Algorithmic Aspects of WQO Theory

S. Schmitz and Ph. Schnoebelen
LSV, ENS Cachan & CNRS, Université Paris-Saclay, 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, wqos 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 two advanced courses taught at the *24th European Summer School in Logic, Language and Information* (ESSLLI 2012) on August 6–10, 2012 in Opole, Poland, and at the *28th European Summer School in Logic* (ESSLLI 2016) on August 22–26, 2016 in Bolzano-Bozen, Italy; we also employ these notes as 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).

These notes 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, partly based on (Finkel and Schnoebelen, 2001),
- Chapter 2 proves complexity upper bounds for the use of Dickson's Lemma—this chapter is adapted chiefly from (Schmitz and Schnoebelen, 2011)—,

- Chapter 3 details how to derive Ackermannian lower bounds on decision problems, drawing heavily on (Schnoebelen, 2010a), and
- Chapter 4 investigates ideals of wqos and their applications, chiefly based on (Goubault-Larrecq et al., 2016; Blondin et al., 2014; Lazić and Schmitz, 2015a).

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 presents a few problems of enormous complexity. These are based on (Schmitz, 2016b).

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 $\equiv \stackrel{\text{def}}{=} \leq \cap \geq$, and gives rise to a canonical partial ordering between the equivalence classes, and to a *strict ordering* $< \stackrel{\text{def}}{=} \leq \smallsetminus \geq = \leq \smallsetminus \equiv$ 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 non negative 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 more 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 characterisations, 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
finite antichain condition

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 characterisations is quite useful; see Exercise 1.3 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.

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.[1] 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 colour every edge $\{i, j\}$ (where $i < j$) with one of three colours. 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 colour. 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 colour for the edges of I must be red. Hence the original sequence has a infinite increasing subsequence: $(A, \leq)$ satisfies (wqo.2).

Ramsey Theorem

1.1.2 Upward-closed Subsets of wqos

upward-closure

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.

upward-closed
downward-closure
downward-closed

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 stabilise, i.e., $\bigcup_{i\in\mathbb{N}} U_i$ is $U_k = U_{k+1} = U_{k+2} = \ldots$ for some k.

Equivalently, a qo $(A, \leq)$ is a wqo iff $(\mathit{Down}(A), \subseteq)$, the set of its downward-closed subsets ordered by inclusion, is well-founded.

This characterisation is sometimes expressed by saying that upward-closed sets

[1] See Exercise 1.5 for an elementary proof that does not use Ramsey's Theorem.

satisfy the Ascending Chain Condition. See Exercise 1.7 for the equivalence of (wqo.4) with the other characterisations. ascending chain condition

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 (Finite Basis Property)**.** *Let $(A, \leq)$ be a wqo. Any upward-closed $U \subseteq A$ can be written under the form $U = \uparrow\{x_1, \ldots, x_n\}$ for some $x_1, \ldots, x_n \in A$, i.e., as the upward-closure of finitely many elements.*

(See Exercise 1.8 for a proof.) One can see how, using this representation, the comparison of possibly infinite (but upward-closed) sets can be reduced to finitely many comparisons of elements.

The complement of a downward-closed set D is upward-closed. Hence downward-closed subsets of a wqo can be characterised 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". In Chapter 4 we consider other representations for downward-closed sets. 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 1.3).

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

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

See Exercise 1.12 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

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

Example 1.10 (Subword ordering)**.** We use ε to denote the empty sequence. Over $(\Sigma, =)$, where $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ is a 3-letter alphabet and where different letters are incomparable, the word $\mathtt{abb}$ is a subword of $\mathtt{c\underline{ab}ca\underline{b}}$, as witnessed by the underlined letters, and written $\mathtt{abb} \leq_* \mathtt{cabcab}$. On the other hand $\mathtt{bba} \not\leq_* \mathtt{cabcab}$. 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 $4{\cdot}1{\cdot}3 \not\leq_* 1{\cdot}5{\cdot}0{\cdot}3{\cdot}0{\cdot}0$. 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 in several different ways, see Exercise 1.15.

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

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

(See Exercise 1.16 for the definition of homeomorphic embeddings and a proof of Kruskal's Theorem.)

Finally, a further generalisation of Kruskal's Tree Theorem exists for graphs (Robertson and Seymour, 2010):

Graph Minor Theorem

Theorem 1.12 (Robertson and Seymour's Graph-Minor Theorem)**.** *The set of (finite undirected) graphs node-labelled 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 1.17 to 1.20.

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 (existential) *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. image-finite
It is *Post-effective* if these elements can effectively be computed from s. *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 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 $s = s_0 \to s_1 \to \cdots \to s_n \geq t$?

control-state reachability

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 $x \leq_A x'$.

backward coverability

predecessor set

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 $U \subseteq S$, define its (existential) *predecessor set*

$$Pre_\exists(U) \stackrel{\text{def}}{=} \{s \in S \mid \exists s' \in U,\ s \to s'\} \ . \tag{1.3}$$

The backward analysis computes the limit $Pre^*_\exists(U)$ of the sequence

$$U = U_0 \subseteq U_1 \subseteq \cdots \text{ where } U_{n+1} \stackrel{\text{def}}{=} U_n \cup Pre_\exists(U_n) \ . \tag{1.4}$$

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

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

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

effective pred-basis

A corollary is that sequence (1.4) stabilises to $Pre^*_\exists(U)$ 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 $\uparrow Pre_\exists(\uparrow\{s\})$.[2]

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

Exercises 1.18 and 1.20 present variants of this algorithm for different notions of compatibility.

