Section 1.2.

Exercise 2.4 investigates the application of the Length Function Theorem to the program termination proofs of Section 1.3.1, and Exercise 2.14 to the Karp & Miller trees of Section 1.3.3. These applications remain quite generic, thus to make matters more concrete beforehand, let us mention that, in the case of vector addition systems with states (Example 1.13), lossy counter machines (Section 3.1), reset machines (Section 3.5), or other examples of well-structured counter machines with transitions controlled by $g(x) = x + b$ for some b—which is a function in $\mathscr{F}_1$—, with d counters, and with p states, the Length Function Theorem yields an upper bound in $\mathscr{F}_{d+1}$ on the length of controlled bad sequences. This is improved to $\mathscr{F}_d$ by Corollary 2.35 on page 45. When b or p is part of the input, this rises to $\mathscr{F}_{d+1}$, and when d is part of the input, to F_ω, which asymptotically dominates every primitive-recursive function.

2.2.1 Termination Algorithm

Let us consider the Termination problem of Section 1.2.1. Let $\mathcal{S} = \langle S, \rightarrow, \leq \rangle$ be a WSTS over a normed wqo $(S, \leq, |.|)$ where the norm $|.|$ is also the size for a concrete representation of elements in S, let s_0 be an initial state in S with $n = |s_0| + 1$, and let $g(|s|)$ be an upper bound on the space required to compute some s' from s verifying $s \rightarrow s'$. We can reasonably expect g to be increasing and honest, and use it as a control over sequences of states: we compute an upper bound

$$f(n) \geq L_{g,S}(n) \ . \tag{2.24}$$

As the Length Function Theorem and all the related results allow to derive *honest* upper bounds, this value can be computed in space elementary-recursive in f.

Because any run of $\mathcal{S}$ of length $\ell \overset{\text{def}}{=} f(n) + 1$ is necessarily good, we can replace the algorithm in the proof of Proposition 1.15 by an algorithm that looks for a finite witness of non-termination of form

$$s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_\ell \ . \tag{2.25}$$

This algorithm requires space at most $g^\ell(n)$ at any point i to compute some s_{i+1}, which yields a nondeterministic algorithm working in space elementary in $g^\ell(n)$. This falls in the same class as $f(n)$ itself in our setting—see Exercise 2.13 for an analysis of g^ℓ.

2.2.2 Coverability Algorithm

Recall that the algorithm of Section 1.2.2 for WSTS coverability of t from s, relied on the saturation of a sequence (1.4) on page 6 of subsets of S. In order to derive an upper complexity bound on this problem, we look instead at how long we might have to wait until this sequence proves coverability, i.e. consider the length of

$$\uparrow\{t\} = I_0 \subsetneq I_1 \subsetneq \cdots \subsetneq I_\ell, \text{ where } s \in I_\ell \text{ but } s \notin I_i \text{ for any } i < \ell\ . \tag{2.26}$$

For each $i = 1, \ldots, \ell$, let s_i be a minimal element in the non-empty set $I_i \smallsetminus I_{i-1}$; then there must be one such $s_\ell \leq s$ that does not appear in any of the I_i for $i < \ell$, and we consider a particular sequence

$$s_1, s_2, \ldots, s_\ell \leq s\ . \tag{2.27}$$

Note that $s_j \not\geq s_i$ for $j > i$, since $s_j \notin I_i$ and the sequence $s_1, s_2, \ldots$ in (2.27) is bad—this also proves the termination of the $(I_i)_i$ sequence in (2.26).

We now need to know how the sequence in (2.27) is controlled. Note that in general $s_i \not\to s_{i+1}$, thus we really need to consider the sets of minimal elements in (2.26) and bound more generally the length of *any* sequence of s_i's where each s_i is a minimal element of $I_i \smallsetminus I_{i-1}$. Assume again that $\mathcal{S} = \langle S, \to, \leq\rangle$ is a WSTS over a normed wqo $(S, \leq, |.|)$ where the norm $|.|$ is also the size for a concrete representation of states in S. Also assume that $s' \leq s$ can be tested in space elementary in $|s'| + |s|$, and that elements of $pb(s)$ can be computed in space $g(|s|)$ for a honest increasing g: then $\ell \leq L_{g,S}(|t| + 1)$.

There is therefore a sequence

$$t = s'_0, s'_1, \ldots, s'_\ell = s_\ell \leq s \text{ where } s'_{i+1} \in pb(s'_i) \tag{2.28}$$

of minimal elements in $(I_i)_i$ that eventually yields $s_\ell \leq s$. We derive again a nondeterministic algorithm that looks for a witness (2.28) of bounded length. Furthermore, each s'_i verifies $|s'_i| \leq g^\ell(|t| + 1)$, which means that this algorithm works in nondeterministic space elementary in $g^\ell(|t| + 1) + |s|$.

2.3 Bounding the Length Function

This section and the next together provide a proof for the Length Function Theorem. The first part of this proof investigates the properties of bad controlled sequences and derives by induction over polynomial nwqos a *bounding function* $M_{g,A}(n)$ on the length of (g, n)-controlled bad sequences over A (see Proposition 2.20 on page 38). The second part, detailed in Section 2.4, studies the properties of the $M_{g,A}$ functions, culminating with their classification in the Grzegorczyk hierarchy.

2.3.1 Residual nwqos and a Descent Equation

Returning to the length function, let us consider a very simple case, namely the case of sequences over $\mathbb{N}$: one can easily see that

$$L_{g,\mathbb{N}}(n) = n \tag{2.29}$$

because the longest (g, n)-controlled bad sequence over $\mathbb{N}$ is simply

$$n - 1, n - 2, \dots, 1, 0 \tag{2.30}$$

of length n.

Formally, (2.30) proves that $L_{g,\mathbb{N}}(n) \geq n$; an argument for the converse inequality could use roughly the following lines: in any (g, n)-controlled bad sequence of natural integers $k, l, m, \dots$ over $\mathbb{N}$, once the first element $k < n$ has been fixed, the remaining elements $l, m, \dots$ have to be chosen inside a finite set $\{0, \dots, k - 1\}$ of cardinal k—or the sequence would be good. Thus this suffix, which itself has to be bad, is of length at most

$$L_{g,\Gamma_k}(n) = \begin{cases} 0 & \text{if } n = 0 \\ k & \text{otherwise} \end{cases} \tag{2.31}$$

by the *pigeonhole principle*. Choosing $k = n - 1$ maximizes the length of the bad sequence in (2.31), which shows that $L_{g,\mathbb{N}}(n) \leq n$.

This argument is still a bit blurry (and will soon be cleared out), but it already contains an important insight: in a (g, n)-controlled bad sequence $a_0, a_1, a_2, \dots$ over some nwqo A, we can distinguish between the first element a_0, which verifies $|a_0|_A < n$, and the suffix sequence $a_1, a_2, \dots$, which

1. verifies $a_0 \not\leq a_i$ for all $i > 0$,
2. is itself a bad sequence—otherwise the full sequence $a_0, a_1, a_2, \dots$ would be good,
3. is controlled by $(g, g(n))$—otherwise the full sequence $a_0, a_1, a_2, \dots$ would not be (g, n)-controlled.

Item 1 motivates the following definition:

residual nwqo

Definition 2.9 (Residuals)**.** For a nwqo A and an element $a \in A$, the *residual nwqo* A/a is the substructure (a nwqo) induced by the subset $A/a \overset{\text{def}}{=} \{a' \in A \mid a \not\leq a'\}$ of elements that are not above a.

Example 2.10 (Residuals)**.** For all $l < k$ and $i \in \{1, \dots, k\}$:

$$\mathbb{N}/l = [k]/l = [l]\,, \qquad\qquad \Gamma_k/a_i \equiv \Gamma_{k-1}\,. \tag{2.32}$$

The conditions 1–3 on the suffix sequence $a_1, a_2, \ldots$ show that it is a $(g, g(n))$-controlled bad sequence over A/a_0. Thus by choosing an $a'_0 \in A_{<n}$ that maximizes $L_{g,A/a'_0}(g(n))$ through some suffix sequence $a'_1, a'_2, \ldots$, we can construct a (g, n)-controlled bad sequence $a'_0, a'_1, a'_2, \ldots$ of length $1 + L_{g,A/a'_0}(g(n))$, which shows

$$L_{g,A}(n) \geq \max_{a \in A_{<n}} \left\{1 + L_{g,A/a}(g(n))\right\} . \tag{2.33}$$

The converse inequality is easy to check: consider a maximal (g, n)-controlled bad sequence $a''_0, a''_1, \ldots$ over A, thus of length $L_{g,A}(n)$. If this sequence is not empty, i.e. if $L_{g,A}(n) > 0$, then $a''_0 \in A_{<n}$ and its suffix $a''_1, a''_2, \ldots$ is of length $L_{g,A/a''_0}(g(n))$—or we could substitute a longer suffix. Hence:

Proposition 2.11 (Descent Equation)**.** Descent Equation

$$L_{g,A}(n) = \max_{a \in A_{<n}} \left\{1 + L_{g,A/a}(g(n))\right\} . \tag{2.34}$$

This reduces the $L_{g,A}$ function to a finite combination of L_{g,A_i}'s where the A_i's are residuals of A, hence "smaller" sets. Residuation is well-founded for nwqos: a sequence of successive residuals $A \supsetneq A/a_0 \supsetneq A/a_0/a_1 \supsetneq \cdots$ is necessarily finite since $a_0, a_1, \ldots$ must be a bad sequence. Hence the recursion in the Descent Equation is well-founded and can be used to evaluate $L_{g,A}(n)$. This is our starting point for analyzing the behaviour of length functions.

Example 2.12. Let us consider the case of $L_{g,[k]}(n)$ for $k < n$: by induction on k, we can see that

$$L_{g,[k]}(n) = k . \tag{2.35}$$

Indeed, this holds trivially for $[0] = \emptyset$, and for the induction step, it also holds for $k + 1 < n$ since $[k + 1]_{<n} = [k + 1]$ and thus by the Descent Equation

$$\begin{aligned} L_{g,[k+1]}(n) &= \max_{l \in [k+1]} \left\{1 + L_{g,[k+1]/l}(g(n))\right\} \\ &= \max_{l \in [k+1]} \left\{1 + L_{g,[l]}(g(n))\right\} \\ &= \max_{l \in [k+1]} \left\{1 + l\right\} \\ &= 1 + k \end{aligned}$$

using (2.32) and the induction hypothesis on $l \leq k < n \leq g(n)$.

Example 2.13. Let us consider again the case of $L_{g,\mathbb{N}}$: by the Descent Equation,

$$\begin{aligned} L_{g,\mathbb{N}}(n) &= \max_{k \in \mathbb{N}_{<n}} \left\{1 + L_{g,\mathbb{N}/k}(g(n))\right\} \\ &= \max_{k \in \mathbb{N}_{<n}} \left\{1 + L_{g,[k]}(g(n))\right\} \\ &= \max_{k \in \mathbb{N}_{<n}} \left\{1 + k\right\} \\ &= n \end{aligned}$$

thanks to (2.32) and (2.35) on $k < n$.

2.3.2 Reflecting nwqos

The reader might have noticed that Example 2.13 does not quite follow the reasoning that led to (2.29) on page 32: although we started by decomposing bad sequences into a first element and a suffix as in the Descent Equation, we rather used (2.31) to treat the suffix by seeing it as a bad sequence over Γ_{n-1} and deduce $L_{g,\mathbb{N}}(n)$. However, as already mentioned in Example 2.3 on page 27, $\Gamma_{n-1} \not\equiv [n-1]$ in general.

We can reconcile the analyses made for (2.29) on page 32 and in Example 2.13 by noticing that bad sequences are never shorter in Γ_{n-1} than in $[n-1]$. We will prove this formally using *reflections*, which let us simplify instances of the Descent Equation by replacing all A/a for $a \in A_{<n}$ by a single (or a few) A' that is larger than any of the considered A/a's—but still reasonably small compared to A, so that a well-founded inductive reasoning remains possible.

nwqo reflection

Definition 2.14. A *nwqo reflection* is a mapping $h\colon A \to B$ between two nwqos that satisfies the two following properties:

$$\forall a, a' \in A : h(a) \leq_B h(a') \text{ implies } a \leq_A a' \,, \tag{2.36}$$

$$\forall a \in A : |h(a)|_B \leq |a|_A \,. \tag{2.37}$$

In other words, a nwqo reflection is an order reflection that is also norm-decreasing (not necessarily strictly).

We write $h\colon A \hookrightarrow B$ when h is a nwqo reflection and say that B *reflects* A. This induces a relation between nwqos, written $A \hookrightarrow B$.

Reflection is transitive since $h\colon A \hookrightarrow B$ and $h'\colon B \hookrightarrow C$ entails $h' \circ h\colon A \hookrightarrow C$. It is also reflexive, hence reflection is a quasi-ordering. Any nwqo reflects its induced substructures since $\mathrm{Id}\colon X \hookrightarrow A$ when X is a substructure of A. Thus $\Gamma_0 \hookrightarrow A$ for any A, and $\Gamma_1 \hookrightarrow A$ for any non-empty A.

Example 2.15 (Reflections)**.** Among the basic nwqos from Example 2.3, we note the following relations (or absences thereof). For any $k \in \mathbb{N}$, $[k] \hookrightarrow \Gamma_k$, while $\Gamma_k \not\hookrightarrow [k]$ when $k \geq 2$. The reflection of induced substructures yields $[k] \hookrightarrow \mathbb{N}$ and $\Gamma_k \hookrightarrow \Gamma_{k+1}$. Obviously, $\mathbb{N} \not\hookrightarrow [k]$ and $\Gamma_{k+1} \not\hookrightarrow \Gamma_k$.

Reflections preserve controlled bad sequences. Let $h\colon A \hookrightarrow B$, consider a sequence $s = a_0, a_1, \ldots$ over A, and write $h(s)$ for $h(a_0), h(a_1), \ldots$, a sequence over B. Then by (2.36), $h(s)$ is bad when s is, and by (2.37), it is (g, n)-controlled when s is. Hence we can complete the picture of the monotonicity properties of L started in Remark 2.2 on page 26:

Proposition 2.16 (Monotonicity of L in A)**.**

$$A \hookrightarrow B \textit{ implies } L_{g,A}(n) \leq L_{g,B}(n) \textit{ for all } g, n \,. \tag{2.38}$$

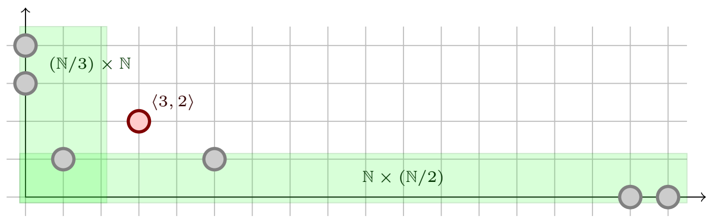
$(\mathbb{N}/3)\times\mathbb{N}$
$\langle 3,2\rangle$
$\mathbb{N}\times(\mathbb{N}/2)$

Figure 2.3: The elements of the bad sequence (2.42) and the two regions for the decomposition of $\mathbb{N}^2/\langle 3,2\rangle$.

This is the last missing piece for deducing (2.29) from (2.31): since $[k] \hookrightarrow \Gamma_k$, $L_{g,[k]}(n) \leq L_{g,\Gamma_k}(n)$ by Proposition 2.16—the converse inequality holds for $k < n$, as seen with (2.31) and (2.35), but not for $k \geq n > 1$ as seen in Example 2.3.

Remark 2.17 (Reflection is a Precongruence). Reflections are compatible with product and sum:

$$A \hookrightarrow A' \text{ and } B \hookrightarrow B' \text{ imply } A+B \hookrightarrow A'+B' \text{ and } A\times B \hookrightarrow A'\times B' \,. \tag{2.39}$$

Inductive Residual Computations. We may now tackle our first main problem: computing residuals A/a. The Descent Equation, though it offers a powerful way of computing the length function, can very quickly lead to complex expressions, as the nwqos $A/a_0/a_1/\cdots/a_n$ become "unstructured", i.e. have no nice definition in terms of $+$ and $\times$. Residuation allows us to approximate these sets s.t. the computation can be carried out without leaving the realm of polynomial nwqos, leading to an *inductive* computation of A/a over the structure of the polynomial nwqo A.

The base cases of this induction were already provided as (2.32) for finite sets Γ_k, and

$$\mathbb{N}/k \hookrightarrow \Gamma_k \tag{2.40}$$

for the naturals $\mathbb{N}$—because $\mathbb{N}/k = [k]$ by (2.32), and then $[k] \hookrightarrow \Gamma_k$ as seen in Example 2.15—, which was implicit in the computation of $L_{g,\mathbb{N}}$ in (2.29). Regarding disjoint sums $A+B$, it is plain that

$$(A+B)/\langle 1,a\rangle = (A/a)+B\,, \qquad (A+B)/\langle 2,b\rangle = A+(B/b)\,, \tag{2.41}$$

and reflections are not required.

The case of cartesian products $A\times B$ is different: Let $g(x)=2x$ and consider the following $(g,4)$-controlled bad sequence over $\mathbb{N}^2$

$$\langle 3,2\rangle,\ \langle 5,1\rangle,\ \langle 0,4\rangle,\ \langle 17,0\rangle,\ \langle 1,1\rangle,\ \langle 16,0\rangle,\ \langle 0,3\rangle\ . \tag{2.42}$$

Our purpose is to reflect $\mathbb{N}^2/\langle 3,2\rangle$ into a simpler polynomial nwqo. The main intuition is that, for each tuple $\langle a,b\rangle$ in the suffix, $\langle 3,2\rangle \not\leq \langle a,b\rangle$ entails that

$3 \not\leq a$ or $2 \not\leq b$. Thus we can partition the elements of this suffix into two groups: the pairs where the first coordinate is in $\mathbb{N}/3$, and the pairs where the second coordinate is in $\mathbb{N}/2$—an element might fulfill both conditions, in which case we choose an arbitrary group for it. Thus the elements of the suffix can be either from $(\mathbb{N}/3)\times\mathbb{N}$ or from $\mathbb{N}\times(\mathbb{N}/2)$, and the whole suffix can be reflected into their disjoint sum $(\mathbb{N}/3)\times\mathbb{N}+\mathbb{N}\times(\mathbb{N}/2)$.

For our example (2.42), we obtain the decomposition (see also Figure 2.3)

$$\langle 3,2\rangle, \left\{ \begin{array}{lllllll} \langle 5,1\rangle, & . & \langle 17,0\rangle, & \langle 1,1\rangle, & \langle 16,0\rangle, & . & \in \mathbb{N}\times(\mathbb{N}/2) \\ . & \langle 0,4\rangle, & . & . & . & \langle 0,3\rangle & \in (\mathbb{N}/3)\times\mathbb{N} \end{array} \right. \tag{2.43}$$

We could have put $\langle 1,1\rangle$ in either $\mathbb{N}\times(\mathbb{N}/2)$ or $(\mathbb{N}/3)\times\mathbb{N}$ but we had no choice for the other elements of the suffix. Observe that the two subsequences $\langle 0,4\rangle\langle 0,3\rangle$ and $\langle 5,1\rangle,\langle 17,0\rangle,\langle 1,1\rangle,\langle 16,0\rangle$ are indeed bad, but not necessarily $(g,g(4))$-controlled: $|\langle 17,0\rangle| = 17 \geq 16 = g(g(4))$. However, we do not see them as *independent* sequences but consider their disjoint sum instead, so that their elements inherit their positions from the original sequence, and indeed the suffix sequence in (2.43) is $(g,g(4))$-controlled.

By a straightforward generalization of the argument:

$$(A\times B)/\langle a,b\rangle \hookrightarrow \big((A/a)\times B\big) + \big(A\times(B/b)\big)\ . \tag{2.44}$$

Since it provides reflections instead of isomorphisms, (2.44) is not meant to support exact computations of A/a by induction over the structure of A (see Exercise 2.5). More to the point, it yields over-approximations that are sufficiently precise for our purposes while bringing important simplifications when we have to reflect the A/a for all $a \in A_{<n}$.

2.3.3 A Bounding Function

It is time to wrap up our analysis of L. We first combine the inductive residuation and reflection operations into *derivation relations* ∂_n: intuitively, the relation $A\ \partial_n\ A'$ is included in the relation "$A/a \hookrightarrow A'$ for some $a\in A_{<n}$" (see Lemma 2.19 for the formal statement). More to the point, the derivation relation captures a *particular* way of reflecting residuals, which enjoys some good properties: for every n, given A a nwqo in *polynomial normal form* (recall Remark 2.7 on page 28), $\partial_n A$ is a *finite* set of polynomial nwqos also in PNF, defined inductively by

nwqo derivation

$$\partial_n 0 \overset{\text{def}}{=} \emptyset\ , \tag{2.45}$$

$$\partial_n \mathbb{N}^0 \overset{\text{def}}{=} \{0\}\ , \tag{2.46}$$

$$\partial_n \mathbb{N}^d \overset{\text{def}}{=} \{\mathbb{N}^{d-1}\cdot(n-1)d\}\ , \tag{2.47}$$

$$\partial_n(A+B) \overset{\text{def}}{=} \big((\partial_n A)+B\big)\cup\big(A+(\partial_n B)\big)\ , \tag{2.48}$$

for $d > 0$ and A, B in PNF; in these definitions the $+$ operations are lifted to act upon nwqo sets S by $A + S \overset{\text{def}}{=} \{A + A' \mid A' \in S\}$ and symmetrically. Note that (2.46) can be seen as a particular case of (2.47) if we ignore the undefined $\mathbb{N}^{0-1}$ and focus on its coefficient 0.

An important fact that will become apparent in the next section is

Fact 2.18 (Well-Foundedness). *The relation $\partial \overset{\text{def}}{=} \bigcup_n \partial_n$ is well-founded.*

The definition of ∂_n verifies:

Lemma 2.19. *Let A be a polynomial nwqo in PNF and $a \in A_{<n}$ for some n. Then there exists A' in $\partial_n A$ s.t. $A/a \hookrightarrow A'$.*

Proof. Let $A \equiv \mathbb{N}^{d_1} + \cdots + \mathbb{N}^{d_m}$ in PNF and let $a \in A_{<n}$ for some n; note that the existence of a rules out the case of $m = 0$ (i.e. $A \equiv \Gamma_0$), thus (2.45) vacuously verifies the lemma.

We proceed by induction on $m > 0$: the base case is $m = 1$, i.e. $A \equiv \mathbb{N}^d$, and perform a nested induction on d: if $d = 0$, then $A \equiv \Gamma_1$, thus $A/a \equiv \Gamma_0$ by (2.32): this is in accordance with (2.46), and the lemma holds. If $d = 1$, i.e. $A \equiv \mathbb{N}$, then $A/a \hookrightarrow \Gamma_a$ by (2.40), and then $\Gamma_a \hookrightarrow \Gamma_{n-1} \equiv \mathbb{N}^0 \cdot (n-1)$ as seen in Example 2.15 since $a < n$, thus (2.47) verifies the lemma. For the induction step on $d > 1$,

$$A \equiv \mathbb{N}^d = \mathbb{N} \times \mathbb{N}^{d-1}$$

and thus $a = \langle k, b\rangle$ for some $k \in \mathbb{N}_{<n}$ and $b \in \mathbb{N}^{d-1}_{<n}$. By (2.44),

$$A/a \hookrightarrow \big((\mathbb{N}/k) \times \mathbb{N}^{d-1}\big) + \big(\mathbb{N} \times (\mathbb{N}^{d-1}/b)\big) .$$

Using the ind. hyp. on $\mathbb{N}/k$ along with Remark 2.17,

$$\begin{aligned} &\hookrightarrow \big((\mathbb{N}^0 \cdot (n-1)) \times \mathbb{N}^{d-1}\big) + \big(\mathbb{N} \times (\mathbb{N}^{d-1}/b)\big) \\ &\equiv (\mathbb{N}^{d-1} \cdot (n-1)) + \big(\mathbb{N} \times (\mathbb{N}^{d-1}/b)\big) . \end{aligned}$$

Using the ind. hyp. on $\mathbb{N}^{d-1}/b$ along with Remark 2.17,

$$\begin{aligned} &\hookrightarrow (\mathbb{N}^{d-1} \cdot (n-1)) + \big(\mathbb{N} \times (\mathbb{N}^{d-2} \cdot (d-1)(n-1))\big) \\ &\equiv \mathbb{N}^{d-1} \cdot d(n-1) , \end{aligned}$$

in accordance with (2.47).

For the induction step on $m > 1$, i.e. if $A \equiv B + C$, then wlog. $a = \langle 1, b\rangle$ for some $b \in B_{<n}$ and thus by (2.41) $A/a = (B/b) + C$. By ind. hyp., there exists $B' \in \partial_n B$ s.t. $B/b \hookrightarrow B'$, thus $A/a \hookrightarrow B' + C$ by Remark 2.17, the latter nwqo being in $\partial_n A$ according to (2.48). □

The computation of derivatives can be simplified by replacing (2.45) and (2.48) by a single equation (see Exercise 2.6):

$$\partial_n A = \{B + \partial_n \mathbb{N}^d \mid A \equiv B + \mathbb{N}^d, d \geq 0\} . \tag{2.49}$$

bounding function

The Bounding Function $M_{g,A}$ for A a polynomial nwqo in PNF is defined by

$$M_{g,A}(n) \overset{\text{def}}{=} \max_{A' \in \partial_n A} \left\{1 + M_{g,A'}(g(n))\right\} . \tag{2.50}$$

This function M is well-defined as a consequence of Fact 2.18 and of the finiteness of $\partial_n A$ for all n and A; its main property is

Proposition 2.20. *For any polynomial nwqo A in PNF, any control function g, and any initial control n,*

$$L_{g,A}(n) \leq M_{g,A}(n) . \tag{2.51}$$

Proof. Either $A_{<n}$ is empty and then $L_{g,A}(n) = 0 \leq M_{g,A}(n)$, or there exists some $a \in A_{<n}$ that maximizes $L_{g,A/a}(g(n))$ in the Descent Equation, i.e.

$$L_{g,A}(n) = 1 + L_{g,A/a}(g(n)) .$$

By Lemma 2.19 there exists $A' \in \partial_n A$ s.t. $A/a \hookrightarrow A'$, thus by Proposition 2.16

$$L_{g,A}(n) \leq 1 + L_{g,A'}(g(n)) .$$

By well-founded induction on $A' \in \partial_n A$, $L_{g,A'}(g(n)) \leq M_{g,A'}(g(n))$, thus

$$L_{g,A}(n) \leq 1 + M_{g,A'}(g(n)) \leq M_{g,A}(n)$$

by definition of M. $\square$

2.4 ⋆ Classification in the Grzegorczyk Hierarchy

Now equipped with a suitable bound $M_{g,A}(n)$ on the length of (g,n)-controlled bad sequences over A, the only remaining issue is its classification inside the Grzegorczyk hierarchy. We first exhibit a very nice isomorphism between polynomial nwqos (seen up to isomorphism) and their *maximal order types*, which are ordinals below ω^ω.