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

```
SIMPLE (a, b)
c ⟵ 1
while a > 0 ∧ b > 0
    l : ⟨a, b, c⟩ ⟵ ⟨a − 1, b, 2c⟩
  or
    r : ⟨a, b, c⟩ ⟵ ⟨2c, b − 1, 1⟩
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 specialised: 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 characterisations 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.

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:

well-founded relation
Noetherian relation

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

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

ranking function

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

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.

Transition Invariants. A generalisation 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

disjunctive termination argument

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

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 1.23 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 over-approximation $\mathrm{wf}(R_j) \supseteq R_j$, 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 over-approximated 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 over-approximation $\mathrm{wf}(\mathrm{wf}(R_{j_1}) \fatsemi R_{j_2})$ of the composition of R_{j_1} with R_{j_2}, then $\mathrm{wf}(\mathrm{wf}(\mathrm{wf}(R_{j_1}) \fatsemi R_{j_2}) \fatsemi R_{j_3})$ 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:

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

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

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

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

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

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,

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \gamma A \vdash B\end{array}}{\gamma \vdash A \supset B}\ (\supset_\mathsf{R}) \qquad \dfrac{\begin{array}{c}\vdots\\ \alpha \vdash A\end{array} \qquad \dfrac{\begin{array}{c}\vdots\\ \beta B \vdash C\end{array}}{\alpha\beta(A \supset B) \vdash C}\ (\supset_\mathsf{L})}{\alpha\beta\gamma \vdash C}\ (\mathsf{Cut})$$

can be rewritten into

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \gamma A \vdash B\end{array} \qquad \begin{array}{c}\vdots\\ \alpha \vdash A\end{array}}{\alpha\gamma \vdash B}\ (\mathsf{Cut}) \qquad \begin{array}{c}\vdots\\ \beta B \vdash C\end{array}}{\alpha\beta\gamma \vdash C}\ (\mathsf{Cut})$$

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 generalised to whether a sequent $\alpha \vdash A$ is derivable using (Ax–$\supset_{\mathsf{R}}$).

[RI] Relevant Implication
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 labelled 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 1.26 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 $\mathcal{S} = \langle S, \to, \leq \rangle$, 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 $\mathrm{Sub}(A)$ 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 $\mathrm{Seq}(A) \stackrel{\text{def}}{=} \mathbb{N}^{\mathrm{Sub}(A)} \times \mathrm{Sub}(A)$; we let

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

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

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

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

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 $\mathrm{Seq}(A)$, this is equivalent to having $\alpha \leq \alpha'$ (for the product ordering over $\mathbb{N}^{\mathrm{Sub}(A)}$), $\sigma(\alpha) = \sigma(\alpha')$, and $B = B'$: $\ll$ over $\mathrm{Seq}(A)$ is thus defined as a product ordering between the three wqos $(\mathbb{N}^{\mathrm{Sub}(A)}, \leq)$, $(\mathcal{P}(\mathrm{Sub}(A)), =)$, and $(\mathrm{Sub}(A), =)$, and therefore by Dickson's Lemma:

multiset support

contraction ordering

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

Then, by Exercise 1.15, the qo $(S, \leq)$, where $\leq$ is *Hoare's ordering* applied to $\ll$, is a wqo, and we easily see that $\mathcal{S} = \langle S, \to, \leq \rangle$ is a WSTS with effective pred-basis and a decidable ordering (see Exercise 1.25), thus the coverability problem for

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

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–$\supset_{\mathsf{R}}$) 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

Vector Addition Systems (VAS) are systems where d counters evolve by nondeterministically 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).

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

covering

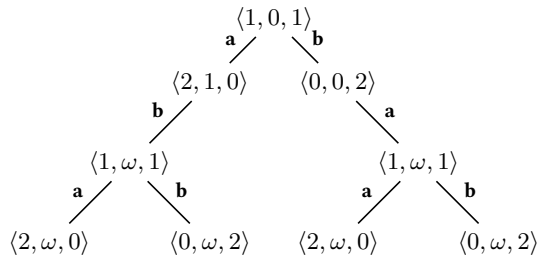

Figure 1.2: A Karp & Miller tree constructed for the VAS $\langle\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}, \langle 1, 0, 1\rangle\rangle$ with translations $\mathbf{a} = \langle 1, 1, -1\rangle$, $\mathbf{b} = \langle -1, 0, 1\rangle$, and $\mathbf{c} = \langle 0, -1, 0\rangle$.

---

1. for any $\mathbf{x} \in \mathrm{Reach}(\mathcal{V})$, 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 $\mathrm{Reach}(\mathcal{V})$.

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.

Karp & Miller tree

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

Assume a node s of the tree is labelled 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 $\mathbf{a}$-labelled 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 $j = 1, \ldots, d$,
$$\mathbf{y}''(j) \stackrel{\text{def}}{=} \begin{cases} \omega & \text{if } \mathbf{y}'(j) > \mathbf{y}_i(j) \\ \mathbf{y}'(j) & \text{otherwise.} \end{cases} \tag{1.22}$$
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 labelled 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 1.28.

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;

(3) given a linearly ordered finite alphabet Σ —e.g., $\mathtt{a} < \mathtt{b} < \cdots < \mathtt{z}$— the set of finite sequences Σ^* with *prefix ordering* $\leq_{\text{pref}}$ or *lexicographic ordering* $\leq_{\text{lex}}$;

prefix ordering

lexicographic ordering

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

subsets

(5) $(\mathcal{P}(\mathbb{N}), \sqsubseteq_{\text{S}})$, where $\sqsubseteq_{\text{S}}$, called *Smyth's ordering*, is given by $U \sqsubseteq_{\text{S}} V \stackrel{\text{def}}{\Leftrightarrow} \forall m \in V, \exists n \in U, n \leq m$;

Smyth's ordering

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

Exercise 1.2. Let $(A, \leq_1)$ and $(A, \leq_2)$ be two wqos over the same support set. Show that $(A, \leq_1 \cap \leq_2)$ is a wqo.

Exercise 1.3 (Generalised 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

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

(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.

Exercise 1.4 (Lexicographic Sum and Product)**.** Let $(A_1, \leq_1)$ and $(A_2, \leq_2)$ be two qos. The *lexicographic sum* $A_1 + A_2$ is the qo $(A_1 + A_2, \leq_{\leq_{\text{lex}}})$ with same support set $\{1\} \times A_1 \cup \{2\} \times A_2$ as for the disjoint sum but with an ordering defined with

ohic sum

$$\langle i, x\rangle \leq_{+_{\leq_{\text{lex}}}} \langle j, y\rangle \overset{\text{def}}{\Leftrightarrow} i < j \ \vee \ i = j \wedge x \leq_i y\,.$$

Similarly, the *lexicographic product* $A_1 \times_{\leq_{\text{lex}}} A_2$ has the same support set $A_1 \times A_2$ as for the Cartesian product but ordered with

lexicographic product

$$\langle x, y\rangle \leq_{\times_{\leq_{\text{lex}}}} \langle x', y'\rangle \overset{\text{def}}{\Leftrightarrow} x <_1 x' \ \vee \ x \equiv_1 x' \wedge y \leq_2 y'$$

(1) Show that $A_1 +_{\leq_{\text{lex}}} A_2$ is a linear ordering iff A_1 and A_2 are.

(2) Show that $A_1 \times_{\leq_{\text{lex}}} A_2$ is a linear ordering when A_1 and A_2 are but that the reciprocal does not hold.

(3) Show that $A_1 +_{\leq_{\text{lex}}} A_2$ and $A_1 \times_{\leq_{\text{lex}}} A_2$ are wqos when A_1 and A_2 are.

Exercise 1.5 (Equivalence of (wqo.1), (wqo.2), and (wqo.3))**.** Assume that $(A, \leq)$ is a wqo in the sense of Definition 1.1. We want to show that it satisfies Definition 1.3 without invoking Ramsey's Theorem as was done in Section 1.1.1. For this we follow Erdös et al. (1950):

(1) Consider an infinite sequence $x_0, x_1, x_2, \ldots$ over A and write M for the set $\{i \in \mathbb{N} \mid x_i \not\leq x_j \text{ for all } j > i\}$. Show that M is finite.

(2) Conclude and show that (wqo.1) implies (wqo.2).

(3) Prove, using similar ideas, that (wqo.3) implies (wqo.1).

Exercise 1.6 (How Many Antichains?)**.** Assume that $(A, \leq)$ is countable and well-founded. Show that $(A, \leq)$ is wqo iff the set of its antichains is countable.

Exercise 1.7 (Ascending Chain Condition)**.** Show that (wqo.4) is equivalent with the other definition(s) of wqos.

Exercise 1.8 (Finite Basis Property)**.**

(1) For a qo $(A, \leq)$ and a subset $U \subseteq A$, we say that x is a "*minimal element of* U" if $x \in U$ and there is no $y \in U$ with $y < x$. Show that every element of a well-founded qo is larger than or equal to a minimal element of the qo.

(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.

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

Exercise 1.9 (Linear WQOs)**.**

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

linearisation

order-extension principle

(2) (wqo.6) Prove that a qo is a wqo iff all its linearisations are well-founded (a result from (Wolk, 1967)), where a *linearisation* of $(A, \leq)$ is any linear qo $(A, \preccurlyeq)$ with same support set and such that $x \leq y$ implies $x \preccurlyeq y$ (and such that $x < y$ implies $x \prec y$). Here one may assume the Order-extension principle: "every qo has a linearisation".

sparser-than ordering

Exercise 1.10 $(\mathbb{Z}^k, \leq_{\text{sparse}})$**.** We consider the *sparser-than ordering*. Given $\mathbf{a} = (a_1, \ldots, a_k)$ and $\mathbf{b} = (b_1, \ldots, b_k)$ two tuples in $\mathbb{Z}^k$, we define

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

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

Exercise 1.11 (Rado's Structure)**.** We consider the following set $R = \{(a, b) \in \mathbb{N}^2 \mid a < b\}$ ordered with

$$(a, b) \leq_R (a', b') \stackrel{\text{def}}{\Leftrightarrow} (a = a' \wedge b \leq b') \vee b < a' .$$

This structure $(R, \leq_R)$ appeared in (Rado, 1954) and is called *Rado's structure*, see exer-

Rado's structure

cises 1.13 and 1.15 for applications.

Show that $(R, \leq_R)$ is a wqo.

Exercise 1.12 (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 1.13 (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

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

(1) We start with the ω-sequence extension of $(\mathbb{N}, \leq)$ and consider ω-words $v, w \in \mathbb{N}^\omega$ of natural numbers. We say that an ω-word $v \in \mathbb{N}^\omega$ is *unbounded* if it contains arbitrarily large natural numbers. What can you say about unbounded ω-words and $\leq_\omega$?