2.4.1 Maximal Order Types

Consider a wqo $(A, \leq)$: it defines an associated strict ordering $< \overset{\text{def}}{=} \{(a,a') \in A^2 \mid a \leq a' \text{ and } a' \not\leq a\}$. There are many possible *linearizations* $\prec$ of $<$, i.e. linear orders with $< \subseteq \prec$, obtained by equating equivalent elements and "orienting" the pairs of incomparable elements (a,a') of $(A,\leq)$. Each of these linearizations is a well-ordering and is thus isomorphic to some ordinal, called its *order type*, that intuitively captures its "length." The *maximal order type* of $(A,\leq)$ is then defined as the maximal such order type over all the possible linearizations; it provides a measure of the complexity of the (n)wqo.

order type

maximal order type

Example 2.21 (Maximal Order Types)**.** In a finite set Γ_k, the strict ordering is empty and the $k!$ different linear orders over Γ_k are all of order type k. In an initial segment of the naturals $[k]$ (respectively in the naturals $\mathbb{N}$), the only linearization is the natural ordering $<$ itself, which is of order type k (respectively ω):

$$o(\Gamma_k) = o([k]) = k\,, \qquad\qquad o(\mathbb{N}) = \omega\,. \tag{2.52}$$

Remark 2.22. By definition of the maximal order type of a nwqo A, if $A \equiv A'$ then $o(A) = o(A')$.

As seen with our example, the maximal order type of a polynomial nwqo is not necessarily finite, which prompts us to recall a few elements of ordinal notations.

Ordinal Terms. Let ε_0 be the smallest solution of the equation $\omega^x = x$. It is well-known that ordinals below ε_0 can be written down in a canonical way as ordinal terms in *Cantor Normal Form* (CNF), i.e. sums

Cantor Normal Form

$$\alpha = \omega^{\beta_1} + \cdots + \omega^{\beta_m} = \sum_{i=1}^{m} \omega^{\beta_i} \tag{2.53}$$

with $\beta_1 \geq \cdots \geq \beta_m \geq 0$ and each β_i itself a term in CNF. We write 1 for ω^0 and $\alpha \cdot n$ for $\overbrace{\alpha + \cdots + \alpha}^{n \text{ times}}$. Recall that the *direct sum* operator $+$ is associative ($(\alpha+\beta)+\gamma = \alpha + (\beta+\gamma)$) and idempotent ($\alpha + 0 = \alpha = 0 + \alpha$) but not commutative (e.g. $1 + \omega = \omega \neq \omega + 1$). An ordinal term α of form $\gamma + 1$ is called a *successor ordinal*. Otherwise, if not 0, it is a *limit ordinal*, usually denoted λ. We write CNF(α) for the set of ordinal terms $\alpha' < \alpha$ in CNF (which is in bijection with the ordinal α, and we use ordinal terms in CNF and set-theoretic ordinals interchangeably).

successor ordinal
limit ordinal

When working with terms in CNF, the *ordinal ordering* $<$, which is a well ordering over ordinals, has a syntactic characterization akin to a lexicographic ordering:

ordinal ordering

$$\sum_{i=1}^{m} \omega^{\beta_i} < \sum_{i=1}^{n} \omega^{\beta'_i} \Leftrightarrow \begin{cases} m < n \text{ and } \forall 1 \leq i \leq m, \beta_i = \beta'_i, \text{ or} \\ \exists 1 \leq j \leq \min(m,n), \beta_j < \beta'_j \text{ and } \forall 1 \leq i < j, \beta_i = \beta'_i\,. \end{cases} \tag{2.54}$$

Also recall the definitions of the *natural sum* $\alpha \oplus \alpha'$ and *natural product* $\alpha \otimes \alpha'$ of two terms in CNF:

natural sum
natural product

$$\sum_{i=1}^{m} \omega^{\beta_i} \oplus \sum_{j=1}^{n} \omega^{\beta'_j} \stackrel{\text{def}}{=} \sum_{k=1}^{m+n} \omega^{\gamma_k}\,, \quad \sum_{i=1}^{m} \omega^{\beta_i} \otimes \sum_{j=1}^{n} \omega^{\beta'_j} \stackrel{\text{def}}{=} \bigoplus_{i=1}^{m} \bigoplus_{j=1}^{n} \omega^{\beta_i \oplus \beta'_j}\,, \tag{2.55}$$

where $\gamma_1 \geq \cdots \geq \gamma_{m+n}$ is a reordering of $\beta_1, \ldots, \beta_m, \beta'_1, \ldots, \beta'_n$.

Maximal Order Types. We map polynomial nwqos $(A, \leq, |.|_A)$ to ordinals in ω^ω using the *maximal order type* $o(A)$ of the underlying wqo $(A, \leq)$. Formally, $o(A)$ can be computed inductively using (2.52) and the following characterization:

Fact 2.23. *For any wqos A and B*

$$o(A+B) = o(A) \oplus o(B)\,, \qquad o(A \times B) = o(A) \otimes o(B)\,. \tag{2.56}$$

Example 2.24. Given a polynomial nwqo in PNF $A \equiv \sum_{i=1}^{m} \mathbb{N}^{d_i}$, its associated maximal order type is $o(A) = \bigoplus_{i=1}^{m} \omega^{d_i}$, which is in ω^ω. It turns out that o is a bijection between polynomial nwqos and ω^ω (see Exercise 2.7).

It is more convenient to reason with ordinal arithmetic rather than with polynomial nwqos, and we lift the definitions of ∂ and M to ordinals in ω^ω. Define for all α in ω^ω and all d, n in $\mathbb{N}$

$$\partial_n \omega^d \overset{\text{def}}{=} \begin{cases} 0 & \text{if } d = 0 \\ \omega^{d-1} \cdot (d(n-1)) & \text{otherwise} \end{cases} \tag{2.57}$$

$$\partial_n \alpha \overset{\text{def}}{=} \left\{ \gamma \oplus \partial_n \omega^d \mid \alpha = \gamma \oplus \omega^d \right\} \tag{2.58}$$

$$M_{g,\alpha}(n) \overset{\text{def}}{=} \max_{\alpha' \in \partial_n \alpha} \left\{ 1 + M_{g,\alpha'}(g(n)) \right\}. \tag{2.59}$$

Equation (2.57) restates (2.46) and (2.47) using maximal order types, while (2.58) and (2.59) mirror respectively (2.49) and (2.50) but work in ω^ω; one easily obtains the following slight variation of Proposition 2.20:

Corollary 2.25. *For any polynomial nwqo A, any control function g, and any initial control n,*

$$L_{g,A}(n) \leq M_{g,o(A)}(n)\,. \tag{2.60}$$

A benefit of ordinal notations is that the well-foundedness of ∂ announced in Fact 2.18 is now an immediate consequence of $<$ being a well ordering: one can check that for any n, $\alpha' \in \partial_n \alpha$ implies $\alpha' < \alpha$ (see Exercise 2.8).

Example 2.26. One can check that

$$M_{g,k}(n) = k \qquad\qquad M_{g,\omega}(n) = n\,. \tag{2.61}$$

(Note that if $n > 0$ this matches $L_{g,\Gamma_k}(n)$ exactly by (2.31)). This follows from

$$\partial_n k = \begin{cases} \emptyset & \text{if } k = 0 \\ \{k-1\} & \text{otherwise.} \end{cases} \tag{2.62}$$

2.4.2 The Cichoń Hierarchy

A second benefit of working with ordinal indices is that we can exercise a richer theory of subrecursive hierarchies, for which many results are known. Let us first introduce the basic concepts.

FUNDAMENTAL SEQUENCES. Subrecursive hierarchies are defined through assignments of *fundamental sequences* $(\lambda_x)_{x<\omega}$ for limit ordinal terms λ, verifying $\lambda_x < \lambda$ for all x and $\lambda = \sup_x \lambda_x$. The usual way to obtain families of fundamental sequences is to fix a particular sequence ω_x for ω and to define on ordinal terms in CNF

fundamental sequence

$$(\gamma + \omega^{\beta+1})_x \overset{\text{def}}{=} \gamma + \omega^\beta \cdot \omega_x \,, \qquad (\gamma + \omega^\lambda)_x \overset{\text{def}}{=} \gamma + \omega^{\lambda_x} \,. \tag{2.63}$$

We always assume the standard assignment $\omega_x \overset{\text{def}}{=} x$ in the remainder of the chapter.

PREDECESSORS. Given an assignment of fundamental sequences, one defines the (x-indexed) *predecessor* $P_x(\alpha) < \alpha$ of an ordinal $\alpha \neq 0$ as

ordinal predecessor

$$P_x(\alpha + 1) \overset{\text{def}}{=} \alpha \,, \qquad P_x(\lambda) \overset{\text{def}}{=} P_x(\lambda_x) \,. \tag{2.64}$$

Thus in all cases $P_x(\alpha) < \alpha$ since $\lambda_x < \lambda$. One can check that for all $\alpha > 0$ and x (see Exercise 2.9)

$$P_x(\gamma + \alpha) = \gamma + P_x(\alpha) \,. \tag{2.65}$$

Observe that predecessors of ordinals in ω^ω are very similar to our derivatives: for $d = 0$, $P_n(\omega^d) = 0$ and otherwise $P_n(\omega^d) = \omega^{d-1} \cdot (n-1) + P_n(\omega^{d-1})$, which is somewhat similar to (2.57), and more generally (2.65) is reminiscent of (2.58) but chooses a particular strategy: always derive the ω^d summand with the smallest d. The relationship will be made more precise in Section 2.4.3 on the following page.

THE CICHOŃ HIERARCHY. Fix a unary function $h\colon \mathbb{N} \to \mathbb{N}$. We define the *Cichoń hierarchy* $(h_\alpha)_{\alpha \in \varepsilon_0}$ by

Cichoń hierarchy

$$h_0(x) \overset{\text{def}}{=} 0, \qquad h_{\alpha+1}(x) \overset{\text{def}}{=} 1 + h_\alpha\big(h(x)\big), \qquad h_\lambda(x) \overset{\text{def}}{=} h_{\lambda_x}(x). \tag{2.66}$$

In the initial segment ω^ω, this hierarchy is closely related to $(M_{g,\alpha})_{\alpha \in \omega^\omega}$: indeed, we already noted the similarities between $P_n(\alpha)$ and $\partial_n \alpha$, and furthermore

Lemma 2.27. *For all $\alpha > 0$ in ε_0 and x,*

$$h_\alpha(x) = 1 + h_{P_x(\alpha)}\big(h(x)\big) \,. \tag{2.67}$$

Proof. By transfinite induction over $\alpha > 0$. For a successor ordinal $\alpha' + 1$, $h_{\alpha'+1}(x) = 1 + h_{\alpha'}\big(h(x)\big) = 1 + h_{P_x(\alpha'+1)}\big(h(x)\big)$. For a limit ordinal λ, $h_\lambda(x) = h_{\lambda_x}(x)$ is equal to $1 + h_{P_x(\lambda_x)}\big(h(x)\big)$ by ind. hyp. since $\lambda_x < \lambda$, which is the same as $1 + h_{P_x(\lambda)}\big(h(x)\big)$ by definition of $P_x(\lambda)$. □

Example 2.28 (Cichoń Hierarchy)**.** First note that $h_k(x) = k$ for all $k < \omega$, x, and h. This can be shown by induction on k: it holds for the base case $k = 0$ by definition, and also for the induction step as $h_{k+1}(x) = 1 + h_k(h(x)) = 1 + k$ by induction hypothesis. Therefore $h_\omega(x) = h_x(x) = x$ regardless of the choice of h.

For ordinals greater than ω, the choice of h becomes significant. Setting $H(x) \overset{\text{def}}{=} x + 1$, we obtain a particular hierarchy $(H_\alpha)_\alpha$ that verifies for instance

$$H_{\omega\cdot 2}(x) = H_{\omega+x}(x) = H_\omega(2x) + x = 3x\ , \tag{2.68}$$

$$H_{\omega^2}(x) = 2^x x - x\ . \tag{2.69}$$

The functions in the Cichoń hierarchy enjoy many more properties, of which we will use the following two:

Fact 2.29 (Argument Monotonicity)**.** *If h is monotone, then each h_α function is also monotone in its argument: if $x \leq x'$ then $h_\alpha(x) \leq h_\alpha(x')$.*

Fact 2.30 (Classification in the Grzegorczyk Hierarchy)**.** *Let $0 < \gamma < \omega$. If h is bounded by a function in $\mathscr{F}_\gamma$ and $\alpha < \omega^{d+1}$, then h_α is bounded by a function in $\mathscr{F}_{\gamma+d}$.*

2.4.3 Monotonicity

One obstacle subsists before we can finally prove the Length Function Theorem: the functions $M_{g,\alpha}$ and h_α are not monotone in the parameter α. Indeed, $\alpha' < \alpha$ does *not* imply $M_{g,\alpha'}(n) \leq M_{g,\alpha}(n)$ for all n: witness the case $\alpha = \omega$ and $\alpha' = n+1$: $M_{g,\omega}(n) = 1+M_{g,n-1}(g(n)) = n$ but $M_{g,n+1}(n) = n+1$ by Example 2.26. Similarly with h_α, as seen with Example 2.28, $h_{x+1}(x) = x + 1 > x = h_\omega(x)$, although $x + 1 < \omega$.

structural ordering

In our case a rather simple ordering is sufficient: we define a *structural ordering* $\sqsubseteq$ for ordinals in ω^ω by

$$\omega^{d_1} + \cdots + \omega^{d_m} \sqsubseteq \omega^{d'_1} + \cdots + \omega^{d'_n} \overset{\text{def}}{\Leftrightarrow} m \leq n \text{ and } \forall 1 \leq i \leq m,\ d_i \leq d'_i \tag{2.70}$$

for ordinal terms in CNF(ω^ω), i.e. $\omega > d_1 \geq \cdots \geq d_m \geq 0$ and $\omega > d'_1 \geq \cdots \geq d'_n \geq 0$. A useful observation is that $\sqsubseteq$ is a precongruence for $\oplus$ (see Exercise 2.10):

$$\alpha \sqsubseteq \alpha' \text{ and } \gamma \sqsubseteq \gamma' \text{ imply } \alpha \oplus \gamma \sqsubseteq \alpha' \oplus \gamma'\ . \tag{2.71}$$

The structural ordering rules out the previous examples, as $x + 1 \not\sqsubseteq \omega$ for all x. This refined ordering yields the desired monotonicity property for M—see Lemma 2.31 next (it can also be proven for h; see Exercise 2.11)—but let us first introduce some notation: we write $\alpha' = \partial_{d,n}\alpha$ if $\alpha = \gamma \oplus \omega^d$ and $\alpha' = \gamma \oplus \partial_n \omega^d$. Then (2.59) can be rewritten as

$$M_{g,\alpha}(n) = \max_{\alpha=\gamma\oplus\omega^d} \left\{1 + M_{g,\partial_{d,n}\alpha}\left(g(n)\right)\right\}. \tag{2.72}$$

Lemma 2.31 (Structural Monotonicity). *Let α, α' be in ω^ω and $x > 0$. If $\alpha \sqsubseteq \alpha'$, then $M_{g,\alpha}(x) \leq M_{g,\alpha'}(x)$.*

Proof. Let us proceed by induction. If $\alpha = 0$, then $M_{g,\alpha}(x) = 0$ and the lemma holds vacuously. Otherwise, for the induction step, write $\alpha = \sum_{i=1}^{m} \omega^{d_i}$ and $\alpha' = \sum_{j=1}^{n} \omega^{d_j}$; there is some maximizing index $1 \leq i \leq m \leq n$ such that

$$M_{g,\alpha}(x) = 1 + M_{g,\partial_{d_i,x}\alpha}\left(g(x)\right) .$$

As $i \leq n$ and $d_i \leq d'_i$, observe that $\partial_{d_i,x}\alpha \sqsubseteq \partial_{d'_i,x}\alpha'$, and by Fact 2.18, we can apply the induction hypothesis:

$$\begin{aligned} M_{g,\alpha}(x) &\leq 1 + M_{g,\partial_{d'_i,x}\alpha'}\left(g(x)\right) \\ &\leq M_{g,\alpha'}(x) . \end{aligned}$$

□

An important consequence of Lemma 2.31 is that there is a *maximizing strategy* for M, which is to always derive along the smallest term:

Lemma 2.32 (Maximizing Strategy). *If $\alpha = \gamma + \omega^d$ for some $d \geq 0$, then*

$$M_{g,\alpha}(n) = 1 + M_{g,\gamma+\partial_n\omega^d}\left(g(n)\right) . \tag{2.73}$$

Proof. Let $\alpha = \gamma \oplus \omega^{d'} \oplus \omega^d$. We claim that if $d \leq d'$ and $n \leq n'$, then

$$\partial_{d,n'}\partial_{d',n}\alpha \sqsubseteq \partial_{d',n'}\partial_{d,n}\alpha . \tag{2.74}$$

The lemma follows immediately from the claim, Lemma 2.31, and the fact that g is increasing.

The claim itself is easy to check using (2.71): abusing notations for the cases of $d = 0$ or $d' = 0$,

$$\begin{aligned} \partial_{d,n'}\partial_{d',n}\alpha &= \gamma \oplus \left(\omega^{d'-1} \cdot (d'(n-1)) + \omega^{d-1} \cdot (d(n'-1))\right) \\ \partial_{d',n'}\partial_{d,n}\alpha &= \gamma \oplus \left(\omega^{d'-1} \cdot (d'(n'-1)) + \omega^{d-1} \cdot (d(n-1))\right) . \end{aligned}$$

Observe that $d'(n-1) + d(n'-1) \leq d'(n'-1) + d(n-1)$, i.e. that the second line has at least as many terms as the first line, and thus fulfills the first condition of the structural ordering in (2.70). Furthermore, it has at least as many $w^{d'-1}$ terms, thus fulfilling the second condition of (2.70). □

Let us conclude with a comparison between derivatives and predecessors: define for $d \geq 0$ the function

$$f_d(x) \overset{\text{def}}{=} dx - d + 1 , \tag{2.75}$$

then we have the following relationship:

Corollary 2.33. *If $0 < \alpha < \omega^{d+1}$, then $M_{g,\alpha}(n) \leq 1 + M_{g,P_{f_d(n)}(\alpha)}(g(n))$.*

Proof. Since $0 < \alpha < \omega^{d+1}$, it can be written in CNF as $\alpha = \gamma + \omega^{d'}$ for some $\gamma < \alpha$ and $d' \leq d$. By Lemma 2.32, $M_{g,\alpha}(n) = 1 + M_{g,\gamma+\partial_n\omega^{d'}}(g(n))$. If $d' = 0$, i.e. $\alpha = \gamma + 1$, then

$$\gamma + \partial_n 1 = P_{f_d(n)}(\alpha) = \gamma$$

and the statement holds. Otherwise, by (2.71)

$$\begin{aligned}\gamma + \partial_n\omega^d &= \gamma + \omega^{d'-1} \cdot d'(n-1)\\ &\sqsubseteq \gamma + \omega^{d-1} \cdot d(n-1) + P_{f_d(n)}(\omega^{d-1})\\ &= P_{f_d(n)}(\alpha)\ ,\end{aligned}$$

from which we deduce the result by Lemma 2.31. □

2.4.4 Wrapping Up

We have now all the required ingredients for a proof of the Length Function Theorem. Let us start with a *uniform* upper bound on $M_{g,\alpha}$:

Theorem 2.34 (Uniform Upper Bound). *Let $d > 0$, g be a control function and select a monotone function h such that $h(f_d(x)) \geq f_d(g(x))$ for all x. If $\alpha < \omega^{d+1}$, then*

$$M_{g,\alpha}(n) \leq h_\alpha(f_d(n))\ . \tag{2.76}$$

Proof. We proceed by induction on α: if $\alpha = 0$, then $M_{g,\alpha}(n) = 0 \leq h_\alpha(f_d(n))$ for all n. Otherwise, by Corollary 2.33,

$$M_{g,\alpha}(n) \leq 1 + M_{g,P_{f_d(n)}(\alpha)}(g(x))\ .$$

Because $P_{f_d(n)}(\alpha) < \alpha$, we can apply the induction hypothesis:

$$\begin{aligned}M_{g,\alpha}(n) &\leq 1 + h_{P_{f_d(n)}(\alpha)}(f_d(g(n)))\\ &\leq 1 + h_{P_{f_d(n)}(\alpha)}(h(f_d(n)))\end{aligned}$$

since $h(f_d(n)) \geq f_d(g(n))$ and $h_{P_{f_d(n)}(\alpha)}$ is monotone by Fact 2.29. Finally, by Lemma 2.27,

$$M_{g,\alpha}(n) \leq h_\alpha(f_d(n))\ .$$ □

For instance, for $\alpha = \omega$ (and thus $f_d(x) = x$), Theorem 2.34 yields that

$$M_{g,\omega}(n) \leq h_\omega(n) = n \;, \tag{2.77}$$

which is optimal (recall examples 2.26 and 2.28). Other examples are $g(x) = 2x$, $g(x) = x^2$, $g(x) = 2^x$, where setting $h(x) = g(x)$ fits:

$$\text{If } g(x) \text{ is } 2x,\ x^2, \text{ or } 2^x, \text{ then } M_{g,\alpha}(n) \leq g_\alpha(f_d(n)) \;. \tag{2.78}$$

In a d-dimensional VASS with p states, sequences of configurations are controlled by $g(x) = x+b$ for some maximal increment $b > 0$, and then $h(x) = x+db$ is also a suitable choice, which verifies

$$L_{g,\mathbb{N}^d\times\Gamma_p}(n) \leq M_{g,\omega^d\cdot p}(n) \leq h_{\omega^d\cdot p}(f_d(n)) \leq F_d^{dbp}(f_d(n)) - f_d(n) \;, \tag{2.79}$$

the latter being a function in $\mathscr{F}_d$ when d, b, p are fixed:

Corollary 2.35. *Let $g(x) = x + b$ for some $b > 0$, and fix $d, p \geq 0$. Then $L_{g,\mathbb{N}^d\times\Gamma_p}$ is bounded by a function in $\mathscr{F}_d$.*

Finally, we can choose a generic $h(x) = d(g(x+d)-2)+1$, as in the following proof of the Length Function Theorem:

Theorem 2.8 (Length Function Theorem). *Let g be a control function bounded by some function in $\mathscr{F}_\gamma$ for some $\gamma \geq 1$ and $d, p \geq 0$. Then $L_{g,\mathbb{N}^d\times\Gamma_p}$ is bounded by a function in $\mathscr{F}_{\gamma+d}$.*

Proof. Let $A \equiv \mathbb{N}^d \times \Gamma_p$. The case of $d = 0$ is handled through (2.31), which shows that $L_{g,A}$ is a constant function in $\mathscr{F}_\gamma$.

For $d > 0$ we first use Corollary 2.25:

$$L_{g,A}(n) \leq M_{g,o(A)}(n) \;. \tag{2.80}$$

Observe that $o(A) < \omega^{d+1}$, thus by Theorem 2.34,

$$L_{g,A}(n) \leq h_{o(A)}(f_d(n)) \;, \tag{2.81}$$

where $h(x) = d \cdot (g(x + d) - 2) + 1$ verifies $h(f_d(x)) = h(d(x - 1) + 1) = d(g(dx+1)-2)+1 \geq d(g(dx)-1)+1 \geq d(g(x)-1)+1 = f_d(g(x))$ since g is strictly monotone and $d > 0$. Because h is defined from g using linear operations, for all $\gamma \geq 1$, g is bounded in $\mathscr{F}_\gamma$ if and only if h is bounded in $\mathscr{F}_\gamma$, and thus by Fact 2.30, $L_{g,A}$ is bounded in $\mathscr{F}_{\gamma+d}$. ☐

Remark 2.36. Note that the proof of Theorem 2.34 carries more generally for functions f with $f(x) \geq f_d(x)$: if $h(f(x)) \geq f(g(x))$, then $M_{g,\alpha}(n) \leq h_\alpha(f(n))$. Because proving $h(f_d(x)) \geq f_d(g(x))$ can be problematic, a simpler choice for f can ease the analysis.

One such example is $f(x) = dx + 1$: then, $h(x) = g(x)$ fits if g is *super-homogeneous*, i.e. if it verifies $g(dx) \geq d{\cdot}g(x)$ for all $d, x \geq 1$, and thus $g(dx+1) \geq g(dx) + 1 \geq d \cdot g(x) + 1$ since g is assumed to be strictly increasing:

super-homogeneous function

$$\text{If } g \text{ is super-homogeneous, then } M_{g,\alpha}(n) \leq g_\alpha(dn + 1) \;. \tag{2.82}$$

How good are these upper bounds? We already noted that they were optimal for $\mathbb{N}$ in (2.77), and the sequence (2.1) extracted from the successive configurations of SIMPLE was an example of a bad sequence with length function in $\mathscr{F}_3$. Exercise 2.15 generalizes SIMPLE to arbitrary dimensions d and control functions g and shows that a length $g_{\omega^d}(n)$ can be reached using the lexicographic ordering; this is very close to the upper bounds found for instance in equations (2.77), (2.78), and (2.82). The next chapter will be devoted to complexity lower bounds, showing that for many decision problems, the enormous generic upper bounds we proved here are actually unavoidable.

Exercises

Exercise 2.1 (Disjoint Sums)**.** Let $(A_1, \leq_{A_1})$ and $(A_2, \leq_{A_2})$ be two nwqos. Prove that $(A_1 + A_2, \leq_{A_1+A_2})$ is a nwqo (see (2.5–2.7)).

⋆ **Exercise 2.2** (Fast-Growing Functions)**.**

(1) Show that $F_1(x) = 2x$ and $F_2(x) = 2^x x$ (stated in (2.20)). What are the values of $F_k(0)$ depending on k?

(2) Show that each fast-growing function is strictly *expansive*, i.e. that $F_k(x) > x$ for all k and $x > 0$.

(3) Show that each fast-growing function is strictly *monotone* in its argument, i.e. that for all k and $x' > x$, $F_k(x') > F_k(x)$.

(4) Show that the fast-growing functions are monotone in the parameter k, more precisely that $F_{k+1}(x) \geq F_k(x)$ for all k and $x > 0$.

⋆ **Exercise 2.3** (Grzegorczyk Hierarchy)**.** Each class $\mathscr{F}_k$ of the *Grzegorczyk hierarchy* is formally defined as the closure of the constant *zero function* 0 and *one function* 1, the *sum function* $+\colon x_1, x_2 \mapsto x_1 + x_2$, the *projections* $\pi_i^n\colon x_1, \ldots, x_n \mapsto x_i$ for all $0 < i \leq n$, and the fast-growing function F_k, under two basic operations:

Grzegorczyk hierarchy
zero function
one function
sum function
projection function
substitution

substitution: if $h_0, h_1, \ldots, h_p$ belong to the class, then so does f if

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

limited primitive recursion

limited primitive recursion: if h_1, h_2, and h_3 belong to the class, then so does f if

$$\begin{aligned} f(0, x_1, \ldots, x_n) &= h_1(x_1, \ldots, x_n) , \\ f(y+1, x_1, \ldots, x_n) &= h_2(y, x_1, \ldots, x_n, f(y, x_1, \ldots, x_n)) , \\ f(y, x_1, \ldots, x_n) &\leq h_3(y, x_1, \ldots, x_n) . \end{aligned}$$

primitive recursion

Observe that *primitive recursion* is defined by ignoring the last *limitedness* condition in the previous definition.

(1) Define *cut-off subtraction* $x \dot{-} y$ as $x - y$ if $x \geq y$ and 0 otherwise. Show that the following functions are in $\mathscr{F}_0$: cut-off subtraction

predecessor : $x \mapsto x \dot{-} 1$,

cut-off subtraction : $x, y \mapsto x \dot{-} y$,

odd: $x \mapsto x \bmod 2$.

(2) Show that $F_j \in \mathscr{F}_k$ for all $j \leq k$.

(3) Show that, if a function $f(x_1, \ldots, x_n)$ is linear, then it belongs to $\mathscr{F}_0$. Deduce that $\mathscr{F}_0 = \mathscr{F}_1$.

(4) Show that if a function $f(x_1, \ldots, x_n)$ belongs to $\mathscr{F}_k$ for $k > 0$, then there exists a constant c in $\mathbb{N}$ s.t. for all $x_1, \ldots, x_n$, $f(x_1, \ldots, x_n) < F_k^c(\max_i x_i + 1)$. Why does that fail for $k = 0$?

(5) Deduce that F_{k+1} does not belong to $\mathscr{F}_k$ for $k > 0$.

Exercise 2.4 (Complexity of `while` Programs)**.** Consider a program like SIMPLE that consists of a loop with variables ranging over $\mathbb{Z}$ and updates of linear complexity. Assume we obtain a k-ary disjunctive termination argument like (1.10) on page 8, where we synthetized linear ranking functions ρ_j into $\mathbb{N}$ for each T_j.

What can be told on the complexity of the program itself?

Exercise 2.5 (Residuals of Cartesian Products)**.** For a nwqo A and an element $a \in A$, define the nwqo $\uparrow_A a$ (a substructure of A) by $\uparrow_A a \overset{\text{def}}{=} \{a' \in A \mid a \leq a'\}$. Thus $A/a = A \smallsetminus (\uparrow_A a)$. Prove the following:

$$A \times B/\langle a, b\rangle \not\equiv (A/a + \uparrow_B b) + (A/a \times B/b) + (\uparrow_A a \times B/b)\;, \tag{$*$}$$

$$A \times B/\langle a, b\rangle \not\equiv (A/a \times B) + (A \times B/b)\;. \tag{$\dagger$}$$

Exercise 2.6 (Derivatives)**.** Prove Equation (2.49): $\partial_n A = \{B + \partial_n \mathbb{N}^d \mid A \equiv B + \mathbb{N}^d\}$.

Exercise 2.7 (Maximal Order Types)**.** The mapping from nwqos to their maximal order types is in general not a bijection (recall $o(\Gamma_k) = o([k]) = k$ in Example 2.21). Prove that, if we restrict our attention to *polynomial nwqos*, then o is a bijection from polynomial nwqos (up to isomorphism) to CNF(ω^ω).

Exercise 2.8 (Well Foundedness of ∂)**.** Prove Fact 2.18: The relation $\partial \overset{\text{def}}{=} \bigcup_n \partial_n$ is well-founded.

Exercise 2.9 (Predecessors)**.** Prove Equation (2.65): For all $\alpha > 0$, $P_x(\gamma + \alpha) = \gamma + P_x(\alpha)$.

Exercise 2.10 (Structural Ordering)**.** Prove Equation (2.71): $\sqsubseteq$ is a precongruence for $\oplus$.

Exercise 2.11 (Structural Monotonicity)**.** Let α, α' be in ω^ω, h be a strictly monotone unary function, and $x > 0$. Prove that, if $\alpha \sqsubseteq \alpha'$, then $h_\alpha(x) \leq h_{\alpha'}(x)$. ⋆

⋆ **Exercise 2.12** (r-Bad Sequences)**.** We consider in this exercise a generalization of good sequences: a sequence $a_0, a_1, \ldots$ over a qo $(A, \leq)$ is *r-good* if we can extract an increasing subsequence of length $r+1$, i.e. if there exist $r+1$ indices $i_0 < \cdots < i_r$ s.t. $a_{i_0} \leq \cdots \leq a_{i_r}$. A sequence is *$r$-bad* if it is not r-good. Thus "good" and "bad" stand for "1-good" and "1-bad" respectively.

r-good sequence

r-bad sequence

By wqo.2 (stated on page 2), r-bad sequences over a wqo A are always finite, and using the same arguments as in Section 2.1.1, r-bad (g,n)-controlled sequences over a nwqo A have a maximal length $L_{g,r,A}(n)$. Our purpose is to show that questions about the length of r-bad sequences reduce to questions about bad sequences:

$$L_{g,r,A}(n) = L_{g,A\times\Gamma_r}(n)\;. \qquad (\ddagger)$$

(1) Show that such a maximal (g,n)-controlled r-bad sequence is $(r-1)$-good.

(2) Given a sequence $a_0, a_1, \ldots, a_\ell$ over a nwqo $(A, \leq_A, |.|_A)$, an index i is *p-good* if it starts an increasing subsequence of length $p+1$, i.e. if there exist indices $i = i_0 < \cdots < i_p$ s.t. $a_{i_0} \leq \cdots \leq a_{i_p}$. The *goodness* $\gamma(i)$ of an index i is the largest p s.t. i is p-good. Show that $L_{g,r,A}(n) \leq L_{g,A\times\Gamma_r}(n)$.

(3) Show the converse, i.e. that $L_{g,r,A}(n) \geq L_{g,A\times\Gamma_r}(n)$.

⋆ **Exercise 2.13** (Hardy Hierarchy)**.** A well-known variant of the Cichoń hierarchy is the *Hardy hierarchy* $(h^\alpha)_\alpha$ defined using a unary function $h\colon \mathbb{N} \to \mathbb{N}$ by

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

Observe that h^α is intuitively the αth (transfinite) iterate of the function h. As with the Cichoń hierarchy, one case is of particular interest: that of $(H^\alpha)_\alpha$ for $H(x) \stackrel{\text{def}}{=} x+1$. The Hardy hierarchy will be used in the following exercises and, quite crucially, in Chapter 3.

(1) Show that $H_\alpha(x) = H^\alpha(x) - x$ for all α, x. What about $h_\alpha(x)$ and $h^\alpha(x) - x$ if $h(x) > x$?

(2) Show that $h^{\gamma+\alpha}(x) = h^\gamma\big(h^\alpha(x)\big)$ for all h, γ, α, x with $\gamma + \alpha$ in CNF.

(3) Extend the fast-growing hierarchy to $(F_\alpha)_\alpha$ by $F_{\alpha+1}(x) \stackrel{\text{def}}{=} F_\alpha^{\omega_x}(x)$ and $F_\lambda(x) \stackrel{\text{def}}{=} F_{\lambda_x}(x)$. Show that $H^{\omega^\alpha}(x) = F_\alpha(x)$ for all α, x.

(4) Show that $h_{\gamma+\alpha}(x) = h_\gamma\big(h^\alpha(x)\big) + h_\alpha(x)$ for all h, γ, α, x with $\gamma + \alpha$ in CNF.

(5) Show that h_α measures the finite length of the iteration in h^α, i.e. that $h^\alpha(x) = h^{h_\alpha(x)}(x)$ for all h, α, x—which explains why the Cichoń hierarchy is also called the *length hierarchy*.

Exercise 2.14 (Finite Values in Coverability Trees)**.** Consider the Karp & Miller coverability tree of a d-dimensional VAS $\langle \mathbf{A}, \mathbf{x}_0\rangle$ with maximal increment $b = \max \mathbf{a} \in \mathbf{A}|\mathbf{a}|$, and maximal initial counter value $n = |\mathbf{x}_0|$. Show using Exercise 2.13 that the finite values in this tree are bounded by $h^{\omega^d \cdot d}(f_d(n))$ for $h(x) = x + db$.

⋆⋆ **Exercise 2.15** (Bad Lexicographic Sequences)**.** We consider in this exercise bad sequences over $\mathbb{N}^d$ for the *lexicographic ordering* $\leq_{\text{lex}}$ (with most significant element last) defined by

lexicographic ordering

$$\mathbf{x} <_{\text{lex}} \mathbf{y} \overset{\text{def}}{\Leftrightarrow} \mathbf{x}(d) < \mathbf{y}(d) \text{ or } (\mathbf{x}(d) = \mathbf{y}(d) \\ \text{and } \langle \mathbf{x}(1), \ldots, \mathbf{x}(d-1)\rangle <_{\text{lex}} \langle \mathbf{y}(1), \ldots, \mathbf{y}(d-1)\rangle) .$$

This is a *linearization* of the product ordering over $\mathbb{N}^d$; writing $\mathbb{N}^d_{\text{lex}}$ for the associated nwqo $(\mathbb{N}^d, \leq_{\text{lex}}, |.|)$, we see that

$$L_{g,\mathbb{N}^d_{\text{lex}}}(n) \leq L_{g,\mathbb{N}^d}(n)$$

for all control functions g and initial norms n.

Since $\leq_{\text{lex}}$ is linear, there is a *unique* maximal (g,n)-controlled bad sequence over $\mathbb{N}^d_{\text{lex}}$, which will be easy to measure. Our purpose is to prove that for all n,

$$L_{g,\mathbb{N}^d_{\text{lex}}}(n) = g_{\omega^d}(n) . \tag{§}$$

(1) Set $g(x) = x + 2$. Show that $L_{g,\mathbb{N}^2_{\text{lex}}}(2) = 8 = g_{\omega^2}(2)$.

(2) Show that $L_{g,\mathbb{N}^2}(2) > 8$ with the same $g(x) = x + 2$.

(3) Let $n > 0$, and write a program $\text{LEX}_d(g,n)$ with d counters $\mathbf{x}(1), \ldots, \mathbf{x}(d)$ whose configurations encode the d coordinates of the maximal (g,n)-controlled bad sequence over $\mathbb{N}^d_{\text{lex}}$, along with an additional counter $\mathtt{c}$ holding the current value of the control. The run of $\text{LEX}_d(g,n)$ should be a sequence $(\mathbf{x}_1, \mathtt{c}_1), (\mathbf{x}_2, \mathtt{c}_2), \ldots, (\mathbf{x}_\ell, \mathtt{c}_\ell)$ of pairs $(\mathbf{x}_i, \mathtt{c}_i)$ composed of a vector $\mathbf{x}_i$ in $\mathbb{N}^d$ and of a counter $\mathtt{c}_i$. For instance, with $g(x) = x + 2$, the run of $\text{LEX}_2(g, 2)$ should be

$$(\langle 1,1\rangle,4),(\langle 0,1\rangle,6),(\langle 5,0\rangle,8),(\langle 4,0\rangle,10),(\langle 3,0\rangle,12),(\langle 2,0\rangle,14),(\langle 1,0\rangle,16),(\langle 0,0\rangle,18) .$$

(4) Let $(\mathbf{x}_1, \mathtt{c}_1), (\mathbf{x}_2, \mathtt{c}_2), \ldots, (\mathbf{x}_\ell, \mathtt{c}_\ell)$ be the unique run of $\text{LEX}_d(g,n)$ for $n > 0$. Define

$$\alpha(\mathbf{x}) = \omega^{d-1} \cdot \mathbf{x}(d) + \cdots + \omega^0 \cdot \mathbf{x}(1) \tag{¶}$$

for any vector $\mathbf{x}$ in $\mathbb{N}^d$. Show that, for each $i > 0$,

$$g_{\omega^d}(n) = i + g_{\alpha(\mathbf{x}_i)}(\mathtt{c}_i) . \tag{‖}$$

(5) Deduce (§).

(6) Show that, if $(\mathbf{x}_1, \mathtt{c}_1), (\mathbf{x}_2, \mathtt{c}_2), \ldots, (\mathbf{x}_\ell, \mathtt{c}_\ell)$ is the run of $\text{LEX}_d(g,n)$ for $n > 0$, then $\mathtt{c}_\ell = g^{\omega^d}(n)$.

Bibliographic Notes

This chapter is based mostly on (Figueira et al., 2011; Schmitz and Schnoebelen, 2011). The reader will find earlier analyses of Dickson's Lemma in the works of McAloon (1984) and Clote (1986), who employ *large intervals* in a sequence and their associated Ramsey theory (Ketonen and Solovay, 1981), showing that large enough intervals would result in good sequences. Different combinatorial arguments are provided by Friedman (2001, Theorem 6.2) for bad sequences over $\mathbb{N}^d$, and Howell et al. (1986) for sequences of VASS configurations—where even tighter upper bounds are obtained for Exercise 2.14.

Complexity upper bounds have also been obtained for wqos beyond Dickson's Lemma: Schmitz and Schnoebelen (2011), from which the general framework of normed wqos and derivations is borrowed, tackle Higman's Lemma, and so do Cichoń and Tahhan Bittar (1998) and Weiermann (1994); furthermore the latter provides upper bounds for the more general Tree Theorem of Kruskal.

The hierarchy $(\mathscr{F}_k)_{k\geq 2}$ described as the Grzegorczyk hierarchy in Section 2.1.3 and Section 2.4 is actually due to Löb and Wainer (1970); its relationship with the original Grzegorczyk hierarchy $(\mathscr{E}^k)_k$ (Grzegorczyk, 1953) is that $\mathscr{F}_k = \mathscr{E}^{k+1}$ for all $k \geq 2$. There are actually some difference between our definition of $(F_k)_k$ and that of Löb and Wainer (1970), but it only impacts low indices $k < 2$, and our definition follows contemporary presentations. Maximal order types were defined by de Jongh and Parikh (1977), where the reader will find a proof of Fact 2.23. The Cichoń hierarchy was first published in (Cichoń and Tahhan Bittar, 1998), where it was called the *length hierarchy*. More material on subrecursive hierarchies can be found in textbooks (Rose, 1984; Fairtlough and Wainer, 1998; Odifreddi, 1999) and in Appendix A. Fact 2.29 is proven there as Equation (A.25), and Fact 2.30 is a consequence of lemmas A.6, A.9, and A.16.

3

COMPLEXITY LOWER BOUNDS

The previous chapter has established some very high complexity upper bounds on algorithms that rely on Dickson's Lemma over d-tuples of natural numbers for termination. The Length Function Theorem shows that these bounds can be found in every level of the Grzegorczyk hierarchy when d varies, which means that these bounds are *Ackermannian* when d is part of the input.

Given how large these bounds are, one should wonder whether they are useful at all, i.e. whether there exist natural decision problems that require Ackermannian resources for their resolution. It turns out that such Ackermann complexities pop up regularly with counter systems and Dickson's Lemma—see Section B.2 for more examples. We consider in this chapter the case of lossy counter machines.

Lossy counter machines and Reset Petri nets are two computational models that can be seen as weakened versions of Minsky counter machines. This weakness explains why some problems (e.g. termination) are decidable for these two models, while they are undecidable for the Turing-powerful Minsky machines.

While these positive results have been used in the literature, there also exists a negative side that has had much more impact. Indeed, decidable verification problems for lossy counter machines are Ackermann-hard and hence cannot be answered in primitive-recursive time or space. The construction can also be adapted to Reset Petri nets, incrementing counter machines, etc.

Theorem 3.1 (Hardness Theorem). *Reachability, termination and coverability for lossy counter machines are Ackermann-hard.*

Hardness Theorem|defpageidx

Termination and coverability for Reset Petri nets are Ackermann-hard.

These hardness results turn out to be relevant in several other areas; see the Bibliographic Notes at the end of the chapter.

Outline. Section 3.1 defines counter machines, both reliable and lossy. Section 3.2 builds counter machines that compute Ackermann's function. Section 3.3 puts Minsky machines *on a budget*, a gadget that is essential in Section 3.4 where the main reduction is given and the hardness of reachability and coverability for lossy counter machines is proved. We then show how to deal with reset nets in Section 3.5 and how to prove hardness of termination in Section 3.6.

3.1 Counter Machines

counter machine

Counter machines are a model of computation where a finite-state control acts upon a finite number of *counters*, i.e. storage locations that hold a natural number. The computation steps are usually restricted to simple tests and updates. For *Minsky machines*, the tests are zero-tests and the updates are increments and decrements.

Minky machine

For our purposes, it will be convenient to use a slightly extended model that allows more concise constructions, and that will let us handle reset nets smoothly.

3.1.1 Extended Counter Machines

Formally, an *extended counter machine with n counters*, often just called a *counter machine* (CM), is a tuple $M = (\mathit{Loc}, C, \Delta)$ where $\mathit{Loc} = \{\ell_1, \dots, \ell_p\}$ is a finite set of *locations*, $C = \{\mathtt{c}_1, \dots, \mathtt{c}_n\}$ is a finite set of *counters*, and $\Delta \subseteq \mathit{Loc} \times \mathit{OP}(C) \times \mathit{Loc}$ is a finite set of transition rules. The transition rules are depicted as directed edges (see figs. 3.1 to 3.3 below) between control locations labeled with an instruction $\mathit{op} \in \mathit{OP}(C)$ that is either a *guard* (a condition on the current contents of the counters for the rule to be firable), or an *update* (a method that modifies the contents of the counters), or both. For CMs, the instruction set $\mathit{OP}(C)$ is given by the following abstract grammar:

$$\begin{array}{rlllll}
\mathit{OP}(C) \ni \mathit{op} ::= & \mathtt{c=0?} & \text{/* zero test */} & \mid \mathtt{c:=0} & \text{/* reset */} \\
& \mid \mathtt{c>0?\ c--} & \text{/* decrement */} & \mid \mathtt{c=c'?} & \text{/* equality test */} \\
& \mid \mathtt{c++} & \text{/* increment */} & \mid \mathtt{c:=c'} & \text{/* copy */}
\end{array}$$

where $\mathtt{c}, \mathtt{c}'$ are any two counters in C. (We also allow a $\mathtt{no\_op}$ instruction that does not test or modify the counters.)

A *Minsky machine* is a CM that only uses instructions among zero tests, decrements and increments (the first three types). Petri nets and Vector Addition Systems with States (VASS) can be seen as counter machines that only use decrements and increments (i.e. Minsky machines without zero-tests).

3.1.2 Operational Semantics

The operational semantics of a CM $M = (\mathit{Loc}, C, \Delta)$ is given under the form of transitions between its configurations. Formally, a *configuration* (written $\sigma, \theta, \dots$)

of M is a tuple $(\ell, \mathbf{a})$ with $\ell \in \mathit{Loc}$ representing the "current" control location, and $\mathbf{a} \in \mathbb{N}^C$, a C-indexed vector of natural numbers representing the current contents of the counters. If C is some $\{\mathtt{c}_1, \ldots, \mathtt{c}_n\}$, we often write $(\ell, \mathbf{a})$ under the form $(\ell, a_1, \ldots, a_n)$. Also, we sometimes use labels in vectors of values to make them more readable, writing e.g. $\mathbf{a} = (0, \ldots, 0, \mathtt{c}_k{:}1, 0, \ldots, 0)$.

Regarding the behavior induced by the rules from Δ, there is a *transition* (also called a *step*) $\sigma \xrightarrow{\delta}_{\text{std}} \sigma'$ if, and only if, σ is some $(\ell, a_1, \ldots, a_n)$, σ' is some $(\ell', a'_1, \ldots, a'_n)$, $\Delta \ni \delta = (\ell, \mathit{op}, \ell')$ and either:

op is $\mathtt{c}_k\mathtt{=0?}$ *(zero test):* $a_k = 0$, and $a'_i = a_i$ for all $i = 1, \ldots, n$, or

op is $\mathtt{c}_k\mathtt{>0?}\ \mathtt{c}_k\mathtt{--}$ *(decrement):* $a'_k = a_k - 1$, and $a'_i = a_i$ for all $i \neq k$, or

op is $\mathtt{c}_k\mathtt{++}$ *(increment):* $a'_k = a_k + 1$, and $a'_i = a_i$ for all $i \neq k$, or

op is $\mathtt{c}_k\mathtt{:=0}$ *(reset):* $a'_k = 0$, and $a'_i = a_i$ for all $i \neq k$, or

op is $\mathtt{c}_k\mathtt{=}\mathtt{c}_p\mathtt{?}$ *(equality test):* $a_k = a_p,$ and $a'_i = a_i$ for all $i = 1, \ldots, n$, or

op is $\mathtt{c}_k\mathtt{:=}\mathtt{c}_p$ *(copy):* $a'_k = a_p$, and $a'_i = a_i$ for all $i \neq k$.

(The steps carry a "std" subscript to emphasize that we are considering the usual, standard, operational semantics of counter machines, where the behavior is *reliable*.)

As usual, we write $\sigma \xrightarrow{\Delta}_{\text{std}} \sigma'$, or just $\sigma \rightarrow_{\text{std}} \sigma'$, when $\sigma \xrightarrow{\delta}_{\text{std}} \sigma'$ for some $\delta \in \Delta$. Chains $\sigma_0 \rightarrow_{\text{std}} \sigma_1 \rightarrow_{\text{std}} \cdots \rightarrow_{\text{std}} \sigma_m$ of consecutive steps, also called *runs*, are denoted $\sigma_0 \rightarrow^*_{\text{std}} \sigma_m$, and also $\sigma_0 \rightarrow^+_{\text{std}} \sigma_m$ when $m > 0$. Steps may also be written more precisely under the form $M \vdash \sigma \rightarrow_{\text{std}} \sigma'$ when several counter machines are at hand and we want to be explicit about where the steps take place.

For a vector $\mathbf{a} = (a_1, \ldots, a_n)$, or a configuration $\sigma = (\ell, \mathbf{a})$, we let $|\mathbf{a}| = |\sigma| = \sum_{i=1}^n a_i$ denote its *size*. For $N \in \mathbb{N}$, we say that a run $\sigma_0 \rightarrow \sigma_1 \rightarrow \cdots \rightarrow \sigma_m$ is N-bounded if $|\sigma_i| \leq N$ for all $i = 0, \ldots, n$.

3.1.3 Lossy Counter Machines

Lossy counter machines (LCM) are counter machines where the contents of the counters can decrease non-deterministically (the machine can "leak", or "lose data"). lossy counter machine

Technically, it is more convenient to see lossy machines as counter machines with a different operational semantics (and not as a special class of machines): thus it is possible to use simultaneously the two semantics and relate them. Incrementing errors too are handled by introducing a different operational semantics, see Exercise 3.3.

Formally, this is defined via the introduction of a partial ordering between the configurations of M:

$$(\ell, a_1, ..., a_n) \leq (\ell', a'_1, ..., a'_n) \overset{\text{def}}{\Leftrightarrow} \ell = \ell' \wedge a_1 \leq a'_1 \wedge \cdots \wedge a_n \leq a'_n. \tag{3.1}$$

$\sigma \leq \sigma'$ can be read as "σ is σ' after some losses (possibly none)."

Now "lossy" steps, denoted $M \vdash \sigma \xrightarrow{\delta}_{\text{lossy}} \sigma'$, are given by the following definition:

$$\sigma \xrightarrow{\delta}_{\text{lossy}} \sigma' \overset{\text{def}}{\Leftrightarrow} \exists\theta, \theta', (\sigma \geq \theta \ \wedge \ \theta \xrightarrow{\delta}_{\text{std}} \theta' \ \wedge \ \theta' \geq \sigma'). \tag{3.2}$$

Note that reliable steps are a special case of lossy steps:

$$M \vdash \sigma \rightarrow_{\text{std}} \sigma' \text{ implies } M \vdash \sigma \rightarrow_{\text{lossy}} \sigma'. \tag{3.3}$$

3.1.4 Behavioral Problems on Counter Machines

We consider the following decision problems:

Reachability: given a CM M and two configurations σ_{ini} and σ_{goal}, is there a run $M \vdash \sigma_{\text{ini}} \rightarrow^* \sigma_{\text{goal}}$?

Coverability: given a CM M and two configurations σ_{ini} and σ_{goal}, is there a run $M \vdash \sigma_{\text{ini}} \rightarrow^* \sigma$ for some configuration $\sigma \geq \sigma_{\text{goal}}$ that covers σ_{goal}?

(Non-)Termination: given a CM M and a configuration σ_{ini}, is there an infinite run $M \vdash \sigma_{\text{ini}} \rightarrow \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \cdots$?

These problems are parameterized by the class of counter machines we consider and, more importantly, by the operational semantics that is assumed. Reachability and termination are decidable for lossy counter machines, i.e. counter machines assuming lossy steps, because they are well-structured. Observe that, for lossy machines, reachability and coverability coincide (except for runs of length 0). Coverability is often used to check whether a control location is reachable from some σ_{ini}. For the standard semantics, the same problems are undecidable for Minsky machines but become decidable for VASS and, except for reachability, for Reset nets (see Section 3.5).

3.2 Hardy Computations

Hardy hierarchy

The *Hardy hierarchy* $(H^\alpha \colon \mathbb{N} \rightarrow \mathbb{N})_{\alpha < \varepsilon_0}$ is a hierarchy of ordinal-indexed functions, much like the *Cichoń hierarchy* introduced in Section 2.4.2. Its definition and properties are the object of Exercise 2.13 on page 48, but let us recall the following:

$$H^0(n) \overset{\text{def}}{=} n, \qquad H^{\alpha+1}(n) \overset{\text{def}}{=} H^\alpha(n+1), \qquad H^\lambda(n) \overset{\text{def}}{=} H^{\lambda_n}(n). \tag{3.4}$$

Observe that H^1 is the successor function, and more generally H^α is the αth iterate of the successor function, using diagonalisation to treat limit ordinals. Its relation with the *fast growing hierarchy* $(F_\alpha)_{\alpha<\varepsilon_0}$ is that

$$H^{\omega^\alpha}(n) = F_\alpha(n) \tag{3.5}$$

while its relation with the Cichoń hierarchy $(H_\alpha)_{\alpha<\varepsilon_0}$ is that

$$H^\alpha(n) = H_\alpha(n) + n\ . \tag{3.6}$$

Thus $H^\omega(n) = H^n(n) = 2n$, $H^{\omega^2}(n) = 2^n n$ is exponential, H^{ω^3} non-elementary, and H^{ω^ω} Ackermannian; in fact we set in this chapter

$$\mathit{Ack}(n) \overset{\text{def}}{=} F_\omega(n) = H^{\omega^\omega}(n) = H^{\omega^n}(n). \tag{3.7}$$

Two facts that we will need later can be deduced from (3.6) and the corresponding properties for the functions in the Cichoń hierarchy: Hardy functions are monotone in their argument:

Fact 3.2 (see Fact 2.29)**.** *If $n \leq n'$ then $H^\alpha(n) \leq H^\alpha(n')$ for all $\alpha < \varepsilon_0$.*

They are also monotone in their parameter relatively to the *structural ordering* defined in Section 2.4.3 on page 42:

Fact 3.3 (see Exercise 2.11)**.** *If $\alpha \sqsubseteq \alpha'$, then $H^\alpha(n) \leq H^{\alpha'}(n)$ for all n.*

The $(F_\alpha)_\alpha$ hierarchy provides a more abstract packaging of the main stops of the (extended) *Grzegorczyk hierarchy* and requires lighter notation than the Hardy hierarchy $(H^\alpha)_\alpha$. However, with its tail-recursive definition, the Hardy hierarchy is easier to implement as a while-program or as a counter machine. Below we weakly implement Hardy computations with CMs. Formally, a (forward) *Hardy computation* is a sequence

Hardy computation

$$\alpha_0; n_0 \to \alpha_1; n_1 \to \alpha_2; n_2 \to \cdots \to \alpha_\ell; n_\ell \tag{3.8}$$

of evaluation steps implementing the equations in (3.4) seen as left-to-right rewrite rules. It guarantees $\alpha_0 > \alpha_1 > \alpha_2 > \cdots$ and $n_0 \leq n_1 \leq n_2 \leq \cdots$ and keeps $H^{\alpha_i}(n_i)$ invariant. We say it is *complete* when $\alpha_\ell = 0$ and then $n_\ell = H^{\alpha_0}(n_0)$ (we also consider incomplete computations). A *backward* Hardy computation is obtained by using (3.4) as right-to-left rules. For instance,

$$\omega^\omega; m \to \omega^m; m \to \omega^{m-1}\cdot m; m \tag{3.9}$$

constitute the first three steps of the forward Hardy computation starting from $\omega^\omega; m$ if $m > 0$.