(2) With a bounded ω-word $v \in \mathbb{N}^\omega$, of the form $v = x_0, x_1, x_2, \ldots$, we associate $L(v)$, defined as $L(v) \stackrel{\text{def}}{=} \limsup_i x_i = \lim_{k\to\infty} \max_{i\geq k} x_i$ (note that $L(v)$ is a finite number since v is bounded), we let $M(v)$ be the first index such that $x_i \leq L(v)$ for all $i \geq M(v)$. The finite sequence $\dot{v} \stackrel{\text{def}}{=} x_0, \ldots, x_{M(v)-1}$ is the shortest prefix of v such that v can be written $v = \dot{v}.\ddot{v}$ with $\ddot{v}$ an ω-length suffix having all its elements bounded by $L(v)$.

Assume that $w = y_0, y_1, y_2, \ldots$ is a second bounded ω-word and show that

$$L(v) \leq L(w) \text{ implies } \ddot{v} \leq_\omega \ddot{w}\,, \tag{E}$$

$$\big(L(v) \leq L(w) \ \wedge\ \dot{v} \leq_* \dot{w}\big) \text{ implies } v \leq_\omega w\,. \tag{E$'$}$$

(3) Eq. (E$'$) gives a sufficient condition for $v \leq_\omega w$. Is it a necessary condition?

(4) Show that $(\mathbb{N}^\omega, \leq_\omega)$ is a wqo.

(5) Generalise the previous question and show that $(A^\omega, \leq_\omega)$ is a wqo when $(A, \leq)$ is a *linear* wqo.

(6) We consider a finite alphabet $(\Sigma, =)$ equipped with the empty ordering. Show that its ω-sequence extension $(\Sigma^\omega, \leq_\omega)$ is a wqo.

(7) Show that $(R^\omega, \leq_{R,\omega})$, the ω-sequence extension of Rado's structure $(R, \leq_R)$ —see Exercise 1.11—, is *not* a wqo.

(8) We return to the general case where $(A, \leq)$ is a wqo. Show that $(A^\omega, \leq_\omega)$ is well-founded.

Exercise 1.14 (Higman's Lemma for Matrices?)**.** A quasi-ordering $(A, \leq)$ leads to a natural notion of embedding on $\mathrm{Mat}[A]$ —the set of rectangular matrices $M, N, \ldots$ with elements from A— by letting $M \leq_{\mathrm{Mat}} N$ when there is a submatrix N' of N (i.e., a matrix derived from N by removing some lines and columns) s.t. $M \leq_\times N'$ (i.e., M and N' have same dimensions and $M[i,j] \leq N'[i,j]$ for all i, j). Does $(A, \leq)$ wqo imply $(\mathrm{Mat}[A], \leq_{\mathrm{Mat}})$ wqo?

Exercise 1.15 (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

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

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

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

(1) Show that $(\mathcal{P}(A), \sqsubseteq_{\mathrm{H}})$ and $(\mathcal{P}(A), \sqsubseteq_{\mathrm{S}})$ are qos. Are they antisymmetric when $(A, \leq)$ is? Are they total when $(A, \leq)$ is? Are they well-founded when $(A, \leq)$ is?

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

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

(4) Prove the following characterisation of wqos:

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

(5) 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.

(6) Show that $(\mathcal{P}_f(R), \sqsubseteq_{\mathrm{S}})$ and $(\mathcal{P}(R), \sqsubseteq_{\mathrm{H}})$ are *not* wqos, where R is Rado's structure —see Exercise 1.11.

Exercise 1.16 (Kruskal's Tree Theorem)**.** For a qo $(A, \leq)$, we write $T(A)$ for the set of finite trees node-labelled 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 labelled 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

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

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 1.12) and construct a "minimal infinite bad sequence" $S = t_0, t_1, t_2, \ldots$ 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 $t_0, \ldots, t_{k-1}$. By construction S is infinite and is bad.

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

(2) Derive a contradiction, i.e, show that S contains an increasing pair.

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 1.17 (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

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

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

reflexive transitive compatibility

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

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 U is upward-closed, then $Pre^*_\exists(U)$ is also upward-closed. Does Lemma 1.16 still hold?

(2) Let K_0 be a finite basis of U. Lift *pb*—recall the *effective pred-basis* function from page 6—to operate on finite sets. The sequence

$$K_0 \subseteq K_1 \subseteq \cdots \text{ where } K_{n+1} \stackrel{\text{def}}{=} K_n \cup pb(K_n) \qquad (\S)$$

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 $\uparrow K = Pre^*_\exists(U)$.

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

Exercise 1.19 (Strict Compatibility)**.** We strengthen in this exercise (compatibility) to a stronger notion of compatibility that allows the decidability of finiteness. Consider the following replacement for (compatibility):

strict compatibility

$$s \to s' \wedge s < t \text{ implies } \exists t',\ s' < t',\ t \to t'\ . \qquad (\text{sc})$$

Assume that S is image-finite and has strict compatibility. We further assume, for simplification purposes, that $\leq$ is antisymmetric (i.e., it is a partial order, where different elements cannot be equivalent). A run $s_0 \to s_1 \to s_2 \to \cdots$ is *repeats-free* if $s_i \neq s_j$ whenever $i \neq j$.

(1) Show that S has an infinite repeats-free run starting from s_0 iff $Post^*(s_0)$ is infinite.

(2) Show that S has an infinite repeats-free run from s_0 iff it has a finite repeats-free run that contains an increasing pair, i.e., some $i < j$ with $s_i \leq s_j$.

(3) Conclude that the following problem is decidable for image-finite, *Post*-effective WSTSs with strict compatibility and decidable and antisymmetric $\leq$:

[Fin] Finiteness

instance: A transition system $\langle S, \to\rangle$, a qo $(S, \leq)$, and a state s_0 in S.
question: Is $Post^*(s_0)$ finite?

(4) Generalise the previous result so that one does not require antisymmetry of $\leq$.

Exercise 1.20 (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

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

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

downward WSTS

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 $s = s_0 \to s_1 \to \cdots \to s_n \leq t$?

Exercise 1.21 (WSTSs Everywhere)**.** We consider a transition system $\mathcal{S} = (S, \to)$ where S is a recursive (but otherwise arbitrary) set of configurations. For $s \in S$, let *maxtime*(s) be the length of the longest run starting from s. We let *maxtime*$(s) = \omega$ when arbitrary long runs exist. Define $s \leq_T t$ when *maxtime*$(s) \leq$ *maxtime*(t) assuming the obvious total ordering over $\mathbb{N} \cup \{\omega\}$.

(1) Show that $(S, \to, \leq_T)$ is a WSTS.

(2) Can we use WSTS theory and decide whether $\mathcal{S}$ terminates (starting from some s_0) when it is $Post$-effective and image-finite?

Program Termination

Exercise 1.22. Show that the weaker condition

$$R \subseteq T_1 \cup \cdots \cup T_k \qquad (\P)$$

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

Exercise 1.23 (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 1.24 (Cut Elimination & Subformula Property)**.** Prove Lemma 1.18.

Exercise 1.25 (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 1.26 (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 inasmuch 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 1.27. Show that $\mathbb{N}_\omega$ is a wqo.

⋆⋆ **Exercise 1.28** (Covering)**.** The aim of this exercise is to complete the proof of Theorem 1.21 and show that the set of labels $C \subseteq \mathbb{N}^d_\omega$ of the Karp & Miller tree T forms a covering according to Definition 1.20.

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

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

threshold 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 $\mathbf{A}^*$.

(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 $\mathbb{N}^d_\omega$, first define

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

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

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

its set of finite components. We introduce for this question a variant of the construction Karp & Miller graph 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.

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 labelled 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-node ω values are introduced by step 2, which we call *ω-nodes*.

Prove that, if $s_0 \xrightarrow{u}_T s$ labelled $\mathbf{y}$ 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 $w_1, \ldots, w_n$ in $\mathbf{A}^+$,
- numbers $k_1, \ldots, k_n$ 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 proofs of Higman's Lemma in Exercise 1.12 and of Kruskal's Tree Theorem in Exercise 1.16 using a "minimal bad sequence" argument is due to Nash-Williams (1963). 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 constructions of exercises 1.13 to 1.15—see (Marcone, 1994) for a good reference, and (Rado, 1954) or (Jančar, 1999) for a characterisation 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 1.17 to 1.20. 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, 2013).

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 emphasises 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 1.26 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 **R** and its variants the *entailment logic* **E** and *ticket logic* **T**. Urquhart (1999) proved $\mathbf{R}_{\supset,\wedge}$ to be Ackermannian (see CRI on page 114), while RI was shown to be 2ExpTime-complete only very recently (Schmitz, 2016a). The decidability of implicative ticket logic $\mathbf{T}_{\supset}$ was recently proven by Padovani (2013), and its complexity is 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 112. The notion of *covering* can be generalised to complete WSTS, but they are in general not finite as in the VAS case (Finkel and Goubault-Larrecq, 2012); see Chapter 4.

2

COMPLEXITY UPPER BOUNDS

As seen in Chapter 1, many algorithms rely on well quasi orderings for their proof of 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 characterisations 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 (a, b)
c ⟵ 1
while a > 0 ∧ b > 0
    l : ⟨a, b, c⟩ ⟵ ⟨a − 1, b, 2c⟩
  or
    r : ⟨a, b, c⟩ ⟵ ⟨2c, b − 1, 1⟩
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 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 than 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

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

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.

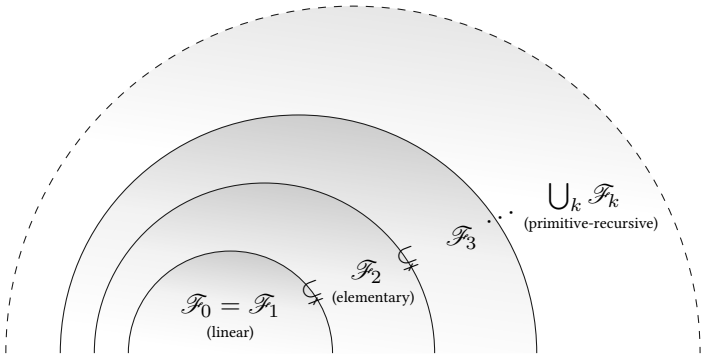

Figure 2.2: The Grzegorczyk hierarchy of primitive-recursive functions.

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 $A_{\leq n} \stackrel{\text{def}}{=} \{a \in A \mid |a|_A \leq n\}$ 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 *infinity norm* $|\langle m,n\rangle|_{\mathbb{N}^2} \stackrel{\text{def}}{=} |\langle m,n\rangle|_\infty = \max(m,n)$; 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 $a_0, a_1, a_2, \ldots$ over A is (g,n)-*controlled* for some initial norm $n \in \mathbb{N} \stackrel{\text{def}}{\Leftrightarrow}$

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

In particular, $|a_0|_A \leq n$, 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 $n = \max(a_0, b_0)$. Observe that the empty sequence is always a controlled sequence.