3.2.1 Encoding Hardy Computations

Ordinals below ω^{m+1} are easily encoded as vectors in $\mathbb{N}^{m+1}$: given a vector $\mathbf{a} = (a_m, \ldots, a_0) \in \mathbb{N}^{m+1}$, we define its associated ordinal in ω^{m+1} as

$$\alpha(\mathbf{a}) \overset{\text{def}}{=} \omega^m \cdot a_m + \omega^{m-1} \cdot a_{m-1} + \cdots + \omega^0 \cdot a_0 \; . \tag{3.10}$$

Observe that ordinals below ω^{m+1} and vectors in $\mathbb{N}^{m+1}$ are in bijection through α.

We can then express Hardy computations for ordinals below ω^{m+1} as a rewrite system $\xrightarrow{H}$ over pairs $\langle \mathbf{a}; n\rangle$ of vectors in $\mathbb{N}^{m+1}$ and natural numbers:

$$\langle a_m, \ldots, a_0 + 1; n\rangle \rightarrow \langle a_m, \ldots, a_0; n+1\rangle \; , \tag{D$_1$}$$

$$\langle a_m, \ldots, a_k + 1, \overbrace{0, \ldots, 0}^{k>0 \text{ zeroes}}; n\rangle \rightarrow \langle a_m, \ldots, a_k, n, \overbrace{0, \ldots, 0}^{k-1 \text{ zeroes}}; n\rangle \; . \tag{D$_2$}$$

The two rules (D$_1$) and (D$_2$) correspond to the successor and limit case of (3.4), respectively. Computations with these rules keep $H^{\alpha(\mathbf{a})}(n)$ invariant.

A key property of this encoding is that it is *robust* in presence of "losses." Indeed, if $\mathbf{a} \leq \mathbf{a}'$, then $\alpha(\mathbf{a}) \sqsubseteq \alpha(\mathbf{a}')$ and Fact 3.3 shows that $H^{\alpha(\mathbf{a})}(n) \leq H^{\alpha(\mathbf{a}')}(n)$. More generally, adding Fact 3.2 to the mix,

Lemma 3.4 (Robustness). *If $\mathbf{a} \leq \mathbf{a}'$ and $n \leq n'$ then $H^{\alpha(\mathbf{a})}(n) \leq H^{\alpha(\mathbf{a}')}(n')$.*

Now, $\xrightarrow{H}$ terminates since $\langle \mathbf{a}; n\rangle \xrightarrow{H} \langle \mathbf{a}'; n'\rangle$ implies $\alpha(\mathbf{a}) > \alpha(\mathbf{a}')$. Furthermore, if $\mathbf{a} \neq \mathbf{0}$, one of the rules among (D$_1$) and (D$_2$) can be applied to $\langle \mathbf{a}; n\rangle$. Hence for all $\mathbf{a}$ and n there exists some n' such that $\langle \mathbf{a}; n\rangle \xrightarrow{H}{}^{*} \langle \mathbf{0}; n'\rangle$, and then $n' = H^{\alpha(\mathbf{a})}(n)$. The reverse relation $\xrightarrow{H}{}^{-1}$ terminates as well since, in a step $\langle \mathbf{a}'; n'\rangle \xrightarrow{H}{}^{-1} \langle \mathbf{a}; n\rangle$, either n' is decreased, or it stays constant and the number of zeroes in $\mathbf{a}'$ is increased.

3.2.2 Implementing Hardy Computations with Counter Machines

Being tail-recursive, Hardy computations can be evaluated via a simple while-loop that implements the $\xrightarrow{H}$ rewriting. Fix a level $m \in \mathbb{N}$: we need $m+2$ counters, one for the n argument, and $m+1$ for the $\mathbf{a} \in \mathbb{N}^{m+1}$ argument.

We define a counter machine $M_H(m) = (\mathit{Loc}_H, C, \Delta_H)$, or M_H for short, with $C = \{\mathrm{a}_0, \mathrm{a}_1, \ldots, \mathrm{a}_m, \mathrm{n}\}$. Its rules are defined pictorially in Figure 3.1: they implement $\xrightarrow{H}$ as a loop around a central location ℓ_H, as captured by the following lemma, which relies crucially on Lemma 3.4:

Lemma 3.5 (Behavior of M_H). *For all $\mathbf{a}, \mathbf{a}' \in \mathbb{N}^{m+1}$ and $n, n' \in \mathbb{N}$:*

1. *If $\langle \mathbf{a}; n\rangle \xrightarrow{H} \langle \mathbf{a}'; n'\rangle$ then $M_H \vdash (\ell_H, \mathbf{a}, n) \rightarrow^{*}_{\mathit{std}} (\ell_H, \mathbf{a}', n')$.*

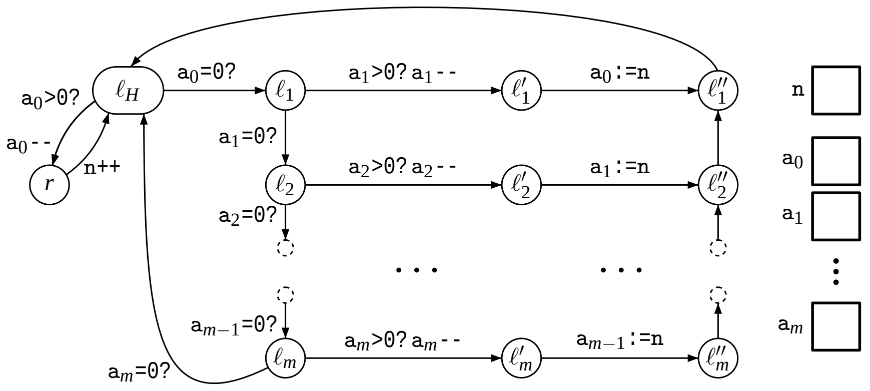

Figure 3.1: $M_H(m)$, a counter machine that implements $\xrightarrow{H}$.

2. *If* $M_H \vdash (\ell_H, \mathbf{a}, n) \to^*_{std} (\ell_H, \mathbf{a}', n')$ *then* $H^{\alpha(\mathbf{a})}(n) = H^{\alpha(\mathbf{a}')}(n')$.
3. *If* $M_H \vdash (\ell_H, \mathbf{a}, n) \to^*_{lossy} (\ell_H, \mathbf{a}', n')$ *then* $H^{\alpha(\mathbf{a})}(n) \geq H^{\alpha(\mathbf{a}')}(n')$.

The rules (D_1–D_2) can also be used from right to left. Used this way, they implement backward Hardy computations, i.e. they *invert* H. This is implemented by another counter machine, $M_{H^{-1}}(m) = (Loc_{H^{-1}}, C, \Delta_{H^{-1}})$, or $M_{H^{-1}}$ for short, defined pictorially in Figure 3.2.

$M_{H^{-1}}$ implements $\xrightarrow{H}{}^{-1}$ as a loop around a central location $\ell_{H^{-1}}$, as captured by Lemma 3.6. Note that $M_{H^{-1}}$ may deadlock if it makes the wrong guess as whether a_i contains $n+1$, but this is not a problem with the construction.

Lemma 3.6 (Behavior of $M_{H^{-1}}$). *For all* $\mathbf{a}, \mathbf{a}' \in \mathbb{N}^{m+1}$ *and* $n, n' \in \mathbb{N}$*:*

1. *If* $\langle \mathbf{a}; n\rangle \xrightarrow{H} \langle \mathbf{a}'; n'\rangle$ *then* $M_{H^{-1}} \vdash (\ell_{H^{-1}}, \mathbf{a}', n') \to^*_{std} (\ell_{H^{-1}}, \mathbf{a}, n)$.
2. *If* $M_{H^{-1}} \vdash (\ell_{H^{-1}}, \mathbf{a}, n) \to^*_{std} (\ell_{H^{-1}}, \mathbf{a}', n')$ *then* $H^{\alpha(\mathbf{a})}(n) = H^{\alpha(\mathbf{a}')}(n')$.
3. *If* $M_{H^{-1}} \vdash (\ell_{H^{-1}}, \mathbf{a}, n) \to^*_{lossy} (\ell_{H^{-1}}, \mathbf{a}', n')$ *then* $H^{\alpha(\mathbf{a})}(n) \geq H^{\alpha(\mathbf{a}')}(n')$.

3.3 Minsky Machines on a Budget

With a Minsky machine $M = (Loc, C, \Delta)$ we associate a Minsky machine $M^b = (Loc_b, C_b, \Delta_b)$. (Note that we are only considering Minsky machines here, and not the extended counter machines from earlier sections.)

M^b is obtained by adding to M an extra "budget" counter B and by adapting the rules of Δ so that any increment (resp. decrement) in the original counters is balanced by a corresponding decrement (resp. increment) on the new counter B,

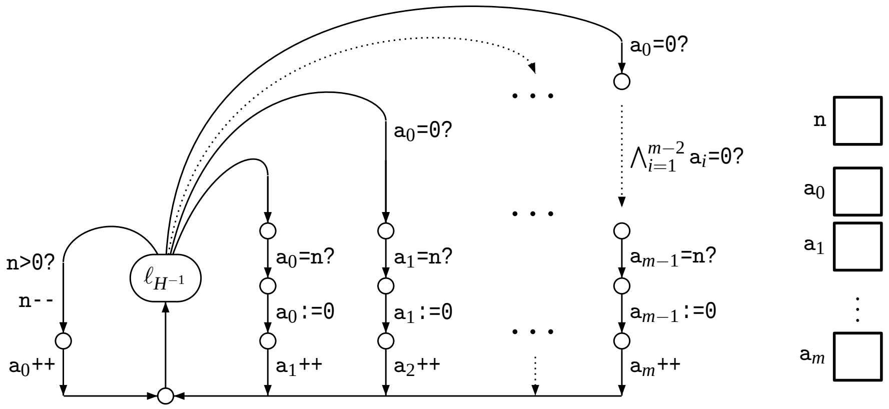

Figure 3.2: $M_{H^{-1}}(m)$, a counter machine that implements $\xrightarrow{H}{}^{-1}$.

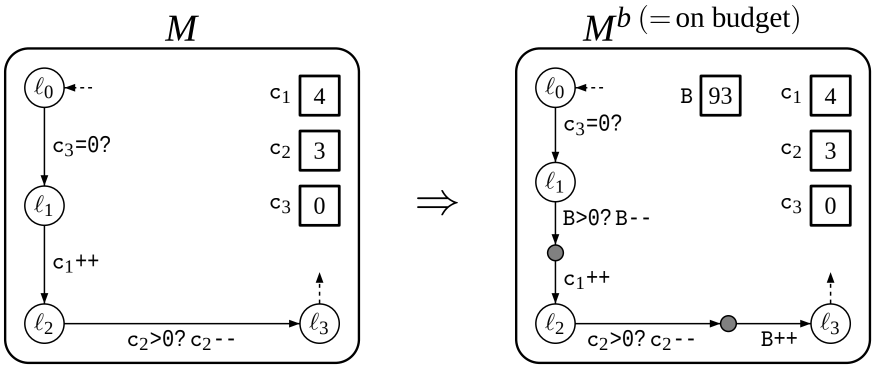

Figure 3.3: From M to M^b (schematically).

so that the sum of the counters remains constant. This is a classic idea in Petri nets. The construction is described on a schematic example (Figure 3.3) that is clearer than a formal definition. Observe that extra intermediary locations (in gray) are used, and that a rule in M that increments some $\mathtt{c}_i$ will be forbidden in M^b when the budget is exhausted.

We now collect the properties of this construction that will be used later. The fact that M^b faithfully simulates M is stated in lemmas 3.8 and 3.9. There and at other places, the restriction to "$\ell, \ell' \in Loc$" ensures that we only relate behavior anchored at the original locations in M (locations that also exist in M^b) and not at one of the new intermediary locations introduced in M^b.

First, the sum of the counters in M^b is a numerical invariant (that is only temporarily disrupted while in the new intermediary locations).

Lemma 3.7. *If* $M^b \vdash (\ell, B, \mathbf{a}) \rightarrow^*_{\mathit{std}} (\ell', B', \mathbf{a}')$ *and* $\ell, \ell' \in Loc$*, then* $B + |\mathbf{a}| = B' + |\mathbf{a}'|$.

Observe that M^b can only do what M would do:

Lemma 3.8. *If* $M^b \vdash (\ell, B, \mathbf{a}) \rightarrow^*_{\mathit{std}} (\ell', B', \mathbf{a}')$ *and* $\ell, \ell' \in \mathit{Loc}$ *then* $M \vdash (\ell, \mathbf{a}) \rightarrow^*_{\mathit{std}} (\ell', \mathbf{a}')$.

Reciprocally, everything done by M can be mirrored by M^b *provided that a large enough budget is allowed.* More precisely:

Lemma 3.9. *If* $M \vdash (\ell, \mathbf{a}) \rightarrow^*_{\mathit{std}} (\ell', \mathbf{a}')$ *is an* N*-bounded run of* M*, then* M^b *has an* N*-bounded run* $M^b \vdash (\ell, B, \mathbf{a}) \rightarrow^*_{\mathit{std}} (\ell', B', \mathbf{a}')$ *for* $B \overset{\text{def}}{=} N - |\mathbf{a}|$ *and* $B' \overset{\text{def}}{=} N - |\mathbf{a}'|$.

Now, the point of the construction is that M^b can distinguish between lossy and non-lossy runs in ways that M cannot. More precisely:

Lemma 3.10. *Let* $M^b \vdash (\ell, B, \mathbf{a}) \rightarrow^*_{\mathit{lossy}} (\ell', B', \mathbf{a}')$ *with* $\ell, \ell' \in \mathit{Loc}$*. Then* $M^b \vdash (\ell, B, \mathbf{a}) \rightarrow^*_{\mathit{std}} (\ell', B', \mathbf{a}')$ *if, and only if,* $B + |\mathbf{a}| = B' + |\mathbf{a}'|$.

Proof Idea. The "($\Leftarrow$)" direction is an immediate consequence of (3.3).

For the "($\Rightarrow$)" direction, we consider the hypothesized run $M^b \vdash (\ell, B, \mathbf{a}) = \sigma_0 \rightarrow_{\text{lossy}} \sigma_1 \rightarrow_{\text{lossy}} \cdots \rightarrow_{\text{lossy}} \sigma_n = (\ell', B', \mathbf{a}')$. Coming back to (3.2), these lossy steps require, for $i = 1, \ldots, n$, some reliable steps $\theta_{i-1} \rightarrow_{\text{std}} \theta'_i$ with $\sigma_{i-1} \geq \theta_{i-1}$ and $\theta'_i \geq \sigma_i$, and hence $|\theta'_i| \geq |\theta_i|$ for $i < n$. Combining with $|\theta_{i-1}| = |\theta'_i|$ (by Lemma 3.7), and $|\sigma_0| = |\sigma_n|$ (from the assumption that $B + |\mathbf{a}| = B' + |\mathbf{a}'|$), proves that all these configurations have same size. Hence $\theta'_i = \sigma_i = \theta_i$ and the lossy steps are also reliable steps. □

Corollary 3.11. *Assume* $M^b \vdash (\ell, B, \mathbf{0}) \rightarrow^*_{\mathit{lossy}} (\ell', B', \mathbf{a})$ *with* $\ell, \ell' \in \mathit{Loc}$*. Then:*

1. $B \geq B' + |\mathbf{a}|$*, and*
2. $M \vdash (\ell, \mathbf{0}) \rightarrow^*_{\mathit{std}} (\ell', \mathbf{a})$ *if, and only if,* $B = B' + |\mathbf{a}|$*. Furthermore, this reliable run of* M *is* B*-bounded.*

3.4 Ackermann-Hardness for Lossy Counter Machines

We now collect the ingredients that have been developed in the previous sections.

Let M be a Minsky machine with two fixed "initial" and "final" locations ℓ_{ini} and ℓ_{fin}. With M and a level $m \in \mathbb{N}$ we associate a counter machine $M(m)$ obtained by stringing together $M_H(m)$, M^b, and $M_{H^{-1}}(m)$ and fusing the extra budget counter B from M^b with the accumulator n of $M_H(m)$ and $M_{H^{-1}}(m)$ (these two share their counters). The construction is depicted in Figure 3.4.

Proposition 3.12. *The following are equivalent:*

1. $M(m)$ *has a lossy run* $(\ell_H, \mathrm{a}_m{:}1, \mathbf{0}, \mathrm{n}{:}m, \mathbf{0}) \rightarrow^*_{\mathit{lossy}} \theta$ *for some* θ *no smaller than* $(\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$.

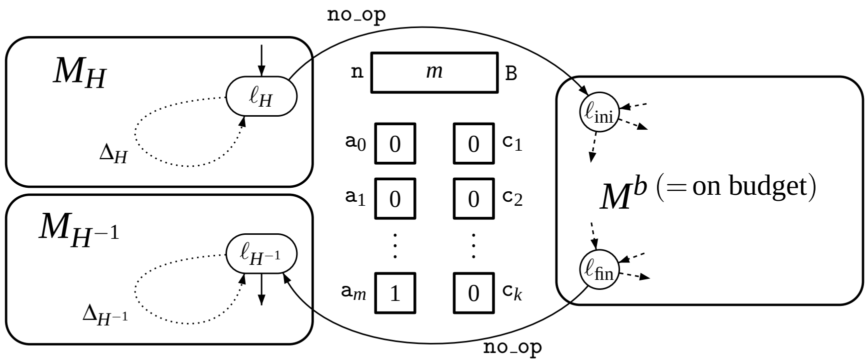

Figure 3.4: Constructing $M(m)$ from M^b, M_H and $M_{H^{-1}}$.

2. *M^b has a lossy run $(\ell_{\text{ini}}, \mathtt{B}{:}Ack(m), \mathbf{0}) \to^*_{lossy} (\ell_{\text{fin}}, Ack(m), \mathbf{0})$.*
3. *M^b has a reliable run $(\ell_{\text{ini}}, Ack(m), \mathbf{0}) \to^*_{std} (\ell_{\text{fin}}, Ack(m), \mathbf{0})$.*
4. *$M(m)$ has a reliable run $(\ell_H, 1, \mathbf{0}, m, \mathbf{0}) \to^*_{std} (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$.*
5. *M has a reliable run $(\ell_{\text{ini}}, \mathbf{0}) \to^*_{std} (\ell_{\text{fin}}, \mathbf{0})$ that is $Ack(m)$-bounded.*

Proof Sketch.

- For "*1 ⇒ 2*", and because coverability implies reachability by (3.2), we may assume that $M(m)$ has a run $(\ell_H, 1, \mathbf{0}, m, \mathbf{0}) \to^*_{\text{lossy}} (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$. This run must go through M^b and be in three parts of the following form:

$$\begin{aligned} (\ell_H, 1, \mathbf{0}, m, \mathbf{0}) &\xrightarrow{\Delta_H}{}^*_{\text{lossy}} (\ell_H, \mathbf{a}, \mathtt{n}{:}x, \mathbf{0}) && \text{(starts in } M_H) \\ \to_{\text{lossy}} (\ell_{\text{ini}}, \ldots, B, \mathbf{0}) &\xrightarrow{\Delta_b}{}^*_{\text{lossy}} (\ell_{\text{fin}}, \ldots, B', \mathbf{c}) && \text{(goes through } M^b) \\ \to_{\text{lossy}} (\ell_{H^{-1}}, \mathbf{a}', x', \ldots) &\xrightarrow{\Delta_{H^{-1}}}{}^*_{\text{lossy}} (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0}). && \text{(ends in } M_{H^{-1}}) \end{aligned}$$

  The first part yields $H^{\alpha(1,\mathbf{0})}(m) \geq H^{\alpha(\mathbf{a})}(x)$ (by Lemma 3.5.3), the third part $H^{\alpha(\mathbf{a}')}(x') \geq H^{\alpha(1,\mathbf{0})}(m)$ (by Lemma 3.6.3), and the middle part $B \geq B' + |\mathbf{c}|$ (by Corollary 3.11.1). Lossiness further implies $x \geq B$, $B' \geq x'$ and $\mathbf{a} \geq \mathbf{a}'$. Now, the only way to reconcile $H^{\alpha(\mathbf{a})}(x) \leq H^{\alpha(1,\mathbf{0})}(m) = Ack(m) \leq H^{\alpha(\mathbf{a}')}(x')$, $\mathbf{a}' \leq \mathbf{a}$, $x' \leq x$, and the monotonicity of F (Lemma 3.4) is by concluding $x = B = B' = x' = Ack(m)$ and $\mathbf{c} = \mathbf{0}$. Then the middle part of the run witnesses $M^b \vdash (\ell_{\text{ini}}, Ack(m), \mathbf{0}) \to^*_{\text{lossy}} (\ell_{\text{fin}}, Ack(m), \mathbf{0})$.

- "*2 ⇒ 5*" is Corollary 3.11.2.

- "*5 ⇒ 3*" is given by Lemma 3.9.

- "*3 ⇒ 4*" is obtained by stringing together reliable runs of the components, relying on lemmas 3.5.1 and 3.6.1 for the reliable runs of M_H and $M_{H^{-1}}$.

- Finally "*3 ⇒ 2*" and "*4 ⇒ 1*" are immediate from (3.3). □

With Proposition 3.12, we have a proof of the Hardness Theorem for reachability and coverability in lossy counter machines: Recall that, for a Minsky machine M, the existence of a run between two given configurations is undecidable, and the existence of a run bounded by $\mathit{Ack}(m)$ is decidable but not primitive-recursive when m is part of the input. Therefore, Proposition 3.12, and in particular the equivalence between its points 1 and 5, states that our construction reduces a nonprimitive-recursive problem to the reachability problem for lossy counter machines.

3.5 Handling Reset Petri Nets

Reset nets are Petri nets extended with special reset arcs that empty a place when a transition is fired. They can equally be seen as special counter machines, called *reset machines*, where actions are restricted to decrements, increments, and resets—note that zero-tests are not allowed in reset machines.

reset machine

It is known that termination and coverability are decidable for reset machines while other properties like reachability of a given configuration, finiteness of the reachability set, or recurrent reachability, are undecidable.

Our purpose is to prove the Ackermann-hardness of termination and coverability for reset machines. We start with coverability and refer to Section 3.6 for termination.

3.5.1 Replacing Zero-Tests with Resets

For a counter machine M, we let $R(M)$ be the counter machine obtained by replacing every zero-test instruction $\mathtt{c}{=}0\,?$ with a corresponding reset $\mathtt{c}:=0$. Note that $R(M)$ is a reset machine when M is a Minsky machine.

Clearly, the behavior of M and $R(M)$ are related in the following way:

Lemma 3.13.

1. *$M \vdash \sigma \rightarrow_{\mathit{std}} \sigma'$ implies $R(M) \vdash \sigma \rightarrow_{\mathit{std}} \sigma'$.*

2. *$R(M) \vdash \sigma \rightarrow_{\mathit{std}} \sigma'$ implies $M \vdash \sigma \rightarrow_{\mathit{lossy}} \sigma'$.*

In other words, the reliable behavior of $R(M)$ contains the reliable behavior of M and is contained in the lossy behavior of M.

We now consider the counter machine $M(m)$ defined in Section 3.4 and build $R(M(m))$.

Proposition 3.14. *The following are equivalent:*

1. *$R(M(m))$ has a reliable run $(\ell_H, \mathrm{a}_m{:}1, \mathbf{0}, \mathrm{n}{:}m, \mathbf{0}) \rightarrow^*_{\mathit{std}} (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$.*

2. $R(M(m))$ has a reliable run $(\ell_H, 1, \mathbf{0}, m, \mathbf{0}) \rightarrow^*_{std} \theta \geq (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$.

3. M has a reliable run $(\ell_{\text{ini}}, \mathbf{0}) \rightarrow^*_{std} (\ell_{\text{fin}}, \mathbf{0})$ that is Ack(m)-bounded.

Proof. For *1* $\Rightarrow$ *3*: The reliable run in $R(M(m))$ gives a lossy run in $M(m)$ (Lemma 3.13.2), and we conclude using "*1*⇒*5*" in Proposition 3.12.

For *3* $\Rightarrow$ *2*: We obtain a reliable run in $M(m)$ ("*5*⇒*4*" in Proposition 3.12) which gives a reliable run in $R(M(m))$ (Lemma 3.13.1), which in particular witnesses coverability.

For *2* $\Rightarrow$ *1*: The covering run in $R(M(m))$ gives a lossy covering run in $M(m)$ (Lemma 3.13.2), hence also a lossy run in $M(m)$ that reaches exactly $(\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$ (e.g. by losing whatever is required at the last step). From there we obtain a reliable run in $M(m)$ ("*1*⇒*4*" in Proposition 3.12) and then a reliable run in $R(M(m))$ (Lemma 3.13.1). ☐

We have thus reduced an Ackermann-hard problem (point *3* above) to a coverability question (point *2* above).

This almost proves the Hardness Theorem for coverability in reset machines, except for one small ingredient: $R(M(m))$ is *not* a reset machine properly because $M(m)$ is an extended counter machine, not a Minsky machine. I.e., we proved hardness for "extended" reset machines. Before tackling this issue, we want to point out that something as easy as the proof of Proposition 3.14 will prove Ackermann-hardness of reset machines by reusing the hardness of lossy counter machines.

In order to conclude the proof of the Hardness Theorem for reset machines, we only need to provide versions of M_H and $M_{H^{-1}}$ in the form of Minsky machines (M and M^b already are Minsky machines) and plug these in Figure 3.4 and Proposition 3.12.

3.5.2 From Extended to Minsky Machines

There are two reasons why we did not provide M_H and $M_{H^{-1}}$ directly under the form of Minsky machines in Section 3.2. Firstly, this would have made the construction cumbersome: Figure 3.2 is already bordering on the inelegant. Secondly, and more importantly, this would have made the proof of lemmas 3.5 and 3.6 more painful than necessary.

Rather than designing new versions of M_H and $M_{H^{-1}}$, we rely on a generic way of transforming extended counter machines into Minsky machines that preserves both the reliable behavior and the lossy behavior in a sense that is compatible with the proof of Proposition 3.12.

Formally, we associate with any extended counter machine $M = (Loc, C, \Delta)$ a new machine $M' = (Loc', C', \Delta')$ such that:

1. Loc' is Loc plus some extra "auxiliary" locations,

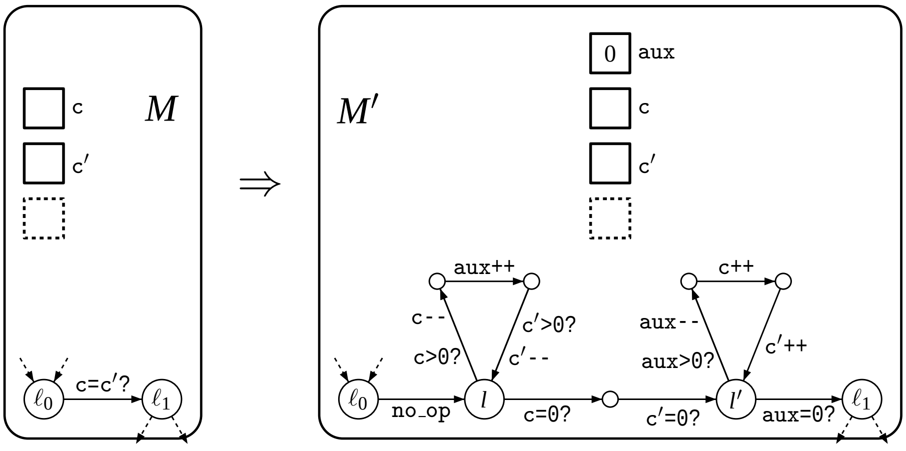

Figure 3.5: From M to M': eliminating equality tests.

2. $C' = C + \{\text{aux}\}$ is C extended with one extra counter,
3. M' only uses zero-tests, increments and decrements, hence it is a Minsky machine,
4. For any $\ell, \ell' \in \mathit{Loc}$ and vectors $\mathbf{c}, \mathbf{c}' \in \mathbb{N}^C$, the following holds:

$$M \vdash (\ell, \mathbf{c}) \to^*_{\text{std}} (\ell', \mathbf{c}') \text{ iff } M' \vdash (\ell, \mathbf{c}, 0) \to^*_{\text{std}} (\ell', \mathbf{c}', 0), \tag{3.11}$$

$$M \vdash (\ell, \mathbf{c}) \to^*_{\text{lossy}} (\ell', \mathbf{c}') \text{ iff } M' \vdash (\ell, \mathbf{c}, 0) \to^*_{\text{lossy}} (\ell', \mathbf{c}', 0). \tag{3.12}$$

The construction of M' from M contains no surprise. We replace equality tests, resets and copies by gadgets simulating them and only using the restricted instruction set of Minsky machines. One auxiliary counter aux is used for temporary storage, and several additional locations are introduced each time one extended instruction is replaced.

We show here how to eliminate equality tests and leave the elimination of resets and copies as Exercise 3.1. Figure 3.5 shows, on a schematic example, how the transformation is defined.

It is clear (and completely classic) that this transformation satisfies (3.11). The trickier half is the "$\Leftarrow$" direction. Its proof is done with the help of the following observations:

- $\mathrm{c} - \mathrm{c}'$ is a numerical invariant in l, and also in l',
- $\mathrm{c} + \mathrm{aux}$ is a numerical invariant in l, and also in l',
- when M' moves from ℓ_0 to l, aux contains 0; when it moves from l to l', both c and c′ contain 0; when it moves from l' to ℓ_1, aux contains 0.

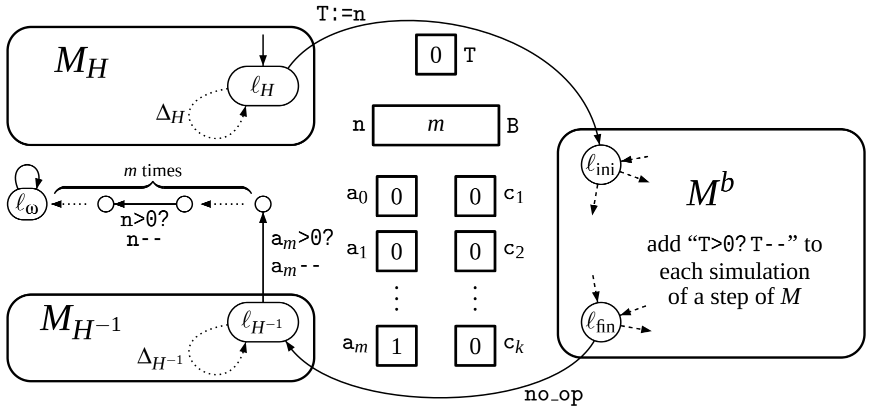

Figure 3.6: Hardness for termination: A new version of $M(m)$.

Then we also need the less standard notion of correctness from (3.12) for this transformation. The "$\Leftarrow$" direction is proved with the help of the following observations:

- $\mathtt{c} - \mathtt{c}'$ can only decrease during successive visits of l, and also of l',
- $\mathtt{c} + \mathtt{aux}$ can only decrease during successive visits of l, and also of l',
- when M' moves from ℓ_0 to l, $\mathtt{aux}$ contains 0; when it moves from l to l', both $\mathtt{c}$ and $\mathtt{c}'$ contain 0; when it moves from l' to ℓ_1, $\mathtt{aux}$ contains 0.

Gathering these observations, we can conclude that a run $M' \vdash (\ell_0, c, c', 0) \rightarrow^*_{\text{lossy}} (\ell_1, d, d', 0)$ implies $d, d' \leq \min(c, c')$. In such a case, M obviously has a lossy step $M \vdash (\ell_0, c, c') \rightarrow_{\text{lossy}} (\ell_1, d, d')$.

3.6 Hardness for Termination

We can prove hardness for termination by a minor adaptation of the proof for coverability. This adaptation, sketched in Figure 3.6, applies to both lossy counter machines and reset machines.

Basically, M^b now uses two copies of the initial budget. One copy in B works as before: its purpose is to ensure that *losses will be detected by a budget imbalance* as in Lemma 3.10. The other copy, in a new counter T, is a time limit that is initialized with n and is decremented with every simulated step of M: its purpose is to ensure that the new M^b always terminates. Since M_H and $M_{H^{-1}}$ cannot run forever (because $\xrightarrow{H}$ and $\xrightarrow{H}{}^{-1}$ terminate, see Section 3.2), we now have a new $M(m)$ that always terminate when started in ℓ_H and that satisfies the following variant of propositions 3.12 and 3.14:

Proposition 3.15. *The following are equivalent:*

1. *$M(m)$ has a lossy run $(\ell_H, 1, \mathbf{0}, \mathrm{n}{:}m, \mathbf{0}) \rightarrow^*_{\mathit{lossy}} \theta \geq (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$.*

2. *$R(M(m))$ has a lossy run $(\ell_H, 1, \mathbf{0}, \mathrm{n}{:}m, \mathbf{0}) \rightarrow^*_{\mathit{lossy}} \theta \geq (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$.*

3. *M has a reliable run $(\ell_{\mathrm{ini}}, \mathbf{0}) \rightarrow^*_{\mathit{std}} (\ell_{\mathrm{fin}}, \mathbf{0})$ of length at most $\mathit{Ack}(m)$.*

Finally, we add a series of $m+1$ transitions that leave from $\ell_{H^{-1}}$, and check that $\sigma_{\mathrm{goal}} \overset{\mathrm{def}}{=} (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$ is covered, i.e., that a_m contains at least 1 and n at least m. If this succeeds, one reaches a new location ℓ_ω, *the only place where infinite looping is allowed unconditionally.* This yields a machine $M(m)$ that has an infinite lossy run if, and only if, it can reach a configuration that covers σ_{goal}, i.e., if, and only if, M has a reliable run of length at most $\mathit{Ack}(m)$, which is an Ackermann-hard problem.

Exercises

Exercise 3.1 (From Extended to Minsky Machines)**.** Complete the translation from extended counter machines to Minsky machines given in Section 3.5.2: provide gadgets for equality tests and resets.

Exercise 3.2 (Transfer Machines)**.** *Transfer machines* are extended counter machines with instruction set reduced to increments, decrements, and *transfers*

transfer machine

$$\mathtt{c}_1\mathtt{+=c}_2\mathtt{;c}_2\mathtt{:=0}. \qquad\qquad \text{/* transfer } \mathtt{c}_2 \text{ to } \mathtt{c}_1 \text{ */}$$

Show that transfer machines can simulate reset machines as far as coverability and termination are concerned. Deduce that the Hardness Theorem also applies to transfer machines.

Exercise 3.3 (Incrementing Counter Machines)**.** *Incrementing counter machines* are Minsky machines with incrementation errors: rather than leaking, the counters may increase nondeterministically, by arbitrary large amounts. This is captured by introducing a new operations semantics for counter machines, with steps denoted $M \vdash \sigma \rightarrow_{\mathrm{inc}} \sigma'$, and defined by:

incrementing counter machine

$$\sigma \xrightarrow{\delta}_{\mathrm{inc}} \sigma' \quad \overset{\mathrm{def}}{\Leftrightarrow} \quad \exists\theta,\theta',\, (\sigma \leq \theta \;\wedge\; \theta \xrightarrow{\delta}_{\mathrm{std}} \theta' \;\wedge\; \theta' \leq \sigma'). \qquad (*)$$

Incrementation errors are thus the symmetrical mirror of losses.

Show that, for a Minksy machine M, one can construct another Minsky machine M^{-1} with

$$M \vdash \sigma_1 \rightarrow_{\mathrm{std}} \sigma_2 \text{ iff } M^{-1} \vdash \sigma_2 \rightarrow_{\mathrm{std}} \sigma_1. \qquad (\dagger)$$

What does it entail for lossy runs of M and incrementing runs of M^{-1}? Conclude that reachability for incrementing counter machines is Ackermannian.

Bibliographic Notes

This chapter is a slight revision of (Schnoebelen, 2010a), with some changes to use Hardy computations instead of fast-growing ones. Previous proofs of Ackermann-hardness for lossy counter machines or related models were published independently by Urquhart (1999) and Schnoebelen (2002).

We refer the reader to (Mayr, 2000; Schnoebelen, 2010b) for decidability issues for lossy counter machines. Reset nets (Araki and Kasami, 1976; Ciardo, 1994) are Petri nets extended with reset arcs that empty a place when the relevant transition is fired. Transfer nets (Ciardo, 1994) are instead extended with transfer arcs that move all the tokens from a place to another upon transition firing. Decidability issues for Transfer nets and Reset nets are investigated by Dufourd et al. (1999); interestingly, some problems are harder for Reset nets than for Transfer nets, although there exists an easy reduction from one to the others as far as the Hardness Theorem is concerned (see Exercise 3.2).

Using lossy counter machines, hardness results relying on the first half of the Hardness Theorem have been derived for a variety of logics and automata dealing with data words or data trees (Demri, 2006; Demri and Lazić, 2009; Jurdziński and Lazić, 2007; Figueira and Segoufin, 2009; Tan, 2010). Actually, these used reductions from counter machines with *incrementation* errors (see Exercise 3.3); although reachability for incrementing counter machines is Ackermann-hard, this does not hold for termination (Bouyer et al., 2012).

Ackermann-hardness has also been shown by reductions from Reset and Transfer nets, relying on the second half of the Hardness Theorem (e.g. Amadio and Meyssonnier, 2002; Bresolin et al., 2012).

The techniques presented in this chapter have been extended to considerably higher complexities for lossy channel systems (Chambart and Schnoebelen, 2008b) and enriched nets (Haddad et al., 2012).

Appendix

SUBRECURSIVE FUNCTIONS

Although the interested reader can easily find comprehensive accounts on subrecursive hierarchies (Rose, 1984; Fairtlough and Wainer, 1998; Odifreddi, 1999), we found it convenient to gather in this self-contained appendix many simple proofs and technical results, many too trivial to warrant being published in full, but still useful in the day-to-day work with hierarchies. We also include some results of Cichoń and Wainer (1983) and Cichoń and Tahhan Bittar (1998), which are harder to find in the literature, and the definition of lean ordinal terms.

The main thrust behind subrecursive functions is to obtain hierarchies of computable functions that lie strictly within the class of all recursive functions. An instance is the extended Grzegorczyk hierarchy $(\mathscr{F}_\alpha)_\alpha$. Such hierarchies are typically defined by generator functions and closure operators (e.g. primitive recursion, and more generally ordinal recursion), and used to draw connections with proof theory, computability, speed of growth, etc.

Our interest however lies mostly in the properties of particular functions in this theory, like the fast-growing functions $(F_\alpha)_\alpha$ or the Hardy functions $(H^\alpha)_\alpha$, which we use as tools for the study of the length of bad sequences.

A.1 Ordinal Terms

The reader is certainly familiar with the notion of *Cantor normal form* (CNF) for ordinals below ε_0, which allows to write any ordinal as an *ordinal term* α following the abstract syntax

$$\alpha ::= 0 \mid \omega^\alpha \mid \alpha + \alpha \ .$$

We take here a reversed viewpoint: our interest lies not in the "set-theoretic" ordinals, but in the set Ω of all ordinal terms. Each ordinal term α is a syntactic object, and denotes a unique ordinal $ord(\alpha)$ by interpretation into ordinal arithmetic, with $+$ denoting direct sum. Using this interpretation, we can define a well-founded ordering on terms by $\alpha' \leq \alpha$ if $ord(\alpha') \leq ord(\alpha)$. Note that the mapping of terms to ordinals is not injective, so the ordering on terms is not antisymmetric.

In this reversed viewpoint, ordinal terms might be in CNF, i.e. sums

$$\alpha = \omega^{\beta_1} + \cdots + \omega^{\beta_m}$$

with $\alpha > \beta_1 \geq \cdots \geq \beta_m \geq 0$ with each β_i in CNF itself. We also use at times the *strict* form

$$\alpha = \omega^{\beta_1} \cdot c_1 + \cdots + \omega^{\beta_m} \cdot c_m$$

where $\alpha > \beta_1 > \cdots > \beta_m \geq 0$ and $\omega > c_1, \ldots, c_m > 0$ and each β_i in strict form—we call the c_i's *coefficients*. Terms α in CNF are in bijection with their denoted ordinals $ord(\alpha)$. We write CNF(α) for the set of ordinal terms $\alpha' < \alpha$ in CNF; thus CNF(ε_0) is a subset of Ω in our view. Having a richer set Ω will be useful later in Section A.8.[1]

We write 1 for ω^0 and $\alpha \cdot n$ for $\overbrace{\alpha + \cdots + \alpha}^{n \text{ times}}$. We work modulo associativity ($(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$) and idempotence ($\alpha + 0 = \alpha = 0 + \alpha$) of $+$. An ordinal term α of form $\gamma + 1$ is called a *successor ordinal term*. Otherwise, if not 0, it is a *limit ordinal term*, usually denoted λ. Note that a $\mathrm{ord}(0) = 0$, $\mathrm{ord}(\alpha + 1)$ is a successor ordinal, and $\mathrm{ord}(\lambda)$ is a limit ordinal if λ is a limit ordinal term.

A.2 Fundamental Sequences and Predecessors

Fundamental Sequences. Subrecursive functions are defined through assignments of *fundamental sequences* $(\lambda_x)_{x<\omega}$ for limit ordinal terms λ in Ω, verifying $\lambda_x < \lambda$ for all x in $\mathbb{N}$ and $\lambda = \sup_x \lambda_x$, i.e. we are interested in a particular sequence of terms of which λ is a limit.

A standard way of obtaining fundamental sequences with good properties for every limit ordinal term λ is to fix a particular sequence $(\omega_x)_{x<\omega}$ for ω and to define

$$(\gamma + \omega^{\beta+1})_x \stackrel{\text{def}}{=} \gamma + \omega^{\beta} \cdot \omega_x\,, \qquad (\gamma + \omega^{\lambda})_x \stackrel{\text{def}}{=} \gamma + \omega^{\lambda_x}\,. \tag{A.1}$$

We assume ω_x to be the value in x of some monotone and expansive function s, typically $s(x) = x$—which we will hold as the standard one—or $s(x) = x + 1$. We will see in Section A.6 how different choices for ω_x influence the hierarchies of functions built from them, in a simple case.

[1] Richer ordinal notations can be designed, notably the *structured ordinals* of Dennis-Jones and Wainer (1984); Fairtlough and Wainer (1992) below ε_0, and of course richer notations are *required* in order to go beyond ε_0.

Predecessors. Given an assignment of fundamental sequences and x in $\mathbb{N}$, one defines the (x-indexed) *predecessor* $P_x(\alpha) < \alpha$ of an ordinal $\alpha \neq 0$ in Ω as

$$P_x(\alpha+1) \overset{\text{def}}{=} \alpha\,, \qquad\qquad P_x(\lambda) \overset{\text{def}}{=} P_x(\lambda_x)\;. \tag{A.2}$$

Lemma A.1. *Assume $\alpha > 0$ in Ω. Then for all x in $\mathbb{N}$*

$$P_x(\gamma+\alpha) = \gamma + P_x(\alpha)\,, \tag{A.3}$$

$$\textit{if } \omega_x > 0 \textit{ then } P_x(\omega^\alpha) = \omega^{P_x(\alpha)} \cdot (\omega_x - 1) + P_x(\omega^{P_x(\alpha)})\;. \tag{A.4}$$

Proof of (A.3). By induction over α. For the successor case $\alpha = \beta + 1$, this goes

$$P_x(\gamma+\beta+1) \overset{\text{(A.2)}}{=} \gamma+\beta \overset{\text{(A.2)}}{=} \gamma + P_x(\beta+1)\;.$$

For the limit case $\alpha = \lambda$, this goes

$$P_x(\gamma+\lambda) \overset{\text{(A.2)}}{=} P_x((\gamma+\lambda)_x) \overset{\text{(A.1)}}{=} P_x(\gamma+\lambda_x) \overset{ih}{=} \gamma + P_x(\lambda_x) \overset{\text{(A.2)}}{=} \gamma + P_x(\lambda)\;. \square$$

Proof of (A.4). By induction over α. For the successor case $\alpha = \beta + 1$, this goes

$$\begin{aligned} P_x(\omega^{\beta+1}) &\overset{\text{(A.2)}}{=} P_x((\omega^{\beta+1})_x) \overset{\text{(A.1)}}{=} P_x(\omega^\beta \cdot \omega_x) \overset{\text{(A.3)}}{=} \omega^\beta \cdot (\omega_x - 1) + P_x(\omega^\beta) \\ &\overset{\text{(A.2)}}{=} \omega^{P_x(\beta+1)} \cdot (\omega_x - 1) + P_x(\omega^{P_x(\beta+1)})\;. \end{aligned}$$

For the limit case $\alpha = \lambda$, this goes

$$\begin{aligned} P_x(\omega^\lambda) &\overset{\text{(A.2)}}{=} P_x((\omega^\lambda)_x) \overset{\text{(A.1)}}{=} P_x(\omega^{\lambda_x}) \overset{ih}{=} \omega^{P_x(\lambda_x)} \cdot (\omega_x - 1) + P_x(\omega^{P_x(\lambda_x)}) \\ &\overset{\text{(A.2)}}{=} \omega^{P_x(\lambda)} \cdot (\omega_x - 1) + P_x(\omega^{P_x(\lambda)})\;. \end{aligned}$$

$\square$

A.3 Pointwise Ordering and Lean Ordinals

Pointwise ordering. An issue with ordinal-indexed hierarchies is that they are typically *not* monotonic in their ordinal index. A way to circumvent this problem is to refine the ordinal ordering; an especially useful refinement is $\prec_x$ defined for $x \in \mathbb{N}$ as the smallest transitive relation satisfying (see Dennis-Jones and Wainer (1984); Fairtlough and Wainer (1992); Cichoń and Tahhan Bittar (1998)):

$$\alpha \prec_x \alpha + 1\;, \qquad\qquad \lambda_x \prec_x \lambda\;. \tag{A.5}$$

In particular, using induction on α, one immediately sees that

$$0 \preccurlyeq_x \alpha\;, \tag{A.6}$$

$$P_x(\alpha) \prec_x \alpha\;. \tag{A.7}$$

The inductive definition of $\prec_x$ implies

$$\alpha' \prec_x \alpha \text{ iff } \begin{cases} \alpha = \beta + 1 \text{ is a successor and } \alpha' \preccurlyeq_x \beta, \text{ or} \\ \alpha = \lambda \text{ is a limit and } \alpha' \preccurlyeq_x \lambda_x. \end{cases} \tag{A.8}$$

Obviously $\prec_x$ is a restriction of $<$, the strict linear quasi-ordering over ordinal terms. For example, $\omega_x \prec_x \omega$ but $\omega_x + 1 \not\prec_x \omega$, although $\mathrm{ord}(\omega_x + 1)$ is by definition a finite ordinal, smaller than $\mathrm{ord}(\omega)$.

The $\prec_x$ relations are linearly ordered themselves

$$\prec_0 \subseteq \cdots \subseteq \prec_x \subseteq \prec_{x+1} \subseteq \cdots \tag{A.9}$$

and, over terms in CNF, $<$ can be recovered by

$$\Bigl(\bigcup_{x\in\mathbb{N}} \prec_x\Bigr) = < . \tag{A.10}$$

We will soon prove these results in Corollary A.4 and Lemma A.5, but we need first some basic properties of $\prec_x$.

Lemma A.2. *For all α, α', γ in Ω and all x in $\mathbb{N}$*

$$\alpha' \prec_x \alpha \textit{ implies } \gamma + \alpha' \prec_x \gamma + \alpha\ , \tag{A.11}$$

$$\omega_x > 0 \textit{ and } \alpha' \prec_x \alpha \textit{ imply } \omega^{\alpha'} \prec_x \omega^{\alpha}\ . \tag{A.12}$$

Proof. All proofs are by induction over α (NB: the case $\alpha = 0$ is impossible).
(A.11): For the successor case $\alpha = \beta + 1$, this goes through

$$\begin{aligned} \alpha' \prec_x \beta + 1 &\text{ implies } \alpha' \preccurlyeq_x \beta && \text{(by (A.8))}\\ &\text{ implies } \gamma + \alpha' \preccurlyeq_x \gamma + \beta \overset{\text{(A.5)}}{\prec_x} \gamma + \beta + 1\ . && \text{(by ind. hyp.)}\end{aligned}$$

For the limit case $\alpha = \lambda$, this goes through

$$\begin{aligned} \alpha' \prec_x \lambda &\text{ implies } \alpha' \preccurlyeq_x \lambda_x && \text{(by (A.8))}\\ &\text{ implies } \gamma + \alpha' \preccurlyeq_x \gamma + \lambda_x \overset{\text{(A.1)}}{=} (\gamma + \lambda)_x \overset{\text{(A.5)}}{\prec_x} \gamma + \lambda\ . && \text{(by ind. hyp.)}\end{aligned}$$

(A.12): For the successor case $\alpha = \beta + 1$, we go through

$$\begin{aligned} \alpha' \prec_x \beta + 1 &\text{ implies } \alpha' \preccurlyeq_x \beta && \text{(by (A.8))}\\ &\text{ implies } \omega^{\alpha'} \preccurlyeq_x \omega^{\beta} = \omega^{\beta} + 0 && \text{(by ind. hyp.)}\\ &\text{ implies } \omega^{\alpha'} \preccurlyeq_x \omega^{\beta} + \omega^{\beta} \cdot (\omega_x - 1) \\ & && \text{(by equations (A.6) and (A.11))}\\ &\text{ implies } \omega^{\alpha'} \preccurlyeq_x \omega^{\beta} \cdot \omega_x = (\omega^{\beta+1})_x \overset{\text{(A.5)}}{\prec_x} \omega^{\beta+1}\ .\end{aligned}$$

For the limit case $\alpha = \lambda$, this goes through

$$\begin{aligned} \alpha' \prec_x \lambda &\text{ implies } \alpha' \preccurlyeq_x \lambda_x && \text{(by (A.8))}\\ &\text{ implies } \omega^{\alpha'} \preccurlyeq_x \omega^{\lambda_x} \overset{\text{(A.1)}}{=} (\omega^{\lambda})_x \overset{\text{(A.5)}}{\prec_x} \omega^{\lambda}\ . && \text{(by ind. hyp.)}\end{aligned}$$

□

Lemma A.2 shows that $\prec_x$ is left congruent for $+$ and congruent for ω-exponentiation. One can observe that it is *not* right congruent for $+$; consider for instance the terms $\omega_x + 1$ and $\omega + 1$: one can see that $\omega_x + 1 \not\prec_x \omega + 1$. Indeed, from $\omega + 1$ the only way of descending through $\succ_x$ is $\omega + 1 \succ_x \omega \succ_x \omega_x$, but $\omega_x \not\succ_x \omega_x + 1$ since $\prec_x \subseteq <$ for terms in CNF(ε_0).

Lemma A.3. *Let λ be a limit ordinal in Ω and $x < y$ in $\mathbb{N}$. Then $\lambda_x \preccurlyeq_y \lambda_y$, and if furthermore $\omega_x > 0$, then $\lambda_x \preccurlyeq_x \lambda_y$.*

Proof. By induction over λ. Write $\omega_y = \omega_x + z$ for some $z \geq 0$ by monotonicity of s (recall that ω_x and ω_y are in $\mathbb{N}$) and $\lambda = \gamma + \omega^\alpha$ with $0 < \alpha$.

If $\alpha = \beta + 1$ is a successor, then $\lambda_x = \gamma + \omega^\beta \cdot \omega_x \preccurlyeq_y \gamma + \omega^\beta \cdot \omega_x + \omega^\beta \cdot z$ by (A.11) since $0 \preccurlyeq_y \omega^\beta \cdot z$. We conclude by noting that $\lambda_y = \gamma + \omega^\beta \cdot (\omega_x + z)$; the same arguments also show $\lambda_x \preccurlyeq_x \lambda_y$.

If α is a limit ordinal, then $\alpha_x \preccurlyeq_y \alpha_y$ by ind. hyp., hence $\lambda_x = \gamma + \omega^{\alpha_x} \preccurlyeq_y \gamma + \omega^{\alpha_y} = \lambda_y$ by (A.12) (applicable since $\omega_y \geq y > x \geq 0$) and (A.11). If $\omega_x > 0$, then the same arguments show $\lambda_x \preccurlyeq_x \lambda_y$. □

Now, using (A.8) together with Lemma A.3, we see

Corollary A.4. *Let α, β in Ω and x, y in $\mathbb{N}$. If $x \leq y$ then $\alpha \prec_x \beta$ implies $\alpha \prec_y \beta$.*

In other words, $\prec_x \subseteq \prec_{x+1} \subseteq \prec_{x+2} \subseteq \cdots$ as claimed in (A.9).

If s is strictly increasing, i.e. if $\omega_x < \omega_{x+1}$ for all x, then the statement of Lemma A.3 can be strengthened to $\lambda_x \prec_y \lambda_y$ and $\lambda_y \prec_x \lambda_y$ when $\omega_x > 0$, and this hierarchy becomes strict at every level x: indeed, $\omega_{x+1} \prec_{x+1} \omega$ but $\omega_{x+1} \prec_x \omega$ would imply $\omega_{x+1} \preccurlyeq_x \omega_x$, contradicting $\prec_x \subseteq <$.