Definition 2.1 (Basic nwqos)**.** We note $[k]$ the nwqo $(\{0,\ldots,k-1\}, \leq, |.|_{[k]})$ defined over the initial segment of the natural numbers, where $|j|_{[k]} \stackrel{\text{def}}{=} j$ for all $0 \leq j < k$, and Γ_k the generic k-elements nwqo $(\{a_0,\ldots,a_{k-1}\}, =, |.|_{\Gamma_k})$ where $|a_j|_{\Gamma_k} \stackrel{\text{def}}{=} 0$ 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 organise such sequences in a tree by sharing common prefixes; this tree has

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

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 $L_{g,A}(n)$ depending on n (and parameterized by g and A), or as a *length function* $L_{g,A} : \mathbb{N} \to \mathbb{N}$. Our purpose will then be to obtain complexity bounds on $L_{g,A}$.

length function

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.

ISOMORPHIMS. For one thing, we will work up to isomorphism: we write $A \equiv B$ when the two nwqos 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.

nwqo isomorphism

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 $(g, 0)$-controlled bad sequence over Γ_2, but that the longest $(g, 0)$-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

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

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

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

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 $\Gamma_k \stackrel{\text{def}}{=} \Gamma_1 \cdot k$.

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

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

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

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

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 $A^d \stackrel{\text{def}}{=} \underbrace{A \times \cdots \times A}_{d \text{ times}}$; 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 $|k|_{\mathbb{N}} \stackrel{\text{def}}{=} k$ 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 $\times$:

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

Γ_1 is neutral for $\times$:

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

$+$ is associative and commutative:

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

$\times$ is associative and commutative:

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

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 SIMPLE$(2, b)$ in Equation (2.2) on page 26, 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 27).

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

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

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

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

Observe that

$$F_1(x) = 2x+1\,, \qquad F_2(x) = 2^{x+1}(x+1)-1\,, \tag{2.20}$$

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

For $k \geq 2$, each level of the Grzegorczyk hierarchy can be characterised 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 F_k are *honest*, i.e. can be computed in time bounded by a function elementary in F_k, $F_k \in \mathscr{F}_k$ 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.

We delay more formal details on $(\mathscr{F}_k)_k$ until Section 2.4 on page 40 and Exercise 2.3 on page 48 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 29), 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 26. 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 2.36 on page 46. 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 $\mathcal{S} = \langle S, \to, \leq \rangle$ 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 $s \to s'$. 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 $\ell \stackrel{\text{def}}{=} f(n) + 1$ 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

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

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 40). 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

$$n, n-1, \ldots, 1, 0 \tag{2.30}$$

of length $n + 1$.

Formally, (2.30) only proves one direction of (2.29), which is 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 $k, l, m, \ldots$ over $\mathbb{N}$, once the first element $k \leq n$ has been fixed, the remaining elements $l, m, \ldots$ have to be chosen inside a finite set $\{0, \ldots, k-1\}$ of cardinal k—or the sequence would be good. Thus this suffix, which itself has to be bad, is of length at most

$$L_{g,\Gamma_k}(n) = k \tag{2.31}$$

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

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 $a_0, a_1, a_2, \ldots$ over some nwqo A, we can distinguish between the first element a_0, which verifies $|a_0|_A \leq n$, and the suffix sequence $a_1, a_2, \ldots$, which

1. verifies $a_0 \not\leq a_i$ for all $i > 0$,
2. is itself a bad sequence—otherwise the full sequence $a_0, a_1, a_2, \ldots$ would be good,
3. is controlled by $(g, g(n))$—otherwise the full sequence $a_0, a_1, a_2, \ldots$ 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 $A/a \stackrel{\text{def}}{=} \{a' \in A \mid a \not\leq a'\}$ of elements that are not above a.

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

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

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 $a'_0 \in A_{\leq n}$ that maximises $L_{g,A/a'_0}(g(n))$ 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

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

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 $a''_0 \in A_{\leq n}$ 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

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

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 analysing the behaviour of length functions.

Example 2.12. Let us consider the case of $L_{g,[k]}(n)$ for $k \leq n + 1$: 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 $k + 1 \leq n + 1$, since then $[k + 1]_{\leq n} = [k + 1]$ and thus by the Descent Equation

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

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

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

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

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

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 34: 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 and to deduce the value of $L_{g,\mathbb{N}}(n)$. However, as already mentioned in Example 2.3 on page 29, $\Gamma_n \not\equiv [n]$ in general.

We can reconcile the analyses made for (2.29) on page 34 and in Example 2.13 by noticing that bad sequences are never shorter in Γ_n than in $[n]$. We will prove this formally using *reflections*, which let us simplify instances of the Descent Equation by replacing all A/a for $a \in A_{\leq n}$ 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 28:

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

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

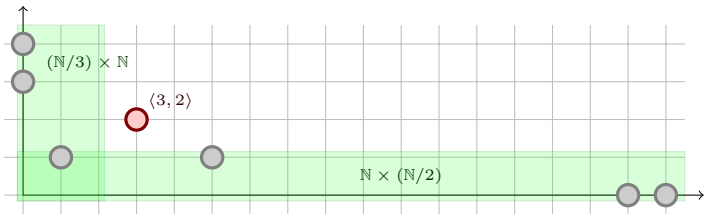

Figure 2.3: The elements of the bad sequence (2.42) and the two regions for the decomposition of $\mathbb{N}^2/\langle 3,2\rangle$.

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 $k \leq n+1$, as seen with (2.31) and (2.35), but not for $k > n+1$ as seen in Example 2.3.

Remark 2.17 (Reflection is a Preconguence). 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, so that 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 $(g,3)$-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 fulfil 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)

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

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 $(g, g(3))$-controlled: $|\langle 17,0\rangle| = 17 \geq 12 = g(g(3))$. 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 $(g, g(3))$-controlled.

By a straightforward generalisation 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 $a \in A_{\leq n}$.

2.3.3 A Bounding Function

nwqo derivation

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 $a \in A_{\leq n}$" (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 30), $\partial_n A$ is a *finite* set of polynomial nwqos also in PNF, defined inductively by

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

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

$$\partial_n \mathbb{N}^d \stackrel{\text{def}}{=} \{\mathbb{N}^{d-1} \cdot nd\} \,, \tag{2.47}$$

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

for $d > 0$ and A, B in PNF; in these definitions the $+$ operations are lifted to act upon nwqo sets S by $A + S \stackrel{\text{def}}{=} \{A + A' \mid A' \in S\}$ 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 $\partial \stackrel{\text{def}}{=} \bigcup_n \partial_n$ is well-founded.*

The definition of ∂_n verifies:

Lemma 2.19. *Let A be a polynomial nwqo in PNF and $a \in A_{\leq n}$ 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 $a \in A_{\leq n}$ 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 $\Gamma_a \hookrightarrow \Gamma_n \equiv \mathbb{N}^0 \cdot n$ as seen in Example 2.15 since $a \leq n$, 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 $k \in \mathbb{N}_{\leq n}$ and $b \in \mathbb{N}^{d-1}_{\leq n}$. 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,

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

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

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

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 $b \in B_{\leq n}$ 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

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

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 $A_{\leq n}$ is empty and then $L_{g,A}(n) = 0 \leq M_{g,A}(n)$, or there exists some $a \in A_{\leq n}$ that maximises $L_{g,A/a}(g(n))$ 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. □

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 $< \stackrel{\text{def}}{=} \{(a, a') \in A^2 \mid a \leq a' \text{ and } a' \not\leq a\}$. There are many possible *linearisations* $\prec$ 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 linearisations 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 linearisations; 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 linearisation is the natural ordering $<$ itself, which is of order type k (respectively ω):

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

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 supremum of the family of ordinals $\{0, 1, \omega, \omega^\omega, \omega^{\omega^\omega}, \ldots\}$ (in other words ε_0 is 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 $\alpha > \beta_1 \geq \cdots \geq \beta_m \geq 0$ 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 $\mathrm{CNF}(\alpha)$ 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

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

$$\sum_{i=1}^{m} \omega^{\beta_i} \oplus \sum_{j=1}^{n} \omega^{\beta'_j} \stackrel{\text{def}}{=} \sum_{k=1}^{m+n} \omega^{\gamma_k}\,, \quad \sum_{i=1}^{m} \omega^{\beta_i} \otimes \sum_{j=1}^{n} \omega^{\beta'_j} \stackrel{\text{def}}{=} \bigoplus_{i=1}^{m} \bigoplus_{j=1}^{n} \omega^{\beta_i \oplus \beta'_j}\,, \tag{2.54}$$

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 characterisation:

Fact 2.23. *For any wqos A and B*

$$o(A + B) = o(A) \oplus o(B)\,, \qquad o(A \times B) = o(A) \otimes o(B)\,. \tag{2.55}$$

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 $\mathbb{N}$

$$\partial_n \omega^d \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } d = 0 \\ \omega^{d-1} \cdot (nd) & \text{otherwise} \end{cases} \tag{2.56}$$

$$\partial_n \alpha \stackrel{\text{def}}{=} \left\{ \gamma \oplus \partial_n \omega^d \mid \alpha = \gamma \oplus \omega^d \right\} \tag{2.57}$$

$$M_{g,\alpha}(n) \stackrel{\text{def}}{=} \max_{\alpha' \in \partial_n \alpha} \left\{ 1 + M_{g,\alpha'}(g(n)) \right\} . \tag{2.58}$$

Equation (2.56) restates (2.46) and (2.47) using maximal order types, while (2.57) and (2.58) 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,*

$$L_{g,A}(n) \leq M_{g,o(A)}(n) . \tag{2.59}$$

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

$$M_{g,k}(n) = k \qquad\qquad M_{g,\omega}(n) = n + 1 . \tag{2.60}$$

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

$$\partial_n k = \begin{cases} \emptyset & \text{if } k = 0 \\ \{k-1\} & \text{otherwise} \end{cases} , \qquad \partial_n \omega = n . \tag{2.61}$$

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

$$(\gamma + \omega^{\beta+1})_x \stackrel{\text{def}}{=} \gamma + \omega^\beta \cdot \omega_x , \qquad (\gamma + \omega^\lambda)_x \stackrel{\text{def}}{=} \gamma + \omega^{\lambda_x} . \tag{2.62}$$

We always assume the standard assignment $\omega_x \stackrel{\text{def}}{=} x + 1$ in the remainder of the chapter. Note that this assignment implies $\lambda_x > 0$ for all x.

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

$$P_x(\alpha+1) \stackrel{\text{def}}{=} \alpha\,, \qquad P_x(\lambda) \stackrel{\text{def}}{=} P_x(\lambda_x)\,. \tag{2.63}$$

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)

$$P_x(\gamma+\alpha) = \gamma + P_x(\alpha)\,. \tag{2.64}$$

Observe that predecessors of ordinals in ω^ω are very similar to our derivatives: for $d = 0$, $P_n(\omega^d) = 0$ and otherwise $P_n(\omega^d) = \omega^{d-1}\cdot n + P_n(\omega^{d-1})$, which is somewhat similar to (2.56), and more generally (2.64) is reminiscent of (2.57) 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

$$h_0(x) \stackrel{\text{def}}{=} 0, \qquad h_{\alpha+1}(x) \stackrel{\text{def}}{=} 1 + h_\alpha\big(h(x)\big), \qquad h_\lambda(x) \stackrel{\text{def}}{=} h_{\lambda_x}(x). \tag{2.65}$$

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,*

$$h_\alpha(x) = 1 + h_{P_x(\alpha)}\big(h(x)\big)\,. \tag{2.66}$$

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 $0 < \lambda_x < \lambda$, 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 $h_\omega(x) = h_{x+1}(x) = x + 1$ regardless of the choice of h.

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

$$H_{\omega\cdot 2}(x) = H_{\omega+x+1}(x) = H_\omega(2x+1) + x = 3x + 1\,, \tag{2.67}$$

$$H_{\omega^2}(x) = (2^{x+1} - 1)(x+1)\,. \tag{2.68}$$

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 $\alpha' = n+2$: $M_{g,\omega}(n) = 1+M_{g,n}(g(n)) = 1+n$ but $M_{g,n+2}(n) = n+2$ by Example 2.26. Similarly with h_α, as seen with Example 2.28, $h_{x+2}(x) = x+2 > x+1 = h_\omega(x)$, although $x + 2 < \omega$.

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

structural ordering

$$\omega^{d_1} + \cdots + \omega^{d_m} \sqsubseteq \omega^{d'_1} + \cdots + \omega^{d'_n} \iff m \leq n \text{ and } \forall 1 \leq i \leq m,\ d_i \leq d'_i \tag{2.69}$$

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):

$$\alpha \sqsubseteq \alpha' \text{ and } \gamma \sqsubseteq \gamma' \text{ imply } \alpha \oplus \gamma \sqsubseteq \alpha' \oplus \gamma' \,. \tag{2.70}$$

The structural ordering rules out the previous examples, as $x + 2 \not\sqsubseteq \omega$ for any 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 (2.58) can be rewritten as

$$M_{g,\alpha}(n) = \max_{\alpha=\gamma\oplus\omega^d} \left\{1 + M_{g,\partial_{d,n}\alpha}\left(g(n)\right)\right\}. \tag{2.71}$$

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 maximising 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:

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

An important consequence of Lemma 2.31 is that there is a *maximising 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*

$$M_{g,\alpha}(n) = 1 + M_{g,\gamma+\partial_n\omega^d}(g(n)) \,. \tag{2.72}$$

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

$$\partial_{d,n'}\partial_{d',n}\alpha \sqsubseteq \partial_{d',n'}\partial_{d,n}\alpha \,. \tag{2.73}$$

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 (2.70): abusing notations for the cases of $d = 0$ or $d' = 0$,

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

Observe that $nd' + n'd \leq n'd' + nd$, i.e. that the second line has at least as many terms as the first line, and thus fulfils the first condition of the structural ordering in (2.69). Furthermore, it has at least as many $w^{d'-1}$ terms, thus fulfilling the second condition of (2.69). □

Let us conclude with a comparison between derivatives and predecessors:

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

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

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

and the statement holds. Otherwise, by (2.70)

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

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 $h(x \cdot d) \geq g(x) \cdot d$ for all x. If $\alpha < \omega^{d+1}$, then*

$$M_{g,\alpha}(n) \leq h_\alpha(nd) \; . \tag{2.74}$$

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

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

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

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

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

$$M_{g,\alpha}(n) \leq h_\alpha(nd) \; . \qquad \square$$

For instance, for $\alpha = \omega$, (and thus $d = 1$), we can choose $h = g$, and Theorem 2.34 yields that

$$M_{g,\omega}(n) \leq g_\omega(n) = n + 1 \; , \tag{2.75}$$

which is optimal (recall examples 2.26 and 2.28).

super-homogeneous function

Other examples where setting $h = g$ fits are $g(x) = 2x, g(x) = x^2, g(x) = 2^x$, etc. More generally, Theorem 2.34 can use $h = g$ if g is *super-homogeneous*, i.e. if it verifies $g(dx) \geq g(x)d$ for all $d, x \geq 1$:

Corollary 2.35. *Let $d > 0$, g be a super-homogeneous control function, and $\alpha < \omega^{d+1}$. Then $L_{g,\alpha}(n) \leq g_\alpha(nd)$.*

We sometimes need to choose h different from g: 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

$$L_{g,\mathbb{N}^d \times \Gamma_p}(n) \leq M_{g,\omega^d \cdot p}(n) \leq h_{\omega^d \cdot p}(nd) \leq F_d^{dbp}(nd) - nd \; , \tag{2.76}$$

the latter being a function in $\mathscr{F}_d$ when d, b, p are fixed according to (2.22):

Corollary 2.36. *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 $h(x) = g(x)d$, 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:

$$L_{g,A}(n) \leq M_{g,o(A)}(n)\,. \tag{2.77}$$

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

$$L_{g,A}(n) \leq h_{o(A)}(nd)\,, \tag{2.78}$$

where $h(xd) = d \cdot g(xd) \geq d \cdot g(x)$ 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 $\mathscr{F}_{\gamma+d}$. □

How good are these upper bounds? We already noted that they were optimal for $\mathbb{N}$ in (2.75), 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 generalises 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 (2.75) and Corollary 2.35. 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 $F_1(x) = 2x + 1$ and $F_2(x) = 2^{x+1}(x + 1) - 1$ (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.

(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 strictly monotone in the parameter k, more precisely that $F_{k+1}(x) > F_k(x)$ for all k, provided that $x > 0$.

⋆ **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, 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
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.

cut-off subtraction

(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 $\mathscr{F}_0$:

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 synthesised 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 $\uparrow_A a \stackrel{\text{def}}{=} \{a' \in A \mid a \leq a'\}$. Thus $A/a = A \setminus (\uparrow_A a)$. Prove the following:

$$A \times B/\langle a, b\rangle \not\equiv (A/a \times \uparrow_B b) + (A/a \times B/b) + (\uparrow_A a \times B/b) \,, \qquad (*)$$

$$A \times B/\langle a, b\rangle \not\equiv (A/a \times B) + (A \times B/b) \,. \qquad (\dagger)$$

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 ∂)**.** Recall that, when working with terms in CNF, the *ordinal ordering* $<$, which is a well ordering over ordinals, has a syntactic characterisation akin to a lexicographic ordering: ordinal ordering

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

Prove Fact 2.18: The relation $\partial \stackrel{\text{def}}{=} \bigcup_n \partial_n$ is well-founded.

Exercise 2.9 (Predecessors)**.** Prove Equation (2.64): For all $\alpha > 0$, $P_x(\gamma+\alpha) = \gamma + P_x(\alpha)$.

Exercise 2.10 (Structural Ordering)**.** Prove Equation (2.70): $\sqsubseteq$ is a precongruence for $\oplus$.

Exercise 2.11 (Structural Monotonicity)**.** Let α, α' be in ω^ω and h be a strictly monotone unary function. 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 generalisation 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 1), 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:

$$L_{g,r,A}(n) = L_{g,A\times\Gamma_r}(n) \,. \tag{§}$$

(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 $F_{\alpha+1}(x) \stackrel{\text{def}}{=} F^{\omega_x}_\alpha(x)$ 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 $h^{\omega^d \cdot d}(nd)$ 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

$$\mathbf{x} <_{\text{lex}} \mathbf{y} \iff \mathbf{x}(d) < \mathbf{y}(d) \text{ or } (\mathbf{x}(d) = \mathbf{y}(d) \text{ and } \langle \mathbf{x}(1), \dots, \mathbf{x}(d-1)\rangle <_{\text{lex}} \langle \mathbf{y}(1), \dots, \mathbf{y}(d-1)\rangle) \,.$$

This is a *linearisation* 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) Let $n > 0$, and write a program $\textsc{lex}_d(g,n)$ with d counters $\mathbf{x}(1), \dots, \mathbf{x}(d)$ 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 $\textsc{lex}_d(g,n)$ should be a sequence $(\mathbf{x}_1, \mathtt{c}_1), (\mathbf{x}_2, \mathtt{c}_2), \dots, (\mathbf{x}_\ell, \mathtt{c}_\ell)$ of pairs $(\mathbf{x}_i, \mathtt{c}_i)$ composed of a vector $\mathbf{x}_i$ in $\mathbb{N}^d$ and of a counter $\mathtt{c}_i$.

(2) Let $(\mathbf{x}_1, \mathtt{c}_1), (\mathbf{x}_2, \mathtt{c}_2), \dots, (\mathbf{x}_\ell, \mathtt{c}_\ell)$ be the unique run of $\textsc{lex}_d(g,n)$ 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 $i > 0$,

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

(3) Deduce (¶).