Lean Ordinals. Let k be in $\mathbb{N}$. We say that an ordinal α in CNF(ε_0) is *k-lean* if it only uses coefficients $\leq k$, or, more formally, when it is written under the strict form $\alpha = \omega^{\beta_1} \cdot c_1 + \cdots + \omega^{\beta_m} \cdot c_m$ with $c_i \leq k$ and, inductively, with k-lean β_i, this for all $i = 1, \ldots, m$. Observe that only 0 is 0-lean, and that any term in CNF is k-lean for some k.

A value k of particular importance for lean ordinal terms is $k = \omega_x - 1$: observe that this is the coefficient value introduced when we compute a predecessor ordinal at x. Stated differently, $(\omega_x - 1)$-leanness is an invariant of predecessor computations: if α is $(\omega_x - 1)$-lean, then $P_x(\alpha)$ is also $(\omega_x - 1)$-lean.

Leanness also provides a very useful characterization of the $\prec_x$ relation in terms of the ordinal ordering over terms in CNF:

Lemma A.5. *Let x be in $\mathbb{N}$, and α in* CNF(ε_0) *be $(\omega_x - 1)$-lean. Then:*

$$\alpha < \gamma \textit{ iff } \alpha \prec_x \gamma \textit{ iff } \alpha \preccurlyeq_x P_x(\gamma) \textit{ iff } \alpha \leq P_x(\gamma)\,. \tag{A.13}$$

One sees $\left(\bigcup_{x\in\mathbb{N}} \prec_x\right) = <$ over terms in CNF(ε_0) as a result of Lemma A.5. The proof relies on the syntactic characterization of the ordinal ordering over terms in CNF(ε_0) by

$$\alpha < \alpha' \Leftrightarrow \begin{cases} \alpha = 0 \text{ and } \alpha' \neq 0, \text{ or} \\ \alpha = \omega^{\beta} + \gamma, \alpha' = \omega^{\beta'} + \gamma' \text{ and} \begin{cases} \beta < \beta', \text{ or} \\ \beta = \beta' \text{ and } \gamma < \gamma'. \end{cases} \end{cases} \tag{A.14}$$

Since $\alpha \preccurlyeq_x P_x(\gamma)$ directly entails all the other statements of Lemma A.5, it is enough to prove:

Claim A.5.1. Let α, γ in CNF(ε_0) and x in $\mathbb{N}$. If α is $(\omega_x - 1)$-lean, then

$$\alpha < \gamma \text{ implies } \alpha \preccurlyeq_x P_x(\gamma)\,.$$

Proof. If $\alpha = 0$, we are done so we assume $\alpha > 0$ and hence $\omega_x > 1$, thus $\alpha = \sum_{i=1}^{m} \omega^{\beta_i} \cdot c_i$ with $m > 0$. Working with terms in CNF allows us to employ the syntactic characterization of $<$ given in (A.14).

We prove the claim by induction on γ, considering two cases:

1. if $\gamma = \gamma' + 1$ is a successor then $\alpha < \gamma$ implies $\alpha \leq \gamma'$, hence $\alpha \overset{ih}{\preccurlyeq_x} \gamma' \overset{\text{(A.2)}}{=} P_x(\gamma)$.

2. if γ is a limit, we claim that $\alpha < \gamma_x$, from which we deduce $\alpha \overset{ih}{\preccurlyeq_x} P_x(\gamma_x) \overset{\text{(A.2)}}{=} P_x(\gamma)$. We consider three subcases for the claim:
   (a) if $\gamma = \omega^{\lambda}$ with λ a limit, then $\alpha = \sum_{i=1}^{m} \omega^{\beta_i} \cdot c_i < \gamma$ implies $\beta_1 < \lambda$, hence $\beta_1 \overset{ih}{\preccurlyeq_x} P_x(\lambda) = P_x(\lambda_x) < \lambda_x$, since β_1 is $(\omega_x - 1)$-lean. Thus $\alpha < \omega^{\lambda_x} = (\omega^{\lambda})_x = \gamma_x$.
   (b) if $\gamma = \omega^{\beta+1}$ then $\alpha < \gamma$ implies $\beta_1 < \beta + 1$, hence $\beta_1 \leq \beta$. Now $c_1 \leq \omega_x - 1$ since α is $(\omega_x - 1)$-lean, hence $\alpha < \omega^{\beta_1} \cdot (c_1 + 1) \leq \omega^{\beta_1} \cdot \omega_x \leq \omega^{\beta} \cdot \omega_x = (\omega^{\beta+1})_x = \gamma_x$.
   (c) if $\gamma = \gamma' + \omega^{\beta}$ with $0 < \gamma', \beta$, then either $\alpha \leq \gamma'$, hence $\alpha < \gamma' + (\omega^{\beta})_x = \gamma_x$, or $\alpha > \gamma'$, and then α can be written as $\alpha = \gamma' + \alpha'$ with $\alpha' < \omega^{\beta}$. In that case $\alpha' \overset{ih}{\preccurlyeq_x} P_x(\omega^{\beta}) \overset{\text{(A.2)}}{=} P_x((\omega^{\beta})_x) < (\omega^{\beta})_x$, hence $\alpha = \gamma' + \alpha' \overset{\text{(A.14)}}{<} \gamma' + (\omega^{\beta})_x \overset{\text{(A.1)}}{=} (\gamma' + \omega^{\beta})_x = \gamma_x$. □

A.4 Ordinal Indexed Functions

Let us recall several classical hierarchies from (Cichoń and Wainer, 1983; Cichoń and Tahhan Bittar, 1998). All the functions we define are over natural numbers. We introduce "relativized" versions of the hierarchies, which employ a unary *control function* $h : \mathbb{N} \to \mathbb{N}$; the "standard" hierarchies then correspond to the special case where the successor function $h(x) = x + 1$ is picked. We will see later in Section A.7 how hierarchies with different control functions can be related.

Hardy Functions. We define the functions $(h^\alpha)_{\alpha\in\Omega}$, each $h^\alpha\colon \mathbb{N}\to\mathbb{N}$, by inner iteration:

$$h^0(x) \overset{\text{def}}{=} x, \qquad h^{\alpha+1}(x) \overset{\text{def}}{=} h^\alpha(h(x)), \qquad h^\lambda(x) \overset{\text{def}}{=} h^{\lambda_x}(x). \tag{A.15}$$

An example of inner iteration hierarchy is the *Hardy hierarchy* $(H^\alpha)_{\alpha\in\Omega}$ obtained from (A.15) in the special case of $h(x)=x+1$:

$$H^0(x) \overset{\text{def}}{=} x, \qquad H^{\alpha+1}(x) \overset{\text{def}}{=} H^\alpha(x+1), \qquad H^\lambda(x) \overset{\text{def}}{=} H^{\lambda_x}(x). \tag{A.16}$$

Cichoń Functions. Again for a unary h, we can define a variant $(h_\alpha)_{\alpha\in\Omega}$ of the Hardy functions called the *length hierarchy* by Cichoń and Tahhan Bittar (1998) and defined by inner and outer iteration:

$$h_0(x) \overset{\text{def}}{=} 0, \qquad h_{\alpha+1}(x) \overset{\text{def}}{=} 1 + h_\alpha(h(x)), \qquad h_\lambda(x) \overset{\text{def}}{=} h_{\lambda_x}(x). \tag{A.17}$$

As before, in the case where $h(x)=x+1$ is the successor function, this yields

$$H_0(x) \overset{\text{def}}{=} 0, \qquad H_{\alpha+1}(x) \overset{\text{def}}{=} 1 + H_\alpha(x+1), \qquad H_\lambda(x) \overset{\text{def}}{=} H_{\lambda_x}(x). \tag{A.18}$$

Those hierarchies are the most closely related to the hierarchies of functions we define for the length of bad sequences.

Fast Growing Functions. Last of all, the *fast growing functions* $(f_\alpha)_{\alpha\in\Omega}$ are defined through

$$f_0(x) \overset{\text{def}}{=} h(x), \qquad f_{\alpha+1}(x) \overset{\text{def}}{=} f_\alpha^{\omega_x}(x), \qquad f_\lambda \overset{\text{def}}{=} f_{\lambda_x}(x), \tag{A.19}$$

while its standard version (for $h(x)=x+1$) is defined by

$$F_0(x) \overset{\text{def}}{=} x+1, \qquad F_{\alpha+1}(x) \overset{\text{def}}{=} F_\alpha^{\omega_x}(x), \qquad F_\lambda(x) \overset{\text{def}}{=} F_{\lambda_x}(x). \tag{A.20}$$

Several properties of these functions can be proved by rather simple induction arguments.

Lemma A.5. *For all $\alpha>0$ in Ω and x in $\mathbb{N}$,*

$$h_\alpha(x) = 1 + h_{P_x(\alpha)}(h(x)) \;, \tag{A.21}$$

$$h^\alpha(x) = h^{P_x(\alpha)}(h(x)) = h^{P_x(\alpha)+1}(x) \;, \tag{A.22}$$

$$f_\alpha(x) = f^{\omega_x}_{P_x(\alpha)}(x) = f_{P_x(\alpha)+1}(x) \;. \tag{A.23}$$

Proof. We only prove (A.21); (A.22) and (A.23) can be proven similarly.

By transfinite induction over α. For a successor ordinal $\alpha+1$, $h_{\alpha+1}(x) = 1+h_\alpha(h(x)) = 1+h_{P_x(\alpha+1)}(h(x))$. For a limit ordinal λ, $h_\lambda(x) = h_{\lambda_x}(x) \overset{ih}{=} 1+h_{P_x(\lambda_x)}(h(x)) \overset{\text{(A.2)}}{=} 1+h_{P_x(\lambda)}(h(x))$, where the ind. hyp. can applied since $\lambda_x<\lambda$. □

Lemma A.6. *Let $h(x) > x$ for all x. Then for all α in Ω and x in $\mathbb{N}$,*

$$h_\alpha(x) \leq h^\alpha(x) - x\ .$$

Proof. By induction over α. For $\alpha = 0$, $h_0(x) = 0 = x - x = h^0(x) - x$. For $\alpha > 0$,

$$\begin{aligned} h_\alpha(x) &= 1 + h_{P_x(\alpha)}(h(x)) && \text{(by Lemma A.5)} \\ &\leq 1 + h^{P_x(\alpha)}(h(x)) - h(x) && \text{(by ind. hyp. since } P_x(\alpha) < \alpha) \\ &\leq h^{P_x(\alpha)}(h(x)) - x && \text{(since } h(x) > x) \\ &= h^\alpha(x) - x\ . && \text{(by (A.22))} \end{aligned}$$

□

Using the same argument, one can check that in particular for $h(x) = x + 1$,

$$H_\alpha(x) = H^\alpha(x) - x\ . \tag{A.24}$$

Lemma A.7. *For all α, γ in Ω, and x,*

$$h^{\gamma+\alpha}(x) = h^\gamma(h^\alpha(x))\,.$$

Proof. By transfinite induction on α. For $\alpha = 0$, $h^{\gamma+0}(x) = h^\gamma(x) = h^\gamma(h^0(x))$. For a successor ordinal $\alpha + 1$, $h^{\gamma+\alpha+1}(x) = h^{\gamma+\alpha}(h(x)) \overset{ih}{=} h^\gamma(h^\alpha(h(x))) = h^\gamma\big(h^{\alpha+1}(x)\big)$. For a limit ordinal λ, $h^{\gamma+\lambda}(x) = h^{(\gamma+\lambda)_x}(x) = h^{\gamma+\lambda_x}(x) \overset{ih}{=} h^\gamma\big(h^{\lambda_x}(x)\big) = h^\gamma\big(h^\lambda(x)\big)$. □

Remark A.8. Some care should be taken with Lemma A.7: $\gamma+\alpha$ is not necessarily a term in CNF. See Remark A.14 on page 78 for a related discussion.

Lemma A.9. *For all β in Ω, and r, x in $\mathbb{N}$,*

$$h^{\omega^\beta \cdot r}(x) = f_\beta^r(x)\ .$$

Proof. In view of Lemma A.7 and $h^0 = f^0 = Id_{\mathbb{N}}$, it is enough to prove $h^{\omega^\beta} = f_\beta$, i.e., the $r = 1$ case. We proceed by induction over β.

For the base case. $h^{\omega^0}(x) = h^1(x) \overset{(A.19)}{=} f_0(x)$.

For a successor $\beta + 1$. $h^{\omega^{\beta+1}}(x) \overset{(A.15)}{=} h^{(\omega^{\beta+1})_x}(x) = h^{\omega^\beta \cdot \omega_x}(x) \overset{ih}{=} f_\beta^{\omega_x}(x) \overset{(A.19)}{=} f_{\beta+1}(x)$.

For a limit λ. $h^{\omega^\lambda}(x) \overset{(A.15)}{=} h^{\omega^{\lambda_x}}(x) \overset{ih}{=} f_{\lambda_x}(x) \overset{(A.19)}{=} f_\lambda(x)$. □

A.5 Pointwise Ordering and Monotonicity

We set to prove in this section the main monotonicity and expansiveness properties of our various hierarchies.

Lemma A.10 (Cichoń and Tahhan Bittar, 1998). *Let h be an expansive monotone function. Then, for all α, α' in Ω and x, y in $\mathbb{N}$,*

$$x < y \textit{ implies } h_\alpha(x) \le h_\alpha(y)\;, \tag{A.25}$$

$$\alpha' \prec_x \alpha \textit{ implies } h_{\alpha'}(x) \le h_\alpha(x)\;. \tag{A.26}$$

Proof. Let us first deal with $\alpha' = 0$ for (A.26). Then $h_0(x) = 0 \le h_\alpha(x)$ for all α and x.

Assuming $\alpha' > 0$, the proof now proceeds by simultaneous transfinite induction over α.

For 0. Then $h_0(x) = 0 = h_0(y)$ and (A.26) holds vacuously since $\alpha' \prec_x \alpha$ is impossible.

For a successor $\alpha + 1$. For (A.25), $h_{\alpha+1}(x) = 1 + h_\alpha(h(x)) \overset{ih(\text{A.25})}{\le} 1 + h_\alpha(h(y)) = h_{\alpha+1}(y)$ where the ind. hyp. on (A.25) can be applied since h is monotone.

For (A.26), we have $\alpha' \preccurlyeq_x \alpha \prec_x \alpha + 1$, hence $h_{\alpha'}(x) \overset{ih(\text{A.26})}{\le} h_\alpha(x) \overset{ih(\text{A.25})}{\le} h_\alpha(h(x)) \overset{(\text{A.17})}{=} h_{\alpha+1}(x)$ where the ind. hyp. on (A.25) can be applied since $h(x) \ge x$.

For a limit λ. For (A.25), $h_\lambda(x) = h_{\lambda_x}(x) \overset{ih(\text{A.25})}{\le} h_{\lambda_x}(y) \overset{ih(\text{A.26})}{\le} h_{\lambda_y}(y) = h_\lambda(y)$ where the ind. hyp. on (A.26) can be applied since $\lambda_x \prec_y \lambda_y$ by Lemma A.3.

For (A.26), we have $\alpha' \preccurlyeq_x \lambda_x \prec_x \lambda$ with $h_{\alpha'}(x) \overset{ih(\text{A.26})}{\le} h_{\lambda_x}(x) = h_\lambda(x)$. □

Essentially the same proof can be carried out to prove the same monotonicity properties for h^α and f_α. As the monotonicity properties of f_α will be handy in the remainder of the section, we prove them now:

Lemma A.11 (Löb and Wainer, 1970). *Let h be a function with $h(x) \ge x$. Then, for all α, α' in Ω, x, y in $\mathbb{N}$ with $\omega_x > 0$,*

$$f_\alpha(x) \ge h(x) \ge x\;. \tag{A.27}$$

$$\alpha' \prec_x \alpha \textit{ implies } f_{\alpha'}(x) \le f_\alpha(x)\;, \tag{A.28}$$

$$x < y \textit{ and } h \textit{ monotone imply } f_\alpha(x) \le f_\alpha(y)\;. \tag{A.29}$$

Proof of (A.27). By transfinite induction on α. For the base case, $f_0(x) = h(x) \geq x$ by hypothesis. For the successor case, assuming $f_\alpha(x) \geq h(x)$, then by induction on $n > 0$, $f^n_\alpha(x) \geq h(x)$: for $n = 1$ it holds since $f_\alpha(x) \geq h(x)$, and for $n+1$ since $f^{n+1}_\alpha(x) = f_\alpha(f^n_\alpha(x)) \geq f_\alpha(x)$ by ind. hyp. on n. Therefore $f_{\alpha+1}(x) = f^{\omega_x}_\alpha(x) \geq x$ since $\omega_x > 0$. Finally, for the limit case, $f_\lambda(x) = f_{\lambda_x}(x) \geq x$ by ind. hyp. □

Proof of (A.28). Let us first deal with $\alpha' = 0$. Then $f_0(x) = h(x) \leq f_\alpha(x)$ for all $x > 0$ and all α by (A.27).

Assuming $\alpha' > 0$, the proof proceeds by transfinite induction over α. The case $\alpha = 0$ is impossible. For the successor case, $\alpha' \preccurlyeq_x \alpha \prec_x \alpha + 1$ with $f_{\alpha+1}(x) = f^{\omega_x - 1}_\alpha(f_\alpha(x)) \overset{\text{(A.27)}}{\geq} f_\alpha(x) \overset{ih}{\geq} f_{\alpha'}(x)$. For the limit case, we have $\alpha' \preccurlyeq_x \lambda_x \prec_x \lambda$ with $f_{\alpha'}(x) \overset{ih}{\leq} f_{\lambda_x}(x) = f_\lambda(x)$. □

Proof of (A.29). By transfinite induction over α. For the base case, $f_0(x) = h(x) \leq h(y) = f_0(y)$ since h is monotone. For the successor case, $f_{\alpha+1}(x) = f^{\omega_x}_\alpha(x) \overset{\text{(A.27)}}{\leq} f^{\omega_y}_\alpha(x) \overset{ih}{\leq} f^{\omega_y}_\alpha(y) = f_{\alpha+1}(y)$ using $\omega_x \leq \omega_y$. For the limit case, $f_\lambda(x) = f_{\lambda_x}(x) \overset{ih}{\leq} f_{\lambda_x}(y) \overset{\text{(A.28)}}{\leq} f_{\lambda_y}(y) = f_\lambda(y)$, where (A.28) can be applied thanks to Lemma A.3. □

A.6 Different Fundamental Sequences

The way we employ ordinal-indexed hierarchies is as *standard* ways of classifying the growth of functions, allowing to derive meaningful complexity bounds for algorithms relying on wqos for termination. It is therefore quite important to use a standard assignment of fundamental sequences in order to be able to compare results from different sources. The definition provided in (A.1) is standard, and the two choices $\omega_x = x$ and $\omega_x = x + 1$ can be deemed as "equally standard" in the literature. We employed $\omega_x = x$ in the rest of the notes, but the reader might desire to compare this to bounds using e.g. $\omega_x = x + 1$—as seen in Lemma A.12, this is possible for strictly increasing h.

A bit of extra notation is needed: we want to compare the Cichoń hierarchies $(h_{s,\alpha})_{\alpha\in\Omega}$ for different choices of s. Recall that s is assumed to be monotone and expansive, which is true of the identity function *id*.

Lemma A.12. *Let α in Ω. If $s(h(x)) \leq h(s(x))$ for all x, then $h_{s,\alpha}(x) \leq h_{id,\alpha}(s(x))$ for all x.*

Proof. By induction on α. For 0, $h_{s,0}(x) = 0 = h_{id,0}(s(x))$. For a successor ordinal $\alpha + 1$, $h_{s,\alpha+1}(x) = 1 + h_{s,\alpha}(h(x)) \overset{ih}{\leq} 1 + h_{id,\alpha}(s(h(x))) \overset{\text{(A.25)}}{\leq} 1 + h_{id,\alpha}(h(s(x))) = h_{id,\alpha+1}(s(x))$ since $s(h(x)) \leq h(s(x))$. For a limit ordinal

λ, $h_{s,\lambda}(x) = h_{s,\lambda_x}(x) \overset{ih}{\leq} h_{id,\lambda_x}(s(x)) \overset{\text{(A.26)}}{\leq} h_{id,\lambda_{s(x)}}(s(x)) = h_{id,\lambda}(s(x))$ where $s(x) \geq x$ implies $\lambda_x \prec_{s(x)} \lambda_{s(x)}$ by Lemma A.3 and allows to apply (A.26). □

A simple corollary of Lemma A.12 for $s(x) = x + 1$ is that, if h is strictly monotone, then $h(x+1) \geq 1 + h(x)$, and thus $h_{s,\alpha}(x) \leq h_{id,\alpha}(x+1)$, i.e. the Cichoń functions for the two classical assignments of fundamental sequences are tightly related and will always fall in the same classes of subrecursive functions. This also justifies not giving too much importance to the choice of s—within reasonable limits.

A.7 Different Control Functions

As in Section A.6, if we are to obtain bounds in terms of a *standard* hierarchy of functions, we ought to provide bounds for $h(x) = x + 1$ as control. We are now in position to prove a statement of Cichoń and Wainer (1983):

Lemma A.13. *For all γ and α in Ω, if h is monotone eventually dominated by F_γ, then f_α is eventually dominated by $F_{\gamma+\alpha}$.*

Proof. By hypothesis, there exists x_0 (which we can assume wlog. verifies $x_0 > 0$) s.t. for all $x \geq x_0$, $h(x) \leq F_\gamma(x)$. We keep this x_0 constant and show by transfinite induction on α that for all $x \geq x_0$, $f_\alpha(x) \leq F_{\gamma+\alpha}(x)$, which proves the lemma. Note that $\omega_x \geq x \geq x_0 > 0$ and thus that we can apply Lemma A.11.

For the base case 0: for all $x \geq x_0$, $f_0(x) = h(x) \leq F_\gamma(x)$ by hypothesis.

For a successor ordinal $\alpha + 1$: we first prove that for all n and all $x \geq x_0$,

$$f_\alpha^n(x) \leq F_{\gamma+\alpha}^n(x) \ . \tag{A.30}$$

Indeed, by induction on n, for all $x \geq x_0$,

$$\begin{aligned} f_\alpha^0(x) &= x = F_{\gamma+\alpha}^0(x) \\ f_\alpha^{n+1}(x) &= f_\alpha(f_\alpha^n(x)) \\ &\leq f_\alpha\big(F_{\gamma+\alpha}^n(x)\big) \qquad \text{(by (A.29) on } f_\alpha \text{ and the ind. hyp. on } n) \\ &\leq F_{\gamma+\alpha}\big(F_{\gamma+\alpha}^n(x)\big) \\ &\qquad \text{(since by (A.27) } F_{\gamma+\alpha}(x) \geq x \geq x_0 \text{ and by ind. hyp. on } \alpha) \\ &= F_{\gamma+\alpha}^{n+1}(x) \ . \end{aligned}$$

Therefore

$$\begin{aligned} f_{\alpha+1}(x) &= f_\alpha^x(x) \\ &\leq F_{\gamma+\alpha}^x(x) \qquad \text{(by (A.30) for } n = x) \\ &= F_{\gamma+\alpha+1}(x) \ . \end{aligned}$$

For a limit ordinal λ*:* for all $x \geq x_0$, $f_\lambda(x) = f_{\lambda_x}(x) \overset{ih}{\leq} F_{\gamma+\lambda_x}(x) = F_{(\gamma+\lambda)_x}(x) = F_{\gamma+\lambda}(x)$. □

Remark A.14. Observe that the statement of Lemma A.13 is one of the few instances in this appendix where ordinal term notations matter. Indeed, nothing forces $\gamma + \alpha$ to be an ordinal term in CNF. Note that, with the exception of Lemma A.5, all the definitions and proofs given in this appendix are compatible with arbitrary ordinal terms in Ω, and not just terms in CNF, so this is not a formal issue.

The issue lies in the intuitive understanding the reader might have of a term "$\gamma + \alpha$", by interpreting $+$ as the direct sum in ordinal arithmetic. This would be a mistake: in a situation where two different terms α and α' denote the same ordinal $ord(\alpha) = ord(\alpha')$, we do not necessarily have $F_\alpha(x) = F_{\alpha'}(x)$: for instance, $\alpha = \omega^{\omega^0}$ and $\alpha' = \omega^0 + \omega^{\omega^0}$ denote the same ordinal ω, but $F_\alpha(2) = F_2(2) = 2^2 \cdot 2 = 2^3$ and $F_{\alpha'}(2) = F_3(2) = 2^{2^{2} \cdot 2} \cdot 2^2 \cdot 2 = 2^{11}$. Therefore, the results on ordinal-indexed hierarchies in this appendix should be understood *syntactically* on ordinal terms, and not semantically on their ordinal denotations.

The natural question at this point is: how do these new fast growing functions compare to the functions indexed by terms in CNF? Indeed, we should check that e.g. $F_{\gamma+\omega^p}$ with $\gamma < \omega^\omega$ is multiply-recursive if our results are to be of any use. The most interesting case is the one where γ is finite but α infinite (which will be used in the proof of Lemma A.16):

Lemma A.15. *Let* $\alpha \geq \omega$ *and* $0 < \gamma < \omega$ *be in* CNF(ε_0)*, and* $\omega_x \overset{\text{def}}{=} x$*. Then, for all* x*,* $F_{\gamma+\alpha}(x) \leq F_\alpha(x+\gamma)$*.*

Proof. We first show by induction on $\alpha \geq \omega$ that

Claim A.15.1. Let $s(x) \overset{\text{def}}{=} x + \gamma$. Then for all x, $F_{id,\gamma+\alpha}(x) \leq F_{s,\alpha}(x)$.

base case for ω*:* $F_{id,\gamma+\omega}(x) = F_{id,\gamma+x}(x) = F_{s,\omega}(x)$,

successor case $\alpha + 1$*:* with $\alpha \geq \omega$, an induction on n shows that $F^n_{id,\gamma+\alpha}(x) \leq F^n_{s,\alpha}(x)$ for all n and x using the ind. hyp. on α, thus $F_{id,\gamma+\alpha+1}(x) = F^x_{id,\gamma+\alpha}(x) \overset{\text{(A.27)}}{\leq} F^{x+\gamma}_{id,\gamma+\alpha}(x) \leq F^{x+\gamma}_{s,\alpha}(x) = F_{s,\alpha+1}(x)$,

limit case $\lambda > \omega$*:* $F_{id,\gamma+\lambda}(x) = F_{id,\gamma+\lambda_x}(x) \overset{ih}{\leq} F_{s,\lambda_x}(x) \overset{\text{(A.28)}}{\leq} F_{s,\lambda_{x+\gamma}}(x) = F_{s,\lambda}(x)$ where (A.28) can be applied since $\lambda_x \prec_x \lambda_{x+\gamma}$ by Lemma A.3 (applicable since $s(x) = x + \gamma > 0$).