(4) 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 $\textsc{lex}_d(g, n)$ 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) and Abriola et al. (2015) 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: for instance, 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 Kruskal's Tree Theorem.

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.7, A.10, and A.17. Exercises 2.4 and 2.12 are taken from (Figueira et al., 2011). Equation (¶) in Exercise 2.15 relates the Cichoń hierarchy with the length of bad sequences for the lexicographic ordering over $\mathbb{N}^d$, i.e. with the length of decreasing sequences over ω^d; this holds more generally for any ordinal below ε_0 along with an appropriate norm, see (Schmitz, 2014).

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.

Hardness Theorem

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

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.

Minsky machine

For *Minsky machines*, the tests are zero-tests and the updates are increments and decrements.

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 $M = (Loc, C, \Delta)$ where $Loc = \{\ell_1, \ldots, \ell_p\}$ is a finite set of *locations*, $C = \{\mathtt{c}_1, \ldots, \mathtt{c}_n\}$ is a finite set of *counters*, and $\Delta \subseteq Loc \times OP(C) \times Loc$ 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 labelled with an instruction $op \in OP(C)$ 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 $OP(C)$ is given by the following abstract grammar:

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

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 $M = (Loc, C, \Delta)$ is given under the form of transitions between its configurations. Formally, a *configuration* (written $\sigma, \theta, \ldots$)

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

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

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

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

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

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

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

As usual, we write $\sigma \xrightarrow{\Delta}_{\text{std}} \sigma'$, or just $\sigma \to_{\text{std}} \sigma'$, when $\sigma \xrightarrow{\delta}_{\text{std}} \sigma'$ for some $\delta \in \Delta$. Chains $\sigma_0 \to_{\text{std}} \sigma_1 \to_{\text{std}} \cdots \to_{\text{std}} \sigma_m$ of consecutive steps, also called *runs*, are denoted $\sigma_0 \to^*_{\text{std}} \sigma_m$, and also $\sigma_0 \to^+_{\text{std}} \sigma_m$ when $m > 0$. Steps may also be written more precisely under the form $M \vdash \sigma \to_{\text{std}} \sigma'$ 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 $\sigma_0 \to \sigma_1 \to \cdots \to \sigma_m$ 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 3.4.

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

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

$\sigma \leq \sigma'$ 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:

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

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

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

3.1.4 Behavioural 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 $M \vdash \sigma_{\text{ini}} \to^* \sigma_{\text{goal}}$?

Coverability: given a CM M and two configurations σ_{ini} and σ_{goal}, is there a run $M \vdash \sigma_{\text{ini}} \to^* \sigma$ 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 $M \vdash \sigma_{\text{ini}} \to \sigma_1 \to \cdots \to \sigma_n \to \cdots$?

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* $(H^\alpha \colon \mathbb{N} \to \mathbb{N})_{\alpha < \varepsilon_0}$ 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 49, but let us recall the following:

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

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 $H^\omega(n) = H^n(n) = 2n+1$, $H^{\omega^2}(n) = 2^{n+1}(n+1)-1$ is exponential, H^{ω^3} non-elementary, and H^{ω^ω} Ackermannian; in fact we set in this chapter

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

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 44:

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

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

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:

$$\langle a_m, \ldots, a_0 + 1; n \rangle \to \langle a_m, \ldots, a_0; n+1 \rangle \ , \tag{D_1}$$

$$\langle a_m, \ldots, a_k + 1, \overbrace{0, \ldots, 0}^{k>0 \text{ zeroes}}; n \rangle \to \langle a_m, \ldots, a_k, n+1, \overbrace{0, \ldots, 0}^{k-1 \text{ zeroes}}; n \rangle \ . \tag{D_2}$$

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 $M_H(m) = (Loc_H, C, \Delta_H)$, or M_H for short, with $C = \{\mathtt{a}_0, \mathtt{a}_1, ..., \mathtt{a}_m, \mathtt{n}\}$. 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 $M_H \vdash (\ell_H, \mathbf{a}, n) \to^*_{std} (\ell_H, \mathbf{a}', n')$.*

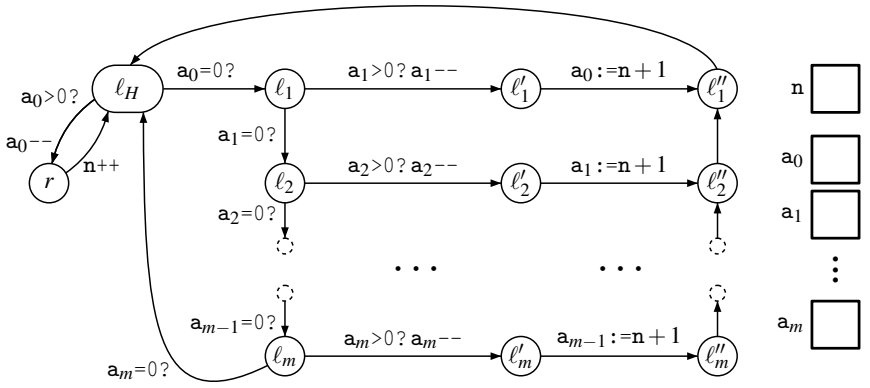

Figure 3.1: $M_H(m)$, a counter machine that implements $\xrightarrow{H}$.

---

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 $\mathtt{a}_i$ 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,

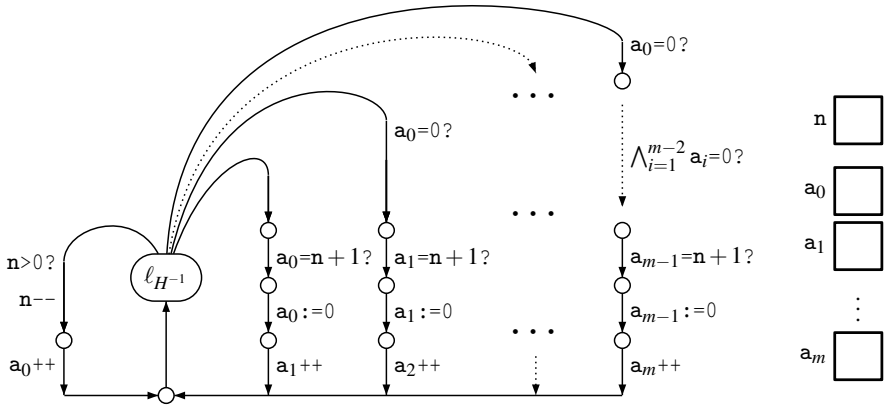

Figure 3.2: $M_{H^{-1}}(m)$, a counter machine that implements $\xrightarrow{H} {}^{-1}$.

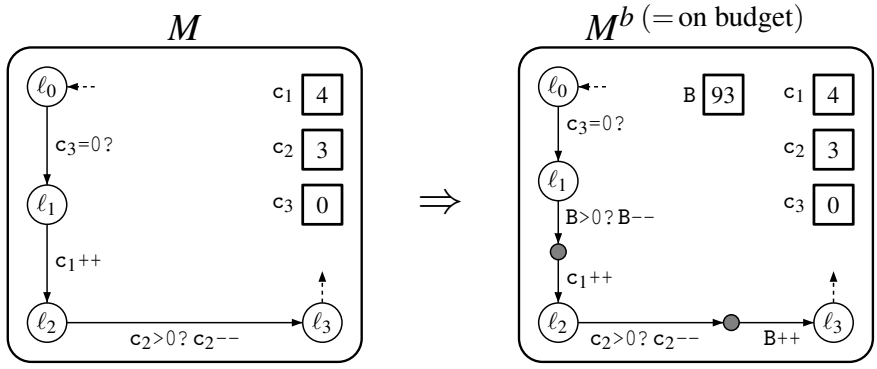

Figure 3.3: From M to M^b (schematically).

---

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 grey) 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 behaviour 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* $M^b \vdash (\ell, B, \mathbf{a}) \to^*_{std} (\ell', B', \mathbf{a}')$ *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 $M^b \vdash (\ell, B, \mathbf{a}) \to^*_{std} (\ell', B', \mathbf{a}')$ and $\ell, \ell' \in$ Loc then $M \vdash (\ell, \mathbf{a}) \to^*_{std} (\ell', \mathbf{a}')$.*

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

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 $M^b \vdash (\ell, B, \mathbf{a}) \to^*_{lossy} (\ell', B', \mathbf{a}')$ with $\ell, \ell' \in$ Loc. Then $M^b \vdash (\ell, B, \mathbf{a}) \to^*_{std} (\ell', B', \mathbf{a}')$ 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 hypothesised run $M^b \vdash (\ell, B, \mathbf{a}) = \sigma_0 \to_{\text{lossy}} \sigma_1 \to_{\text{lossy}} \cdots \to_{\text{lossy}} \sigma_n = (\ell', B', \mathbf{a}')$. Coming back to (3.2), these lossy steps require, for $i = 1, \ldots, n$, some reliable steps $\theta_{i-1} \to_{\text{std}} \theta'_i$ 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 $M^b \vdash (\ell, B, \mathbf{0}) \to^*_{lossy} (\ell', B', \mathbf{a})$ with $\ell, \ell' \in$ Loc. Then:*

1. *$B \geq B' + |\mathbf{a}|$, and*
2. *if $B = B' + |\mathbf{a}|$, then $M \vdash (\ell, \mathbf{0}) \to^*_{std} (\ell', \mathbf{a})$. 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 $(\ell_H, \mathrm{a}_m{:}1, \mathbf{0}, \mathrm{n}{:}m, \mathbf{0}) \to^*_{lossy} \theta$ for some configuration θ with $\theta \geq (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$.*

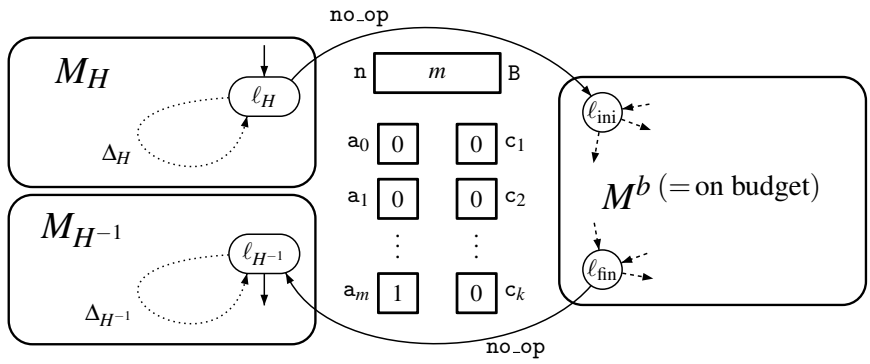

Figure 3.4: Constructing $M(m)$ from M^b, M_H and $M_{H^{-1}}$.

---

2. *M^b has a lossy run* $(\ell_{\text{ini}}, \text{B:}Ack(m), \mathbf{0}) \to^*_{lossy} (\ell_{\text{fin}}, Ack(m), \mathbf{0})$.
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:

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

  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* $\Rightarrow$ *2*" and "*4* $\Rightarrow$ *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 $Ack(m)$ 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 behaviour of M and $R(M)$ are related in the following way:

Lemma 3.13.

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

In other words, the reliable behaviour of $R(M)$ contains the reliable behaviour of M and is contained in the lossy behaviour 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* $(\ell_H, \mathtt{a}_m{:}1, \mathbf{0}, \mathtt{n}{:}m, \mathbf{0}) \to^*_{std} (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$.

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

3. M *has a reliable run* $(\ell_{\text{ini}}, \mathbf{0}) \to^*_{std} (\ell_{\text{fin}}, \mathbf{0})$ *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*$\Rightarrow$*5*" in Proposition 3.12.

For *3* $\Rightarrow$ *2*: We obtain a reliable run in $M(m)$ ("*5*$\Rightarrow$*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*$\Rightarrow$*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 behaviour and the lossy behaviour 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,

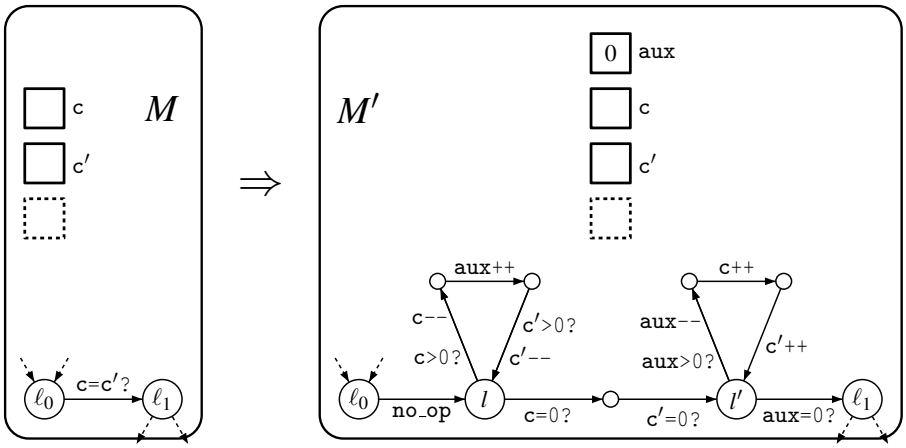

Figure 3.5: From M to M': eliminating equality tests.

---

2. $C' = C + \{\mathtt{aux}\}$ 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 $\mathtt{aux}$ 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 3.2. 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:

- $\mathtt{c} - \mathtt{c}'$ is a numerical invariant in l, and also in l',
- $\mathtt{c} + \mathtt{aux}$ is a numerical invariant in l, and also in 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.

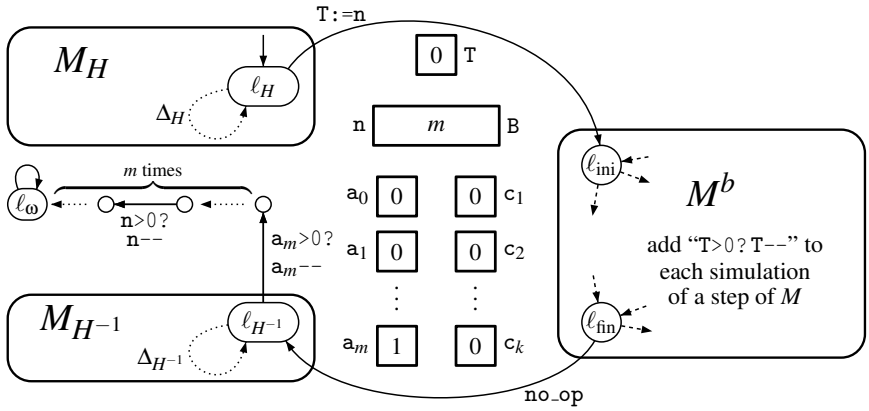

Figure 3.6: Hardness for termination: A new version of $M(m)$.

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 $M' \vdash (\ell_0, c, c', 0) \to^*_{\text{lossy}} (\ell_1, d, d', 0)$ implies $d, d' \leq \min(c, c')$. In such a case, M obviously has a lossy step $M \vdash (\ell_0, c, c') \to_{\text{lossy}} (\ell_1, d, d')$.

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 initialised 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* $(\ell_H, 1, \mathbf{0}, \mathrm{n}{:}m, \mathbf{0}) \to^*_{lossy} \theta \geq (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$.
2. $R(M(m))$ *has a lossy run* $(\ell_H, 1, \mathbf{0}, \mathrm{n}{:}m, \mathbf{0}) \to^*_{lossy} \theta \geq (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$.
3. M *has a reliable run* $(\ell_{\mathrm{ini}}, \mathbf{0}) \to^*_{std} (\ell_{\mathrm{fin}}, \mathbf{0})$ *of length at most* $Ack(m)$.

Finally, we add a series of $m+1$ transitions that leave from $\ell_{H^{-1}}$, and check that $\sigma_{\mathrm{goal}} \stackrel{\text{def}}{=} (\ell_{H^{-1}}, 1, \mathbf{0}, m, \mathbf{0})$ is covered, i.e., that $\mathtt{a}_m$ 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 $Ack(m)$, which is an Ackermann-hard problem.

Exercises

Exercise 3.1. Describe the complete Hardy computations starting from $\alpha; 0$ for $\alpha = \omega$, $\omega \cdot 2$, $\omega \cdot 3$, ω^2, ω^3, ω^ω, and ω^{ω^ω}.

Exercise 3.2 (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 3.3 (Transfer Machines)**.** *Transfer machines* are extended counter machines with instruction set reduced to increments, decrements, and *transfers* transfer machine

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

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 3.4 (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 $M \vdash \sigma \to_{\mathrm{inc}} \sigma'$, and defined by: incrementing counter machine

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

Incrementation errors are thus the symmetrical mirror of losses.

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

$$M \vdash \sigma_1 \to_{\mathrm{std}} \sigma_2 \text{ iff } M^{-1} \vdash \sigma_2 \to_{\mathrm{std}} \sigma_1. \tag{$\dagger$}$$

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 3.3).

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 3.4); 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 reader will find an up-to-date list of decision problems complete for Ackermann complexity in Appendix B.

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

4

IDEALS

The Finite Basis Property of wqos yields finite presentations both for upward-closed subsets, using minimal elements $\bigcup_{1\leq i\leq n} \uparrow x_i$, and for downward-closed subsets, using excluded minors $\bigcap_{1\leq i\leq n} X \smallsetminus \uparrow x_i$. The latter representation however does not lend itself nicely to algorithmic operations. For instance, representing $\downarrow x$ using excluded minors is rather inconvenient.

By contrast, the *ideals* of a wqo provide finite decompositions for downward-closed sets: they are used in algorithms as data structures to represent downward-closed subsets, which mirrors the use of finite bases of minimal elements to represent upward-closed subsets.

In this section we present some basic facts about ideals of wqos and some of their applications to coverability problems. Our emphasis is on genericity: we show how to handle ideals for wqos of the form A^*, $A \times B$, etc., with mild assumptions on the wqos A, B, ...

4.1 Ideals of WQOs

In classical order theory, a subset of a qo $(X, \leq)$ that is $\uparrow x$ for some element x of X is also called a *principal filter*. One way to rephrase the Finite Basis Property is to say that, in any wqo, the upward-closed subsets are finite unions of principal filters. This leads to the natural representation of upward-closed sets in wqos by their minimal elements.

principal filter

There is a dual notion of *principal ideals*, which are all downward-closed subsets of the form $\downarrow x$, where x is any element. In general, one cannot represent all downward-closed subsets as finite unions of principal ideals and this is why we need the more general notion of ideals.

principal ideals

Recall that we write $Up(X)$ —or just Up when X is understood— for the set of upward-closed subsets of X, with typical elements $U, U', \ldots$ Similarly, $Down(X)$,

or just *Down* denotes the set of its downward-closed subsets, with typical elements $D, D', \ldots$

4.1.1 Prime Decompositions of Order-Closed Subsets

prime **Definition 4.1** (Prime Subsets)**.** Let $(X, \leq)$ be a qo.

1. A nonempty $U \in \mathit{Up}(X)$ is *(up) prime* if for any $U_1, U_2 \in \mathit{Up}$, $U \subseteq (U_1 \cup U_2)$ implies $U \subseteq U_1$ or $U \subseteq U_2$.

2. Similarly, a nonempty $D \in \mathit{Down}(X)$ is *(down) prime* if $D \subseteq (D_1 \cup D_2)$ implies $D \subseteq D_1$ or $D \subseteq D_2$.

Observe that all principal filters are up primes and that all principal ideals are down primes.

Lemma 4.2 (Irreducibility)**.** *Let $(X, \leq)$ be a qo.*

1. *$U \in \mathit{Up}(X)$ is prime if, and only if, for any $U_1, \ldots, U_n \in \mathit{Up}$, $U = U_1 \cup \cdots \cup U_n$ implies $U = U_i$ for some i.*

2. *$D \in \mathit{Down}(X)$ is prime if, and only if, for any $D_1, \ldots, D_n \in \mathit{Down}$, $D = D_1 \cup \cdots \cup D_n$ implies $D = D_i$ for some i.*

Proof. We prove 1, the downward-case being identical.
"$\Rightarrow$": By Definition 4.1 (and by induction on n) U prime and $U = U_1 \cup \cdots \cup U_n$ imply $U \subseteq U_i$ for some i, hence $U = U_i$.
"$\Leftarrow$": We check that U meets the criterion for being prime: assume that $U \subseteq U_1 \cup U_2$ for some $U_1, U_2 \in \mathit{Up}$. Then, letting $U_i' \stackrel{\text{def}}{=} U_i \cap U$ for $i = 1, 2$ gives $U = U_1' \cup U_2'$. The assumption of the lemma entails that $U = U_i'$ for some $i \in \{1, 2\}$. Then $U \subseteq U_i$. □

We say that a collection $\{P_1, \ldots, P_n\}$ of up (resp. down) primes is a *(finite) prime decomposition* of $U \in \mathit{Up}(X)$ (resp., of $D \in \mathit{Down}(X)$) if $U = P_1 \cup \cdots \cup P_n$ (resp. $D = P_1 \cup \cdots \cup P_n$). Such decompositions always exist:

Lemma 4.3 (Finite Prime Decomposition)**.** *Let $(X, \leq)$ be a wqo.*

1. *Every upward-closed subset $U \in \mathit{Up}(X)$ is a finite union of up primes.*

2. *Every downward-closed subset $D \in \mathit{Down}(X)$ is a finite union of down primes.*

Proof of 2. By well-founded induction on D. For this, recall that $(\mathit{Down}(X), \subseteq)$ is well-founded (Definition 1.6). If D is empty, it is an empty, hence finite, union of primes. If $D \neq \emptyset$ is not prime, then by Lemma 4.2 it can be written as $D = D_1 \cup \cdots \cup D_n$ with $D \supsetneq D_i$ for all $i = 1, \ldots, n$. By induction hypothesis each D_i is a finite union of primes. Hence D is too. □

Proof of 1. We take a different route: we use the Finite Basis Property, showing that any upward-closed set is a finite union of principal filters. □

A finite prime decomposition $\{P_1, \ldots, P_n\}$ is *minimal* if $P_i \subseteq P_j$ implies $i = j$. We can now state and prove the Canonical Decomposition Theorem.

Theorem 4.4 (Canonical Decomposition)**.** *Let $(X, \leq)$ be a wqo. Any upward-closed U (resp. downward-closed D) has a finite minimal prime decomposition. Furthermore this minimal decomposition is unique. We call it the* canonical decomposition *of U (resp. D).*

canonical decomposition

Proof. By Lemma 4.3, any U (or D) has a finite decomposition: U (or D) $= \bigcup_{i=1}^{n} P_i$. The decomposition can be assumed minimal by removing any P_i that is strictly included in some P_j. To prove uniqueness we assume that $\bigcup_{i=1}^{n} P_i = \bigcup_{j=1}^{m} P'_j$ are two minimal decompositions. From $P_i \subseteq \bigcup_j P'_j$, and since P_i is prime, we deduce that $P_i \subseteq P'_{k_i}$ for some k_i. Similarly, for each P'_j there is ℓ_j such that $P'_j \subseteq P_{\ell_j}$. These inclusions are equalities since $P_i \subseteq P'_{k_i} \subseteq P_{\ell_{k_i}}$ requires $i = \ell_{k_i}$ by minimality of the decomposition. Similarly $j = k_{\ell_j}$ for all j. This one-to-one correspondence shows $\{P_1, \ldots, P_n\} = \{P'_1, \ldots, P'_m\}$. □

4.1.2 Ideals

Definition 4.5 (Directed Sets)**.** A non-empty subset S of a qo $(X, \leq)$ is *(up) directed* if for every $x_1, x_2 \in S$, there exists $x \in S$ such that $x_1 \leq x$ and $x_2 \leq x$. It is *(down) directed* if for every $x_1, x_2 \in S$, there exists $x \in S$ such that $x_1 \geq x$ and $x_2 \geq x$.

directed

Definition 4.6 (Ideals and Filters)**.** A downward-closed up directed subset of a qo $(X, \leq)$ is an *ideal* of X. An upward-closed down directed subset is a *filter* of X.

ideal

filter

We observe that every principal ideal $\downarrow x$ is up directed and is therefore an ideal, and similarly every principal filter $\uparrow x$ is down directed and is therefore a filter. Conversely, when $(X, \leq)$ is a wqo, any filter F is a principal filter: since it is upward-closed, by the Finite Basis Property it has a finite basis of incomparable minimal elements $\{x_1, \ldots, x_n\}$, and since it is down directed we must have $n = 1$, i.e. $F = \uparrow x_1$. However, not all ideals of a wqo are principal. For example, in $(\mathbb{N}, \leq)$, the set $\mathbb{N}$ itself is an ideal (it is up directed) and not of the form $\downarrow n$ for any $n \in \mathbb{N}$. In the following, we focus on ideals and mean "up directed" when writing "directed." We write $\mathit{Idl}(X)$ for the set of all ideals of X.

Example 4.7. If $(A, \leq)$ is a *finite* wqo, its ideals are exactly the principal ideals: $\mathit{Idl}(A) = \{\downarrow a \mid a \in A\}$.

In the case of $\mathbb{N}$, the ideals are exactly the principal ideals and the whole set itself: $\mathit{Idl}(\mathbb{N}) = \{\downarrow n \mid n \in \mathbb{N}\} \cup \{\mathbb{N}\}$.

We can now relate Definition 4.6 with the previous material. The following proposition justifies our interest of ideals. We did not use it as a definition, since many properties are easier to see in terms of directed downward-closed subsets.

Proposition 4.8. *Let $(X, \leq)$ be a wqo.*

1. *The up primes are exactly the filters.*
2. *The down primes are exactly the ideals.*

The first item is clear and we focus on the second.

Proof of "$\subseteq$". We only have to check that a prime P is directed. Assume it is not. Then it contains two elements x_1, x_2 such that $\uparrow x_1 \cap \uparrow x_2 \cap P = \emptyset$. In other words, $P \subseteq (P \smallsetminus \uparrow x_1) \cup (P \smallsetminus \uparrow x_2)$. But $P \smallsetminus \uparrow x_i$ is downward closed for both $i = 1, 2$, so P, being prime, is included in one $P \smallsetminus \uparrow x_i$. This contradicts $x_i \in P$. □

Proof of "$\supseteq$". Consider an ideal $I \subseteq X$. It is downward-closed hence has a canonical prime decomposition $I = P_1 \cup \cdots \cup P_n$. Minimality of the decomposition implies that, for all $i = 1, \ldots, n$, $P_i \not\subseteq \bigcup_{j \neq i} P_j$, hence there is some $x_i \in P_i$ with $x_i \notin \bigcup_{j \neq i} P_j$. If $n \geq 2$, we consider the elements $x_1, x_2 \in I$ as we have just defined. By directedness of I, $x_1, x_2 \leq y$ for some y in I, i.e., in some P_j. Hence x_1 and x_2 are in P_j, which contradicts their construction. We conclude that $n < 2$. The case $n = 0$ is impossible since ideals are nonempty. Hence $n = 1$ and $I = P_1$ is prime. □

Primality of ideals and the Canonical Decomposition Theorem are the key to understanding the benefits of using ideal decompositions as a representation for downward-closed subsets of a wqo. We will discuss implementations and data structures later in Section 4.3 since this requires considering specific wqos, but for the time being let us mention that deciding inclusion $D \subseteq D'$ between two downward-closed subsets given via prime decompositions $D = I_1 \cup \cdots \cup I_n$ and $D' = I'_1 \cup \cdots \cup I'_m$ reduces to a quadratic number $n \times m$ of comparisons between ideals thanks to the primality of ideals, see Eq. (4.2).

We conclude this section with two properties that are sometimes useful for characterising the set of ideals of a given wqo:

Lemma 4.9. *Let $(X, \leq)$ be a wqo, and let $\mathcal{J} \subseteq \mathit{Idl}(X)$ be such that every $D \in \mathit{Down}(X)$ is a finite union of ideals from $\mathcal{J}$. Then $\mathcal{J} = \mathit{Idl}(X)$.*

Proof. Let $I \in \mathit{Idl}(X)$. Since $I \in \mathit{Down}(X)$, $I = J_1 \cup \ldots \cup J_n$ with each $J_i \in \mathcal{J}$. By Lemma 4.2, $I \subseteq J_i$ for some I, and hence $I = J_i \in \mathcal{J}$. □