Returning to the main proof, note that $s(x+1) = x + 1 + \gamma = s(x) + 1$, allowing to apply Lemma A.12, thus for all x,

$$\begin{aligned}
F_{\mathit{id},\gamma+\alpha}(x) &\leq F_{s,\alpha}(x) && \text{(by the previous claim)}\\
&= H_s^{\omega^\alpha}(x) && \text{(by Lemma A.9)}\\
&\leq H_{\mathit{id}}^{\omega^\alpha}(s(x)) && \text{(by Lemma A.12 and (A.24))}\\
&= F_{\mathit{id},\alpha}(s(x)) . && \text{(by Lemma A.9)}
\end{aligned}$$

□

A.8 Classes of Subrecursive Functions

We finally consider how some natural classes of recursive functions can be characterized by closure operations on subrecursive hierarchies. The best-known of these classes is the *extended Grzegorczyk hierarchy* $(\mathscr{F}_\alpha)_{\alpha\in\mathrm{CNF}(\varepsilon_0)}$ defined by Löb and Wainer (1970) on top of the fast-growing hierarchy $(F_\alpha)_{\alpha\in\mathrm{CNF}(\varepsilon_0)}$ for $\omega_x \overset{\text{def}}{=} x$.

Let us first provide some background on the definition and properties of $\mathscr{F}_\alpha$. The class of functions $\mathscr{F}_\alpha$ is the closure of the constant, addition, projection (including identity), and F_α functions, under the operations of

substitution: if $h_0, h_1, \ldots, h_n$ belong to the class, then so does the function f defined by

$$f(x_1,\ldots,x_n) = h_0(h_1(x_1,\ldots,x_n),\ldots,h_n(x_1,\ldots,x_n)) ,$$

limited primitive recursion: if h_1, h_2, and h_3 belong to the class, then so does the function f defined by

$$\begin{aligned}
f(0,x_1,\ldots,x_n) &= h_1(x_1,\ldots,x_n) ,\\
f(y+1,x_1,\ldots,x_n) &= h_2(y,x_1,\ldots,x_n,f(y,x_1,\ldots,x_n)) ,\\
f(y,x_1,\ldots,x_n) &\leq h_3(y,x_1,\ldots,x_n) .
\end{aligned}$$

The hierarchy is strict for $\alpha > 0$, i.e. $\mathscr{F}_{\alpha'} \subsetneq \mathscr{F}_\alpha$ if $\alpha' < \alpha$, because in particular $F_{\alpha'} \notin \mathscr{F}_\alpha$. For small finite values of α, the hierarchy characterizes some well-known classes of functions:

- $\mathscr{F}_0 = \mathscr{F}_1$ contains all the linear functions, like $\lambda x.x+3$ or $\lambda x.2x$, along with many simple ones like *cut-off subtraction*: $\lambda xy.x \dot{-} y$, which yields $x-y$ if $x \geq y$ and 0 otherwise,[2] or simple predicates like *odd*: $\lambda x.x \mod 2$,[3]
- $\mathscr{F}_2$ is exactly the set of elementary functions, like $\lambda x.2^{2^x}$,

[2]By limited primitive recursion; first define $\lambda x.x \dot{-} 1$ by $0 \dot{-} 1 = 0$ and $(y+1) \dot{-} 1 = y$; then $x \dot{-} 0 = x$ and $x \dot{-} (y+1) = (x \dot{-} y) \dot{-} 1$.

[3]By limited primitive recursion: $0 \mod 2 = 0$ and $(y+1) \mod 2 = 1 \dot{-} (y \mod 2)$.

- $\mathscr{F}_3$ contains all the tetration functions, like $\lambda x.\ \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{x \text{ times}}$, etc.

The union $\bigcup_{\alpha<\omega} \mathscr{F}_\alpha$ is the set of primitive-recursive functions, while F_ω is an Ackermann-like non primitive-recursive function. Similarly, $\bigcup_{\alpha<\omega^\omega} \mathscr{F}_\alpha$ is the set of multiply-recursive functions with F_{ω^ω} a non multiply-recursive function.

The following properties (resp. Theorem 2.10 and Theorem 2.11 in (Löb and Wainer, 1970)) are useful: for all α, unary f in $\mathscr{F}_\alpha$, and x,

$$\alpha > 0 \text{ implies } \exists p,\ f(x) \leq F^p_\alpha(x+1)\ , \tag{A.31}$$

$$\exists p,\ \forall x \geq p,\ f(x) \leq F_{\alpha+1}(x)\ . \tag{A.32}$$

Also note that by (A.31), if a unary function g is dominated by some function g' in $\mathscr{F}_\alpha$ with $\alpha > 0$, then there exists p s.t. for all x, $g(x) \leq g'(x) \leq F^p_\alpha(x+1)$. Similarly, (A.32) shows that for all $x \geq p$, $g(x) \leq g'(x) \leq F_{\alpha+1}(x)$.

Let us conclude this appendix with the following lemma, which shows that the difficulties raised by non-CNF ordinal terms (recall Remark A.14) are alleviated when working with the $(\mathscr{F}_\alpha)_\alpha$:

Lemma A.16. *For all $\gamma > 0$ and α, if h is monotone and eventually dominated by a function in $\mathscr{F}_\gamma$, then*

1. *if $\alpha < \omega$, f_α is dominated by a function in $\mathscr{F}_{\gamma+\alpha}$, and*
2. *if $\gamma < \omega$ and $\alpha \geq \omega$, f_α is dominated by a function in $\mathscr{F}_\alpha$.*

Proof of 1. We proceed by induction on $\alpha < \omega$.

For the base case $\alpha = 0$: we have $f_0 = h$ dominated by a function in $\mathscr{F}_\gamma$ by hypothesis.

For the successor case $\alpha = k+1$: by ind. hyp. f_k is dominated by a function in $\mathscr{F}_{\gamma+k}$, thus by (A.31) there exists p s.t. $f_k(x) \leq F^p_{\gamma+k}(x+1) = F^p_{\gamma+k} \circ F_0(x)$. By induction on n, we deduce

$$f^n_k(x) \leq (F^p_{\gamma+k} \circ F_0)^n(x)\ ; \tag{A.33}$$

Therefore,

$$\begin{aligned}
f_{k+1}(x) &= f^x_k(x) && \text{(A.34)}\\
&\overset{\text{(A.33)}}{\leq} (F^p_{\gamma+k} \circ F_0)^x(x) && \text{(A.35)}\\
&\overset{\text{(A.29)}}{\leq} F^{(p+1)x+1}_{\gamma+k}((p+1)x+1) && \text{(A.36)}\\
&= F_{\gamma+k+1}((p+1)x+1)\ ,
\end{aligned}$$

where the latter function $x \mapsto F_{\gamma+k+1}((p+1)x+1)$ is defined by substitution from $F_{\gamma+k+1}$, successor, and $(p+1)$-fold addition, and therefore belongs to $\mathscr{F}_{\gamma+k+1}$. □

Proof of 2. By (A.32), there exists x_0 s.t. for all $x \geq x_0$, $h(x) \leq F_{\gamma+1}(x)$. By lemmas A.13 and A.15, $f_\alpha(x) \overset{\text{(A.29)}}{\leq} f_\alpha(x + x_0) \leq F_\alpha(x + x_0 + \gamma + 1)$ for all x, where the latter function $x \mapsto F_\alpha(x + x_0 + \gamma + 1)$ is in $\mathscr{F}_\alpha$. □

BESTIARY

PROBLEMS OF ENORMOUS COMPLEXITY

Because their main interest lies in characterizing which problems are efficiently solvable, most textbooks in complexity theory concentrate on the frontiers between tractability and intractability, with less interest for the "truly intractable" problems found in ExpTime and beyond. Unfortunately, many natural decision problems are not that tame and require to explore the uncharted classes outside the exponential hierarchy.

This appendix borrows its title from a survey by Friedman (1999), where the reader will find many problems living outside Elementary. We are however not interested in "creating" new problems of enormous complexity, but rather in classifying already known problems in some important stops related to the extended Grzegorczyck hierarchy. Because we wanted this appendix to be reasonably self-contained, we will recall several definitions found elsewhere in these notes.

B.1 Fast-Growing Complexities

Exponential Hierarchy. Let us start where most accounts on complexity stop: define the class of exponential-time problems as

$$\textsc{ExpTime} \stackrel{\text{def}}{=} \bigcup_c \textsc{DTime}\left(2^{n^c}\right)$$

and the corresponding nondeterministic and space-bounded classes as

$$\textsc{NExpTime} \stackrel{\text{def}}{=} \bigcup_c \textsc{NTime}\left(2^{n^c}\right)$$
$$\textsc{ExpSpace} \stackrel{\text{def}}{=} \bigcup_c \textsc{Space}\left(2^{n^c}\right) .$$

Problems complete for ExpTime, like corridor tiling games (Chlebus, 1986) or equivalence of regular tree languages (Seidl, 1990), are *known* not to be in PTime, hence the denomination "truly intractable" or "provably intractable" in the literature.

We can generalize these classes of problems to the *exponential hierarchy*

$$k\text{-}\textsc{ExpTime} \overset{\text{def}}{=} \bigcup_c \textsc{DTime}\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{n^c}}}}}}_{k \text{ times}}\right) ,$$

with the nondeterministic and space-bounded variants defined accordingly. The union of the classes in this hierarchy is the class of *elementary* problems:

$$\textsc{Elementary} \overset{\text{def}}{=} \bigcup_k k\text{-}\textsc{ExpTime} = \bigcup_c \textsc{DTime}\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{n}}}}}}_{c \text{ times}}\right) .$$

Note that we could as easily define Elementary in terms of nondeterministic time bounds, space bounds, alternation classes, etc. Our interest in this appendix lies in the problems found outside this class, for which suitable hierarchies need to be used.

The Extended Grzegorczyk Hierarchy $(\mathscr{F}_\alpha)_{\alpha<\varepsilon_0}$ is an infinite hierarchy of classes of functions f with argument(s) and images in $\mathbb{N}$ (Löb and Wainer, 1970). 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 by

$$F_0(x) \overset{\text{def}}{=} x+1 , \qquad F_{\alpha+1}(x) \overset{\text{def}}{=} F_\alpha^x(x) = \overbrace{F_\alpha(F_\alpha(\cdots F_\alpha}^{x \text{ times}}(x))) ,$$
$$F_\lambda(x) \overset{\text{def}}{=} F_{\lambda_x}(x) ,$$

where $\lambda_x < \lambda$ is the xth element of the *fundamental sequence* for the limit ordinal λ, defined by

$$(\gamma + \omega^{\beta+1})_x \overset{\text{def}}{=} \gamma + \omega^\beta \cdot x , \quad (\gamma + \omega^\lambda)_x \overset{\text{def}}{=} \gamma + \omega^{\lambda_x} .$$

For instance,

$$F_1(x) = 2x , \qquad F_2(x) = 2^x x ,$$
$$F_3(x) > 2^{\cdot^{\cdot^{\cdot^{2}}}}\Big\}{\scriptstyle x \text{ times}} ,$$

F_ω is an Ackermannian function,

F_{ω^ω} is a hyper-Ackermannian function, etc.

For $\alpha \geq 2$, each level of the extended Grzegorczyk hierarchy can be characterized as a class of functions computable with bounded resources

$$\mathscr{F}_\alpha = \bigcup_c \text{FDT}\textsc{ime}\left(F_\alpha^c(n)\right), \tag{B.1}$$

the choice between deterministic and nondeterministic or between time-bounded and space-bounded computations being once more irrelevant because F_2 is already a function of exponential growth. In particular, F_α^c belongs to $\mathscr{F}_\alpha$ for every α and fixed c.

Every function f in $\mathscr{F}_\alpha$ is *honest*, i.e. can be computed in time elementary in itself (Wainer, 1970)—this is a variant of the *time constructible* or *proper complexity* functions found in the literature, but better suited for the high complexities we are considering. Every f is also eventually bounded by $F_{\alpha'}$ if $\alpha < \alpha'$, i.e. there exists a rank $x_{f,\alpha}$ s.t. for all $x_1, \ldots, x_n$, if $\max_i x_i \geq x_{f,\alpha}$, then $f(x_1, \ldots, x_n) \leq F_{\alpha'}(\max_i x_i)$. However, for all $\alpha' > \alpha > 0$, $F_{\alpha'} \notin \mathscr{F}_\alpha$, and the hierarchy $(\mathscr{F}_\alpha)_{\alpha<\varepsilon_0}$ is strict for $\alpha > 0$.

Important Stops. Although some deep results have been obtained on the lower classes,[1] we focus here on the non-elementary classes, i.e. on $\alpha \geq 2$, where we find for instance

$$\begin{aligned}
\mathscr{F}_2 &= \text{FE}\textsc{lementary}\ ,\\
\bigcup_k \mathscr{F}_k &= \text{FP}\textsc{rimitive-}\text{R}\textsc{ecursive}\ ,\\
\bigcup_k \mathscr{F}_{\omega^k} &= \text{FM}\textsc{ultiply-}\text{R}\textsc{ecursive}\ ,\\
\bigcup_{\alpha<\varepsilon_0} \mathscr{F}_\alpha &= \text{FO}\textsc{rdinal-}\text{R}\textsc{ecursive}\ .
\end{aligned}$$

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, we obtain the classification of decision problems displayed in Figure B.1.

Unfortunately, these classes are not quite satisfying for some interesting problems, which are *non* elementary (resp. non primitive-recursive, or non multiply-recursive, ...), but only *barely* so. The issue is that complexity classes like e.g. $\mathscr{F}_3^*$, which is the first class that contains non-elementary problems, are very large: $\mathscr{F}_3^*$ contains for instance problems that require space F_3^{100}, more than a hundred-fold compositions of towers of exponentials. As a result, hardness for $\mathscr{F}_3$ cannot be obtained for the classical examples of non-elementary problems.

[1]See Ritchie (1963) for a characterization of FLinSpace, and for variants see e.g. Cobham (1965); Bellantoni and Cook (1992) for FPTime, or the chapter by Clote (1999) for a survey of these techniques.

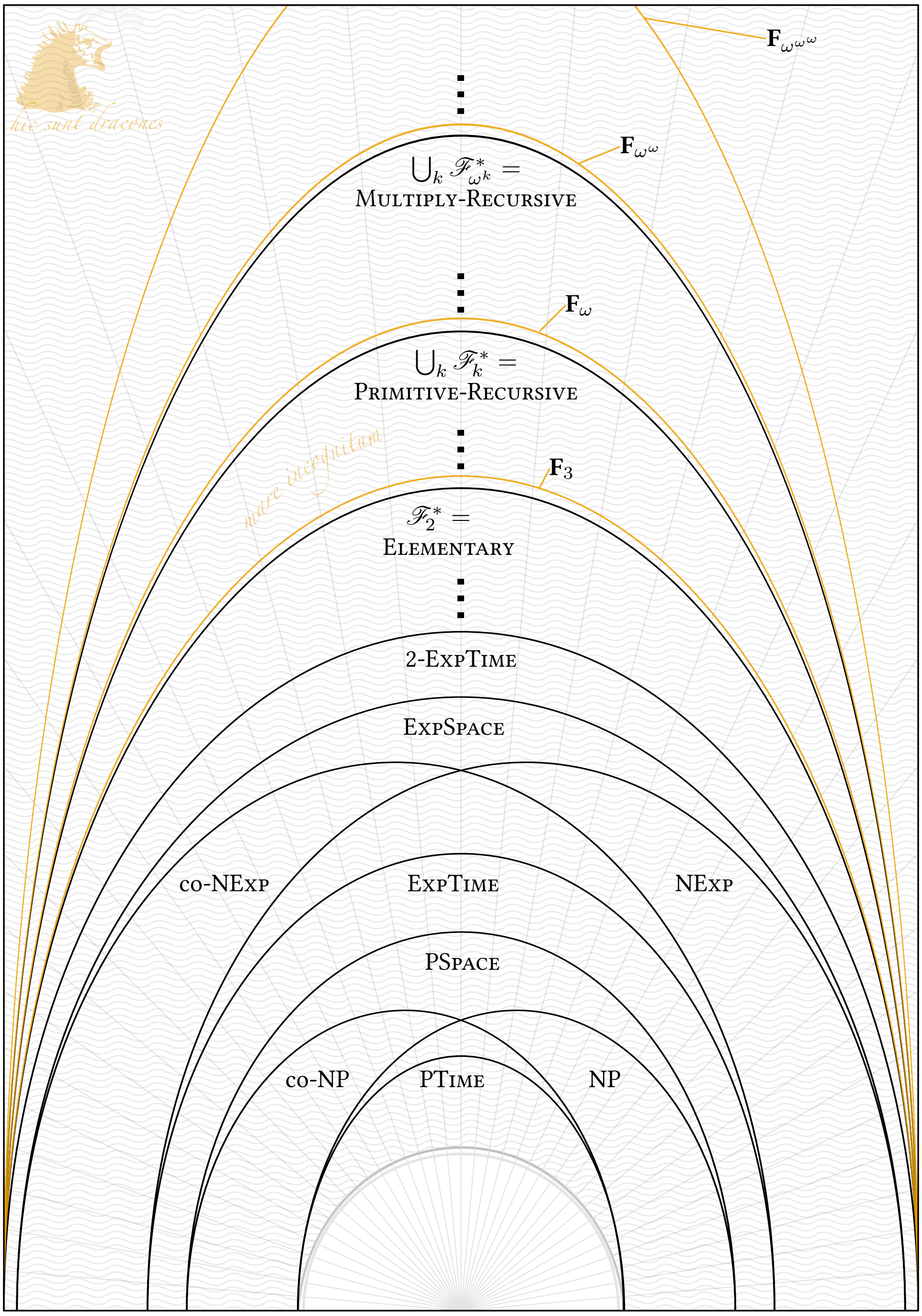

Figure B.1: Some complexity classes.

We therefore introduce *smaller* classes:

$$\mathbf{F}_\alpha \overset{\text{def}}{=} \bigcup_{p \in \cup_{\beta<\alpha} \mathscr{F}_\beta} \text{DT}\textsc{ime}\left(F_\alpha(p(n))\right) \ . \tag{B.2}$$

As previously, the choice of DTIME rather than NTIME or SPACE or ATIME is irrelevant for $\alpha \geq 3$. This yields for instance a class $\mathbf{F}_3$ of non-elementary decision problems closed under elementary reductions, a class $\mathbf{F}_\omega$ of Ackermannian problems closed under primitive-recursive reductions, a class $\mathbf{F}_{\omega^\omega}$ of hyper-Ackermannian problems closed under multiply-recursive reductions, etc.[2] We can name a few of these complexity classes:

$$\mathbf{F}_\omega = \textsc{Ackermannian} \ ,$$
$$\mathbf{F}_{\omega^\omega} = \textsc{Hyper-Ackermannian} \ .$$

Of course, we could replace in (B.2) the class of reductions $\cup_{\beta<\alpha}\mathscr{F}_\beta$ by a more traditional one, like FLOGSPACE or FPTIME, or for $\alpha \geq \omega$ by primitive-recursive reductions in $\bigcup_k \mathscr{F}_k$ as done by Chambart (2011). However this definition better captures the intuition one can have of a problem being "complete for F_α."

A point worth making is that the extended Grzegorczyk hierarchy has multiple natural characterizations: 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 in (B.1), as functions provably total in fragments of Peano arithmetic (Fairtlough and Wainer, 1998), etc.—which make the complexity classes we introduced here *meaningful*.

AN $\mathbf{F}_3$-COMPLETE 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:

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

[2]An alternative class for $\alpha \geq 3$ is

$$\mathbf{F}'_\alpha \overset{\text{def}}{=} \bigcup_c \text{DT}\textsc{ime}\left(F_\alpha(n+c)\right) \ ,$$

which is often sufficient and already robust under changes in the model of computation, but not robust under reductions.

Yet another alternative would be to consider the *Wainer hierarchy* $(\mathscr{H}_\beta)_{\beta<\varepsilon_0}$ of functions (Wainer, 1972), which provides an infinite refinement of each $\mathscr{F}_\alpha$ as $\bigcup_{\beta<\omega^{\alpha+1}}\mathscr{H}_\beta$, but its classes lack both forms of robustness: any f in $\mathscr{H}_\beta$ is bounded by H^β the βth function of the *Hardy hierarchy*. What we define here as $\mathbf{F}_\alpha$ seems closer to $\bigcup_{\beta<\omega^\alpha\cdot 2}\mathscr{H}^*_\beta$.

The decision problem we are interested in is whether two such expressions e_1, e_2 are *equivalent*, i.e. whether $[\![e_1]\!] = [\![e_2]\!]$. Stockmeyer and Meyer (1973) show that this problem is hard for $2^{\cdot^{\cdot^{\cdot 2}}}\}_{\log n \text{ times}}$ space under FLogSpace reductions. Then, $\mathbf{F}_3$-hardness follows by an FElementary reduction from any Turing machine working in space $F_3(p(n))$ into a machine working in space $2^{\cdot^{\cdot^{\cdot 2}}}\}_{\log n \text{ times}}$. That the problem is in $\mathbf{F}_3$ 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 $2^{\cdot^{\cdot^{\cdot 2}}}\}_{n \text{ times}}$, thus in $\mathbf{F}_3$.

B.2 $\mathbf{F}_\omega$-Complete Problems

We gather here some decision problems that can be proven decidable in $\mathbf{F}_\omega$ thanks to Dickson's Lemma over $\mathbb{N}^d$ and to the combinatorial analyses of McAloon (1984); Clote (1986); Figueira et al. (2011). We therefore focus on the references for lower bounds.

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

A d-dimensional VAS is a pair $\langle \mathbf{x}_0, \mathbf{A}\rangle$ where $\mathbf{x}_0$ is an initial configuration in $\mathbb{N}^d$ and $\mathbf{A}$ is a finite set of transitions in $\mathbb{Z}^d$. A transition $\mathbf{a}$ in $\mathbf{A}$ can be applied to a configuration $\mathbf{x}$ in $\mathbb{N}^d$ if $\mathbf{x}' = \mathbf{x} + \mathbf{a}$ is in $\mathbb{N}^d$; the resulting configuration is then $\mathbf{x}'$. 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, which is decidable but of unknown complexity: VAS Reachability (Mayr, 1981; Kosaraju, 1982; Lambert, 1992; Leroux, 2011).

[FCP] Finite Containment Problem

instance: Two VAS $\mathcal{V}_1$ and $\mathcal{V}_2$ known to have finite sets $\text{Reach}(\mathcal{V}_1)$ and $\text{Reach}(\mathcal{V}_2)$ of reachable configurations.

question: Is $\text{Reach}(\mathcal{V}_1)$ included in $\text{Reach}(\mathcal{V}_2)$?

reference: Mayr and Meyer (1981), from an F_ω-bounded version of Hilbert's Tenth Problem. A simpler reduction is given by Jančar (2001) from the halting problem of F_ω-bounded Minsky machines.

comment: Testing whether the set of reachable configurations of a VAS is finite is ExpSpace-complete (Lipton, 1976; Rackoff, 1978). [FCP] provided the initial motivation for the work of McAloon (1984); Clote (1986). [FCP] has been generalized by Jančar (2001) to a large range of behaviourial relations between two VASs. Without the finiteness condition, these questions are undecidable (Hack, 1976; Jančar, 1995, 2001).

Lossy Counter Machines. A *lossy counter machine* (LCM) is syntactically a Minsky machine, but its operational semantics are different: its counter values can decrease nondeterministically at any moment during execution. See Chapter 3 for details.

[LCM] Lossy Counter Machines Reachability
instance: A lossy counter machine M and a configuration σ.
question: Is σ reachable in M with lossy semantics?
reference: Schnoebelen (2010a), by a direct reduction from F_ω-bounded Minsky machines. The first proofs were given independently by Urquhart (1999) and Schnoebelen (2002).
comment: Hardness also holds for terminating LCMs, for coverability in Reset or Transfer Petri nets, and for reachability in counter machines with incrementing errors.

[LCMT] Lossy Counter Machines Termination
instance: A lossy counter machine M.
question: Is every run of M finite?
reference: Schnoebelen (2010a), from [LCM].
comment: Hardness also holds for termination of Reset Petri nets.

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

[CRI] Conjunctive Relevant Implication
instance: A formula A of $\mathbf{R}_{\supset,\wedge}$.
question: Is A a theorem of $\mathbf{R}_{\supset,\wedge}$?
reference: 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 the halting problem in F_ω-bounded Minsky machines.
comment: Hardness also holds for $\mathbf{LR}+$ and any intermediate logic between $\mathbf{R}_{\supset,\wedge}$ and $\mathbf{T}_{\supset,\wedge}$—which might include some undecidable fragments.

Data Logics *&* Register Automata are concerned with structures like words or trees with an additional equivalence relation over the elements. 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 $\mathbb{D}$. Ackermannian complexities often arise in this context, both for automata models (essentially register automata and their many variants) and for logics (which include logics with *freeze* operators and XPath fragments)—the two views being tightly interconnected.

[ARA] Emptiness of Alternating 1-Register Automata
instance: An ARA $\mathcal{A}$.
question: Is $L(\mathcal{A})$ empty?
reference: Demri and Lazić (2006), from reachability in incrementing counter machines [LCM].
comment: There exist many variants of the ARA model, and hardness also holds for the corresponding data logics (e.g. Jurdziński and Lazić, 2007; Demri and Lazić, 2009; Figueira and Segoufin, 2009; Tan, 2010; Figueira, 2012). See [ATA] for the case of linearly ordered data.

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:

[ITL] Finite Linear Satisfiability of $A\bar{A}B\bar{B}$ Interval Temporal Logic
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. $\mathcal{S}, I \models \varphi$?
reference: Montanari et al. (2010), from [LCM].
comment: Hardness already holds for the fragments $\bar{A}B$ and $\bar{A}\bar{B}$ (Bresolin et al., 2012).

B.3 $\mathbf{F}_{\omega^\omega}$-Complete Problems

The following problems have been proven decidable thanks to Higman's Lemma over some finite alphabet. All the complexity upper bounds in $\mathbf{F}_{\omega^\omega}$ stem from the constructive proofs of Weiermann (1994); Cichoń and Tahhan Bittar (1998); Schmitz and Schnoebelen (2011). Again, we point to the relevant references for lower bounds.

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'' s.t. $x' \leq_* x''$ and $mx'' \leq_* x$—where $\leq_*$ denotes subword embedding—, while 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 the WSTS theory. LCS are also the primary source of problems hard for $\mathbf{F}_{\omega^\omega}$:

[LCS] LCS Reachability
instance: A LCS and a configuration (q, x) in $Q \times M^*$.
question: Is (q, x) reachable from the initial configuration?
reference: Chambart and Schnoebelen (2008b), by a direct reduction from F_{ω^ω}-bounded Minsky machines.
comment: Hardness already holds for terminating systems, and for reachability in *faulty channel systems*, where symbols are nondeterministically inserted in the channel at arbitrary positions instead of being lost.

[LCST] LCS Termination
instance: A LCS.
question: Is every sequence of transitions from the initial configuration finite?
reference: Chambart and Schnoebelen (2008b), from [LCS].

There are many interesting applications of these questions; 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 behaviour 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].

Embedding Problems have been 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 substring embedding relation $\leq_*$ and called Post Embedding Problems. There is a wealth of variants and applications, see (Chambart and Schnoebelen, 2008a, 2010; Karandikar and Schnoebelen, 2012).

We give here a slightly different viewpoint, taken from (Barceló et al., 2012), that uses regular relations (i.e. definable by synchronous finite transducers) and rational relations (i.e. definable by finite transducers):

[RatEP] Rational Embedding Problem
instance: A rational relation R included in $(\Sigma^*)^2$.
question: Is $R \cap \leq_*$ non empty?
reference: Chambart and Schnoebelen (2007), from [LCS].

comment: Chambart and Schnoebelen (2007) call this problem the Regular Post Embedding Problem, but the name is misleading due to [RegEP]. 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 s.t. $u(w) \leq_* v(w)$.

[RegEP] Regular Embedding Problem
instance: A regular relation R included in $(\Sigma^*)^2$.
question: Is $R \cap \leq_*$ non empty?
reference: Barceló et al. (2012), from [RatEP].

[GEP] Generalized Embedding Problem
instance: A regular relation R included in $(\Sigma^*)^m$ and a subset I of $\{1, ..., m\}^2$.
question: Does there exist $(w_1, \ldots, w_m)$ in R s.t. for all (i, j) in I, $w_i \leq_* w_j$?
reference: Barceló et al. (2012), from [RegEP].
comment: [RegEP] is the case where $m = 2$ and $I = \{(1, 2)\}$. Barceló et al. (2012) use [GEP] to show the $\mathbf{F}_{\omega^\omega}$-completeness of querying graph databases using particular extended conjunctive regular path queries.

Metric Temporal Logic & Timed Automata allow to reason on *timed words* over $\Sigma \times \mathbb{R}$, where Σ is a finite alphabet and the real values are non-decreasing *timestamps* on events. A *timed automaton* (NTA, Alur and Dill, 1994) is a finite automaton 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 (and is then called an ATA).

Metric temporal logic (MTL, Koymans, 1990) 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 $\tau_j - \tau_i \geq 3$. Satisfiability problems for MTL reduce to emptiness problems for timed automata.

Lasota and Walukiewicz (2008) and Ouaknine and Worrell (2007) prove using WSTS techniques that, in the case of a single clock, emptiness of ATAs is decidable.

[ATA] Emptiness of Alternating 1-Clock Timed Automata
instance: An ATA $\mathcal{A}$.
question: Is $L(\mathcal{A})$ empty?
reference: Lasota and Walukiewicz (2008), from faulty channel systems [LCS].
comment: Hardness already holds for universality of nondeterministic 1-clock timed automata.

[fMTL] Finite Satisfiability of Metric Temporal Logic
instance: An MTL formula φ.
question: Does there exist a finite timed word w s.t. $w, 0 \models \varphi$?

reference: Ouaknine and Worrell (2007), from faulty channel systems [LCS].

Note that recent work on data automata over linearly ordered domains has uncovered some strong ties with timed automata (Figueira et al., 2010; Bojańczyk et al., 2011; Figueira, 2012; Bojańczyk and Lasota, 2012).

B.4 $\mathbf{F}_{\omega^{\omega^\omega}}$-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), data nets and Petri data nets (Lazić et al., 2008), and constrained multiset rewriting systems (Abdulla and Delzanno, 2006). Reductions between the different classes of enriched nets can be found in (Abdulla et al., 2011; Bonnet et al., 2010). Defining these families of nets here would take too much space; see the references for details.

[ENC] Enriched Net Coverability
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?
reference: Haddad et al. (2012), by a direct reduction from the halting problem in $F_{\omega^{\omega^\omega}}$-bounded Minsky machines.
comment: Hardness already holds for bounded, terminating nets.

[ENT] Enriched Net Termination
instance: An enriched net $\mathcal{N}$.
question: Are all the executions of the net finite?
reference: Haddad et al. (2012), from [ENC].

REFERENCES

Abdulla, P.A. and Jonsson, B., 1996. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101. doi:10.1006/inco.1996.0053. Cited on page 91.

Abdulla, P.A., Čerāns, K., Jonsson, B., and Tsay, Y.K., 1996. General decidability theorems for infinite-state systems. In *LICS'96*, pages 313–321. IEEE. doi:10.1109/LICS.1996.561359. Cited on page 20.

Abdulla, P.A., Čerāns, K., Jonsson, B., and Tsay, Y.K., 2000. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1–2):109–127. doi:10.1006/inco.1999.2843. Cited on page 20.

Abdulla, P.A., Bouajjani, A., and d'Orso, J., 2008. Monotonic and downward closed games. *Journal of Logic and Computation*, 18(1):153–169. doi:10.1093/logcom/exm062. Cited on page 20.

Abdulla, P.A., Delzanno, G., and Van Begin, L., 2011. A classification of the expressive power of well-structured transition systems. *Information and Computation*, 209(3):248–279. doi:10.1016/j.ic.2010.11.003. Cited on page 93.

Abdulla, P.A. and Nylén, A., 2001. Timed Petri nets and BQOs. In Colom, J.M. and Koutny, M., editors, *Petri Nets 2001*, volume 2075 of *Lecture Notes in Computer Science*, pages 53–70. Springer. doi:10.1007/3-540-45740-2_5. Cited on page 93.

Abdulla, P.A. and Delzanno, G., 2006. On the coverability problem for constrained multiset rewriting. In *AVIS 2006*. Cited on page 93.

Alur, R. and Dill, D.L., 1994. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235. doi:10.1016/0304-3975(94)90010-8. Cited on page 92.

Amadio, R. and Meyssonnier, Ch., 2002. On decidability of the control reachability problem in the asynchronous π-calculus. *Nordic Journal of Computing*, 9(2):70–101. Cited on page 66.

Araki, T. and Kasami, T., 1976. Some decision problems related to the reachability problem for Petri nets. *Theoretical Computer Science*, 3(1):85–104. doi:10.1016/0304-3975(76)90067-0. Cited on page 66.

Atig, M.F., Bouajjani, A., Burckhardt, S., and Musuvathi, M., 2010. On the verification problem for weak memory models. In *POPL 2010*, pages 7–18. ACM Press. doi:10.1145/1706299.1706303. Cited on page 91.

Barceló, P., Figueira, D., and Libkin, L., 2012. Graph logics with rational relations and the generalized intersection problem. In *LICS 2012*, pages 115–124. IEEE. doi:10.1109/LICS.2012.23. Cited on pages 91, 92.

Bellantoni, S. and Cook, S., 1992. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2(2):97–110. doi:10.1007/BF01201998. Cited on page 85.

Bertrand, N. and Schnoebelen, Ph., 2012. Computable fixpoints in well-structured symbolic model checking. *Formal Methods in System Design*. doi:10.1007/s10703-012-0168-y. To appear. Cited on page 20.

Blass, A. and Gurevich, Y., 2008. Program termination and well partial orderings. *ACM Transactions on Computational Logic*, 9(3):1–26. doi:10.1145/1352582.1352586. Cited on page 20.

Blockelet, M. and Schmitz, S., 2011. Model-checking coverability graphs of vector addition systems. In Murlak, F. and Sankowski, P., editors, *MFCS 2011*, volume 6907 of *Lecture Notes in Computer*

Science, pages 108–119. Springer. doi:10.1007/978-3-642-22993-0_13. Cited on pages 21, 88.

Bojańczyk, M., Klin, B., and Lasota, S., 2011. Automata with group actions. In *LICS 2011*, pages 355–364. doi:10.1109/LICS.2011.48. Cited on page 93.

Bojańczyk, M. and Lasota, S., 2012. A machine-independent characterization of timed languages. In Czumaj, A., Mehlhorn, K., Pitts, A., and Wattenhofer, R., editors, *ICALP 2012*, volume 7392 of *Lecture Notes in Computer Science*, pages 92–103. Springer. doi:10.1007/978-3-642-31585-5_12. Cited on page 93.

Bonnet, R., Finkel, A., Haddad, S., and Rosa-Velardo, F., 2010. Comparing Petri Data Nets and Timed Petri Nets. Research Report LSV-10-23, LSV, ENS Cachan. http://tinyurl.com/82vwcxf. Cited on page 93.

Bouyer, P., Markey, N., Ouaknine, J., Schnoebelen, Ph., and Worrell, J., 2012. On termination and invariance for faulty channel machines. *Formal Aspects of Computing*, 24(4):595–607. doi:10.1007/s00165-012-0234-7. Cited on page 66.

Bresolin, D., Della Monica, D., Montanari, A., Sala, P., and Sciavicco, G., 2012. Interval temporal logics over finite linear orders: The complete picture. In *ECAI 2012*. To appear. Cited on pages 66, 90.

Cardoza, E., Lipton, R., and Meyer, A.R., 1976. Exponential space complete problems for Petri nets and commutative subgroups. In *STOC'76*, pages 50–54. ACM Press. doi:10.1145/800113.803630. Cited on page 21.

Cécé, G., Finkel, A., and Purushothaman Iyer, S., 1996. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31. doi:10.1006/inco.1996.0003. Cited on page 91.

Chambart, P. and Schnoebelen, Ph., 2010. Computing blocker sets for the Regular Post Embedding Problem. In *DLT 2010*, volume 6224 of *Lecture Notes in Computer Science*, pages 136–147. Springer. doi:10.1007/978-3-642-14455-4_14. Cited on page 91.

Chambart, P. and Schnoebelen, Ph., 2007. Post embedding problem is not primitive recursive, with applications to channel systems. In Arvind, V. and Prasad, S., editors, *FSTTCS 2007*, volume 4855 of *Lecture Notes in Computer Science*, pages 265–276. Springer. doi:10.1007/978-3-540-77050-3_22. Cited on pages 91, 92.

Chambart, P. and Schnoebelen, Ph., 2008a. The ω-regular Post embedding problem. In Amadio, R., editor, *FoSSaCS 2008*, volume 4962 of *Lecture Notes in Computer Science*, pages 97–111. Springer. doi:10.1007/978-3-540-78499-9_8. Cited on page 91.

Chambart, P. and Schnoebelen, Ph., 2008b. The ordinal recursive complexity of lossy channel systems. In *LICS 2008*, pages 205–216. IEEE. doi:10.1109/LICS.2008.47. Cited on pages 66, 91.

Chambart, P., 2011. *On Post's Embedding Problem and the complexity of lossy channels.* PhD thesis, ENS Cachan. http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/chambart-these11.pdf. Cited on page 87.

Chlebus, B.S., 1986. Domino-tiling games. *Journal of Computer and System Sciences*, 32(3):374–392. doi:10.1016/0022-0000(86)90036-X. Cited on page 84.

Ciardo, G., 1994. Petri nets with marking-dependent arc cardinality: Properties and analysis. In Valette, R., editor, *Petri nets '94*, volume 815 of *Lecture Notes in Computer Science*, pages 179–198. Springer. doi:10.1007/3-540-58152-9_11. Cited on page 66.

Cichoń, E.A. and Wainer, S.S., 1983. The slow-growing and the Grzecorczyk hierarchies. *Journal of Symbolic Logic*, 48(2):399–408. Cited on pages 67, 72, 77.

Cichoń, E.A. and Tahhan Bittar, E., 1998. Ordinal recursive bounds for Higman's Theorem. *Theoretical Computer Science*, 201(1–2):63–84. doi:10.1016/S0304-3975(97)00009-1. Cited on pages 50, 67, 69, 72, 73, 75, 90.

Clote, P., 1999. Computation models and function algebras. In Griffor, E.R., editor, *Handbook of Computability Theory*, volume 140 of *Studies in Logic and the Foundations of Mathematics*, chapter 17, pages 589–681. Elsevier. doi:10.1016/S0049-237X(99)80033-0. Cited on page 85.

Clote, P., 1986. On the finite containment problem for Petri nets. *Theoretical Computer Science*, 43:

99–105. doi:10.1016/0304-3975(86)90169-6. Cited on pages 49, 88.

Cobham, A., 1965. The intrinsic computational difficulty of functions. In Bar-Hillel, Y., editor, *International Congress for Logic, Methodology and Philosophy of Science*, volume 2, pages 24–30. North-Holland. Cited on page 85.

Cook, B., Podelski, A., and Rybalchenko, A., 2011. Proving program termination. *Communications of the ACM*, 54:88–98. doi:10.1145/1941487.1941509. Cited on page 20.

de Jongh, D.H.J. and Parikh, R., 1977. Well-partial orderings and hierarchies. *Indagationes Mathematicae*, 39(3):195–207. doi:10.1016/1385-7258(77)90067-1. Cited on page 50.

Demri, S., 2006. Linear-time temporal logics with Presburger constraints: An overview. *Journal of Applied Non-Classical Logics*, 16(3–4):311–347. doi:10.3166/jancl.16.311-347. Cited on page 66.

Demri, S. and Lazić, R., 2006. LTL with the freeze quantifier and register automata. In *LICS 2006*, pages 17–26. IEEE. doi:10.1109/LICS.2006.31. Cited on page 90.

Demri, S. and Lazić, R., 2009. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic*, 10(3). doi:10.1145/1507244.1507246. Cited on pages 66, 90.

Dennis-Jones, E. and Wainer, S., 1984. Subrecursive hierarchies via direct limits. In Börger, E., Oberschelp, W., Richter, M., Schinzel, B., and Thomas, W., editors, *Computation and Proof Theory*, volume 1104 of *Lecture Notes in Mathematics*, pages 117–128. Springer. doi:10.1007/BFb0099482. Cited on pages 68, 69.

Dickson, L.E., 1913. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422. doi:10.2307/2370405. Cited on page 20.

Dufourd, C., Jančar, P., and Schnoebelen, Ph., 1999. Boundedness of reset P/T nets. In *ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 301–310. Springer. doi:10.1007/3-540-48523-6_27. Cited on page 66.

Dunn, J.M. and Restall, G., 2002. Relevance logic. In Gabbay, D.M. and Guenthner, F., editors, *Handbook of Philosophical Logic*, volume 6, pages 1–128. Kluwer Academic Publishers. http://consequently.org/papers/rle.pdf. Cited on pages 20, 89.

Fairtlough, M.V.H. and Wainer, S.S., 1992. Ordinal complexity of recursive definitions. *Information and Computation*, 99(2):123–153. doi:10.1016/0890-5401(92)90027-D. Cited on pages 68, 69.

Fairtlough, M. and Wainer, S.S., 1998. Hierarchies of provably recursive functions. In Buss, S., editor, *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, chapter III, pages 149–207. Elsevier. doi:10.1016/S0049-237X(98)80018-9. Cited on pages 50, 67, 87.

Figueira, D. and Segoufin, L., 2009. Future-looking logics on data words and trees. In Královič, R. and Niwiński, D., editors, *MFCS 2009*, volume 5734 of *Lecture Notes in Computer Science*, pages 331–343. Springer. doi:10.1007/978-3-642-03816-7_29. Cited on pages 66, 90.

Figueira, D., Hofman, P., and Lasota, S., 2010. Relating timed and register automata. In Fröschle, S. and Valencia, F., editors, *EXPRESS 2010*, volume 41 of *EPTCS*, pages 61–75. doi:10.4204/EPTCS.41.5. Cited on page 93.

Figueira, D., Figueira, S., Schmitz, S., and Schnoebelen, Ph., 2011. Ackermannian and primitive-recursive bounds with Dickson's Lemma. In *LICS 2011*, pages 269–278. IEEE. doi:10.1109/LICS.2011.39. Cited on pages iii, 49, 88.

Figueira, D., 2012. Alternating register automata on finite words and trees. *Logical Methods in Computer Science*, 8(1):22. doi:10.2168/LMCS-8(1:22)2012. Cited on pages 90, 93.

Finkel, A., 1987. A generalization of the procedure of Karp and Miller to well structured transition systems. In *ICALP'87*, volume 267 of *Lecture Notes in Computer Science*, pages 499–508. Springer. doi:10.1007/3-540-18088-5_43. Cited on page 20.

Finkel, A., 1990. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–179. doi:10.1016/0890-5401(90)90009-7. Cited on page 20.

Finkel, A. and Schnoebelen, Ph., 2001. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92. doi:10.1016/S0304-3975(00)00102-X. Cited on page 20.

Finkel, A. and Goubault-Larrecq, J., 2009. Forward analysis for WSTS, part I: Completions. In *STACS 2009*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 433–444. LZI. doi:10.4230/LIPIcs.STACS.2009.1844. Cited on page 20.

Finkel, A. and Goubault-Larrecq, J., 2012. Forward analysis for WSTS, part II: Complete WSTS. *Logical Methods in Computer Science*. To appear. Cited on pages 20, 21.

Friedman, H.M., 1999. Some decision problems of enormous complexity. In *LICS 1999*, pages 2–13. IEEE. doi:10.1109/LICS.1999.782577. Cited on page 83.

Friedman, H.M., 2001. Long finite sequences. *Journal of Combinatorial Theory, Series A*, 95(1): 102–144. doi:10.1006/jcta.2000.3154. Cited on page 49.

Grzegorczyk, A., 1953. Some classes of recursive functions. *Rozprawy Matematyczne*, 4. http://matwbn.icm.edu.pl/ksiazki/rm/rm04/rm0401.pdf. Cited on page 50.

Hack, M., 1976. The equality problem for vector addition systems is undecidable. *Theoretical Computer Science*, 2(1):77–95. doi:10.1016/0304-3975(76)90008-6. Cited on page 88.

Haddad, S., Schmitz, S., and Schnoebelen, Ph., 2012. The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In *LICS 2012*, pages 355–364. IEEE. doi:10.1109/LICS.2012.46. Cited on pages iii, 66, 93.

Halpern, J.Y. and Shoham, Y., 1991. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962. doi:10.1145/115234.115351. Cited on page 90.

Higman, G., 1952. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(2):326–336. doi:10.1112/plms/s3-2.1.326. Cited on page 20.

Howell, R.R., Rosier, L.E., Huynh, D.T., and Yen, H.C., 1986. Some complexity bounds for problems concerning finite and 2-dimensional vector addition systems with states. *Theoretical Computer Science*, 46:107–140. doi:10.1016/0304-3975(86)90026-5. Cited on page 49.

Jančar, P., 1999. A note on well quasi-orderings for powersets. *Information Processing Letters*, 72 (5–6):155–161. doi:10.1016/S0020-0190(99)00149-0. Cited on pages 16, 20.

Jančar, P., 1995. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301. doi:10.1016/0304-3975(95)00037-W. Cited on page 88.

Jančar, P., 2001. Nonprimitive recursive complexity and undecidability for Petri net equivalences. *Theoretical Computer Science*, 256(1–2):23–30. doi:10.1016/S0304-3975(00)00100-6. Cited on pages 21, 88.

Jurdziński, M. and Lazić, R., 2007. Alternation-free modal mu-calculus for data trees. In *LICS 2007*, pages 131–140. IEEE. doi:10.1109/LICS.2007.11. Cited on pages 66, 90.

Karandikar, P. and Schnoebelen, Ph., 2012. Cutting through regular Post embedding problems. In *CSR 2012*, volume 7353 of *Lecture Notes in Computer Science*, pages 229–240. Springer. doi:10.1007/978-3-642-30642-6_22. Cited on page 91.

Karp, R.M. and Miller, R.E., 1969. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195. doi:10.1016/S0022-0000(69)80011-5. Cited on page 21.

Ketonen, J. and Solovay, R., 1981. Rapidly growing Ramsey functions. *Annals of Mathematics*, 113 (2):27–314. doi:10.2307/2006985. Cited on page 49.

Kosaraju, S.R., 1982. Decidability of reachability in vector addition systems. In *STOC'82*, pages 267–281. ACM Press. doi:10.1145/800070.802201. Cited on page 88.

Koymans, R., 1990. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299. doi:10.1007/BF01995674. Cited on page 92.

Kripke, S.A., 1959. The problem of entailment. In *ASL 1959*, volume 24(4) of *Journal of Symbolic Logic*, page 324. http://www.jstor.org/stable/2963903. Abstract. Cited on page 20.

Kruskal, J.B., 1972. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13(3):297–305. doi:10.1016/0097-3165(72)90063-5. Cited on pages iii, 20.

Lambert, J.L., 1992. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99(1):79–104. doi:10.1016/0304-3975(92)90173-D. Cited on page 88.

Lasota, S. and Walukiewicz, I., 2008. Alternating timed automata. *ACM Transactions on Computational Logic*, 9(2):10. doi:10.1145/1342991.1342994. Cited on page 92.

Lazić, R., Newcomb, T., Ouaknine, J., Roscoe, A., and Worrell, J., 2008. Nets with tokens which carry data. *Fundamenta Informaticae*, 88(3):251–274. Cited on page 93.

Leroux, J., 2011. Vector addition system reachability problem: a short self-contained proof. In *POPL 2011*, pages 307–316. ACM Press. doi:10.1145/1926385.1926421. Cited on page 88.

Lipton, R.J., 1976. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University. Cited on page 88.

Löb, M. and Wainer, S., 1970. Hierarchies of number theoretic functions, I. *Archiv für Mathematische Logik und Grundlagenforschung*, 13:39–51. doi:10.1007/BF01967649. Cited on pages 50, 75, 79, 80, 84.

Lovász, L., 2006. Graph minor theory. *Bulletin of the American Mathematical Society*, 43(1):75–86. doi:10.1090/S0273-0979-05-01088-8. Cited on page 20.

Marcone, A., 1994. Foundations of BQO theory. *Transactions of the American Mathematical Society*, 345(2):641–660. doi:10.1090/S0002-9947-1994-1219735-8. Cited on page 20.

Mayr, E.W., 1981. An algorithm for the general Petri net reachability problem. In *STOC'81*, pages 238–246. ACM Press. doi:10.1145/800076.802477. Cited on page 88.

Mayr, E.W. and Meyer, A.R., 1981. The complexity of the finite containment problem for Petri nets. *Journal of the ACM*, 28(3):561–576. doi:10.1145/322261.322271. Cited on pages 21, 88.

Mayr, R., 2000. Undecidable problems in unreliable computations. In *LATIN 2000*, volume 1776 of *Lecture Notes in Computer Science*, pages 377–386. Springer. doi:10.1007/10719839_37. Cited on page 66.

McAloon, K., 1984. Petri nets and large finite sets. *Theoretical Computer Science*, 32(1–2):173–183. doi:10.1016/0304-3975(84)90029-X. Cited on pages 49, 88.

Meyer, A.R. and Ritchie, D.M., 1967. The complexity of loop programs. In *ACM '67*, pages 465–469. doi:10.1145/800196.806014. Cited on page 87.

Milner, E.C., 1985. Basic WQO- and BQO-theory. In Rival, I., editor, *Graphs and Order. The Role of Graphs in the Theory of Ordered Sets and Its Applications*, pages 487–502. D. Reidel Publishing. Cited on page 20.

Montanari, A., Puppis, G., and Sala, P., 2010. Maximal decidable fragments of Halpern and Shoham's modal logic of intervals. In Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., and Spirakis, P., editors, *ICALP 2010*, volume 6199 of *Lecture Notes in Computer Science*, pages 345–356. Springer. doi:10.1007/978-3-642-14162-1_29. Cited on page 90.

Odifreddi, P.G., 1999. *Classical Recursion Theory, vol. II*, volume 143 of *Studies in Logic and the Foundations of Mathematics*. Elsevier. doi:10.1016/S0049-237X(99)80040-8. Cited on pages 50, 67.

Ouaknine, J.O. and Worrell, J.B., 2007. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, 3(1):8. doi:10.2168/LMCS-3(1:8)2007. Cited on pages 92, 93.

Padovani, V., 2012. Ticket Entailment is decidable. *Mathematical Structures in Computer Science*. arXiv:1106.1875. To appear. Cited on page 20.

Podelski, A. and Rybalchenko, A., 2004. Transition invariants. In *LICS 2004*, pages 32–41. IEEE. doi:10.1109/LICS.2004.1319598. Cited on page 20.

Rackoff, C., 1978. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231. doi:10.1016/0304-3975(78)90036-1. Cited on pages 21, 88.

Rado, R., 1954. Partial well-ordering of sets of vectors. *Mathematika*, 1(2):89–95. doi:10.1112/S0025579300000565. Cited on pages 16, 20.

Ritchie, R.W., 1963. Classes of predictably computable functions. *Transactions of the American Mathematical Society*, 106(1):139–173. doi:10.1090/S0002-9947-1963-0158822-2. Cited on page 85.

Rose, H.E., 1984. *Subrecursion: Functions and Hierarchies*, volume 9 of *Oxford Logic Guides*. Clarendon Press. Cited on pages 50, 67.

Schmitz, S. and Schnoebelen, Ph., 2011. Multiply-recursive upper bounds with Higman's Lemma. In *ICALP 2011*, volume 6756 of *Lecture Notes in Computer Science*, pages 441–452. Springer. doi:10.1007/978-3-642-22012-8_35. Cited on pages iii, 49, 90.

Schnoebelen, Ph., 2002. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261. doi:10.1016/S0020-0190(01)00337-4. Cited on pages 66, 89.

Schnoebelen, Ph., 2010a. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In Hliněný, P. and Kučera, A., editors, *MFCS 2010*, volume 6281 of *Lecture Notes in Computer Science*, pages 616–628. Springer. doi:10.1007/978-3-642-15155-2_54. Cited on pages iii, 66, 89.

Schnoebelen, Ph., 2010b. Lossy counter machines decidability cheat sheet. In Kučera, A. and Potapov, I., editors, *RP 2010*, volume 6227 of *Lecture Notes in Computer Science*, pages 51–75. Springer. doi:10.1007/978-3-642-15349-5_4. Cited on page 66.

Seidl, H., 1990. Deciding equivalence of finite tree automata. *SIAM Journal on Computing*, 19(3): 424–437. doi:10.1137/0219027. Cited on page 84.

Stockmeyer, L.J. and Meyer, A.R., 1973. Word problems requiring exponential time. In *STOC '73*, pages 1–9. ACM Press. doi:10.1145/800125.804029. Cited on pages 87, 88.

Tan, T., 2010. On pebble automata for data languages with decidable emptiness problem. *Journal of Computer and System Sciences*, 76(8):778–791. doi:10.1016/j.jcss.2010.03.004. Cited on pages 66, 90.

Turing, A., 1949. Checking a large routine. In *Report of a Conference on High Speed Automatic Calculating Machines*. Republished in *The early British computer conferences*, pages 70–72, MIT Press, 1989. Cited on page 20.

Urquhart, A., 1984. The undecidability of entailment and relevant implication. *Journal of Symbolic Logic*, 49(4):1059–1073. http://www.jstor.org/stable/2274261. Cited on pages 20, 89.

Urquhart, A., 1999. The complexity of decision procedures in relevance logic II. *Journal of Symbolic Logic*, 64(4):1774–1802. doi:10.2307/2586811. Cited on pages 20, 66, 89.

Wainer, S.S., 1970. A classification of the ordinal recursive functions. *Archiv für Mathematische Logik und Grundlagenforschung*, 13(3):136–153. doi:10.1007/BF01973619. Cited on pages 85, 87.

Wainer, S.S., 1972. Ordinal recursion, and a refinement of the extended Grzegorczyk hierarchy. *Journal of Symbolic Logic*, 37(2):281–292. http://www.jstor.org/stable/2272973. Cited on page 87.

Weiermann, A., 1994. Complexity bounds for some finite forms of Kruskal's Theorem. *Journal of Symbolic Computation*, 18(5):463–488. doi:10.1006/jsco.1994.1059. Cited on pages 50, 90.

INDEX