Lemma 4.10 (Countability)**.** *$\mathit{Up}(X)$, $\mathit{Down}(X)$ and $\mathit{Idl}(X)$ are countable when X is a countable wqo.*

Proof. Recall Lemma 1.7 showing that $U \in \mathit{Up}(X)$ can be characterised by its finite basis, and note that there are only countably many such bases when X is countable. Hence $\mathit{Up}(X)$ is countable, and then $\mathit{Down}(X)$ too since complementation provides a bijection between $\mathit{Up}(X)$ and $\mathit{Down}(X)$. Finally, $\mathit{Idl}(X)$ is a subset of $\mathit{Down}(X)$, hence is countable too. □

4.2 Effective Well Quasi-Orders and Ideals

Our goal is to present generic algorithms based on the fundamental structural properties of wqos and their ideals exhibited in the previous section.

This requires some basic computational assumptions on the wqos at hand. Such assumptions are often summarised informally as "*the wqo $(X, \leq)$ is effective*" and their precise meaning is often defined at a later stage, when one gives sufficient conditions based on the algorithm one is describing. This is just what we did with Proposition 1.15 or Proposition 1.17 in Section 1.2. Sometimes the effectiveness assumptions are not spelled out formally, e.g., when one has in mind applications where the wqo is $(\mathbb{N}^k, \leq_\times)$ or $(\Sigma^*, \leq_*)$ which are obviously "effective" under all expected understandings.

In this chapter, we not only want to consider arbitrary "effective" wqos, we also want to prove that combinations of these effective wqos produce wqos that are effective too. For this purpose, giving a formal definition of effectiveness cannot be avoided. We use a layered definition with a core notion—effective wqos, see Definition 4.11—and the more complete notion of ideally effective wqos, see Definition 4.12. Rather than being completely formal and talk of recursive languages or Gödel numberings, we will allow considering more versatile data structures like terms, tuples, graphs, etc., since we sometimes mention that an algorithm runs in linear-time and these low-level complexity issues may depend on the data structures one uses.

4.2.1 Effective WQOs

Definition 4.11 (Effective WQOs)**.** An *effective wqo* is a wqo $(X, \leq)$ satisfying the following axioms: effective wqo

(XR) There is a computational representation for X which is recursive (*i.e.*, membership in X, is decidable);

(OD) The ordering $\leq$ is decidable (for the above representation);

(IR) There is a computational representation for the ideals of X, which is recursive as well;

(ID) Inclusion of ideals is decidable.

An effective wqo is usually provided via two recursive languages or some other data structures, one for X and one for $\mathit{Idl}(X)$, with two procedures, one for comparing elements and one for comparing ideals.

The motivation behind this first definition is to allow fixing the representation of upward and downward closed sets that will be used in the rest of these notes. Indeed, throughout the rest of the chapter, we assume that upward-closed sets are represented by finite sequences of elements (denoting the union of principal filters generated by said elements) and that downward-closed sets are represented by finite sequences of ideals (again denoting their union), using the data structure provided by (XR). Here we rely on Lemma 4.3, showing that any upward-closed or downward-closed set can be represented in this way.

Note that, for an effective wqo, inclusion between upward-closed sets or between downward-closed sets is decidable. Indeed, assume that U and U' are represented by some finite decompositions of principal filters: $U = \bigcup_i F_i$ and $U' = \bigcup_j F'_j$. Then, by Lemma 4.2, one has the following equivalence:

$$U \subseteq U' \text{ iff } \forall i : \exists j : F_i \subseteq F'_j \,. \tag{4.1}$$

So comparing upward-closed sets reduces to comparing principal filters which is decidable in effective wqos. Exactly the same reasoning allows to compare downward-closed sets:

$$D = \bigcup_i I_i \subseteq D' = \bigcup_j I'_j \text{ iff } \forall i : \exists j : I_i \subseteq I'_j \,. \tag{4.2}$$

Note also that the chosen representation for $\mathit{Up}(X)$ and $\mathit{Down}(X)$ makes union trivially computable. Of course, we may also be interested in computing intersections or complements, and for this we make more effectiveness assumptions.

4.2.2 Ideally Effective WQOs

Now that representations are fixed for X, $\mathit{Idl}(X)$, and consequently also for $\mathit{Up}(X)$ and $\mathit{Down}(X)$, our next definition lists effectiveness assumptions on more set-theoretical operations.

ideally effective wqo

Definition 4.12 (Ideally Effective WQOs)**.** An *ideally effective wqo* is an effective wqo further equipped with procedures for the following operations:

(CF) Computing the complement of any filter F as a downward-closed set, denoted $\neg F$,

(CI) Computing the complement of any ideal I as an upward-closed set, denoted $\neg I$,

(XF) Providing a filter decomposition of X,

(XI) Providing an ideal decomposition of X,

(IF) Computing the intersection of any two filters as an upward-closed set,

(II) Computing the intersection of any two ideals as a downward-closed set,

(IM) Deciding if $x \in I$, given $x \in X$ and $I \in Idl(X)$,

(PI) Computing the principal ideal $\downarrow x$ given $x \in X$.

The procedures witnessing these axioms will be called an *ideal (effective) presentation* of $(X, \leq)$.

Observe that assuming that intersection and complements are computable for filters and ideals implies that it is computable for upward and downward closed sets as well. Indeed, intersection distributes over unions, and complements of a union of filters (resp. ideals) can be handled using complementation for filters (resp. ideals) and intersection for ideals (resp. filters). It only remains the case of an empty union, i.e. for complementing the empty set (which is both upward and downward closed). This is achieved using (XF) and (XI).

Similarly, membership of $x \in X$ in some upward or downward closed set reduces to membership in one of the filters or ideals of its decomposition. Filters membership simply uses (OD): $x \in \uparrow x'$ iff $x \geq x'$. But for ideals, it is *a priori* necessary to assume (IM). The same goes for computing principal filters and principal ideals. Our representation of filters makes the function $x \mapsto \uparrow x$ trivial, but $x \mapsto \downarrow x$ may be quite elaborate (even if it is easy in most concrete examples).

The previous definition obviously contains some redundancies. For instance, ideal membership (IM) is simply obtained using (PI) and (ID): $x \in I$ iff $\downarrow x \subseteq I$. With the following proposition we show that we only need to assume four of the eight axioms above to obtain an equivalent definition, and in the last proposition of this section we prove that this is a minimal system of axioms.

Proposition 4.13. *From a presentation of an effective wqo $(X, \leq)$ further equipped with procedures for (CF), (II), (PI) and (XI), one can compute an ideal presentation for $(X, \leq)$.*

Proof. We explain how to obtain the missing procedures:

(IM) As mentioned above, membership can be tested using (PI) and (ID): $x \in I$ iff $\downarrow x \subseteq I$.

(CI) We actually show a stronger statement, denoted (CD), that complementing an arbitrary downward closed set is computable. This strengthening is used for (IF).

Let D be an arbitrary downward closed set. We compute $\neg D$ as follows:

1. Initialise $U := \emptyset$;

2. While $\neg U \not\subseteq D$ do
   (a) pick some $x \in \neg U \cap \neg D$;
   (b) set $U := U \cup {\uparrow} x$

Every step of this high-level algorithm is computable. The complement $\neg U$ is computed using the description above: $\neg \bigcup_{i=1}^{n} {\uparrow} x_i = \bigcap_{i=1}^{n} \neg {\uparrow} x_i$ which is computed with (CF) and (II) (or with (XI) in case $n = 0$, i.e., for $U = \emptyset$). Then, inclusion $\neg U \subseteq D$ is tested with (ID). If this test fails, then we know $\neg U \cap \neg D$ is not empty. To implement step (a) we enumerate all the elements $x \in X$, each time testing them for membership in U and in D. Eventually, we will find some $x \in \neg U \cap \neg D$.

To prove partial correctness we use the following loop invariant: U is upward closed and $U \subseteq \neg D$. The invariant holds at initialisation and is preserved by the loop's body since ${\uparrow} x$ is upward closed, and since $x \notin D$ and D downward-closed imply ${\uparrow} x \subseteq \neg D$. Thus when/if the loop terminates, both the invariant $U \subseteq \neg D$ and the loop exit condition $\neg U \subseteq D$ hold. Then $U = \neg D$ is the desired result.

Finally, the algorithm terminates since it builds a strictly increasing sequence of upward closed sets, which must be finite (see Definition 1.6).

(IF) This follows from (CF) and (CD), by expressing intersection in terms of complement and union.

(XF) Using (CD) we can compute $\neg\emptyset$. □

Remark 4.14 (On Definition 4.12). The above methods are generic but in many cases there exist simpler and more efficient ways of implementing (CI), (IF), etc. for a given wqo. This is why Definition 4.12 lists eight requirements instead of just four: we will try to provide efficient implementations for all eight.

Do we?

As seen in the above proof, the fact that (CF), (II), (PI) and (XI) entail (CI) is non-trivial. The algorithm for (CD) computes an upward-closed set U from an oracle answering queries of the form "Is $U \cap I$ empty?" for ideals I. This is an instance of the Generalised Valk-Jantzen Lemma, an important tool for showing that some upward-closed sets are computable.

The existence in our definition of redundancies as exhibited by Proposition 4.13 raises the question of whether there are other redundancies. The following proposition answers negatively.

Proposition 4.15 (Halfon). *There are no generic and effective way to produce a procedure for axiom A given an effective wqo and procedures for axioms B, C and D, where $\{A, B, C, D\} = \{$(CF), (II), (PI), (XI)$\}$.*

An important result is that wqos obtained by standard constructions on simpler wqos usually inherit the ideal effectiveness of their component under very

mild assumptions. We just show two simple but important examples in the next section.

4.3 Some Ideally Effective WQOs

Our goal in this section is to argue that many wqos used in computer science are in fact ideally effective, allowing the use of ideal-based representations and algorithms for their downward-closed subsets. Our strategy is to consider the main operations for constructing new wqos from earlier "simpler" wqos and show that the new wqos inherit ideal effectiveness from the earlier ones. The whole section is based on recent, still preliminary, work. In these notes we only describe in details the simplest cases and give pointers to some more elaborate constructions.

4.3.1 Base Cases

Many basic wqos are easily seen to be ideally effective. We consider here a very simple case that can help the reader understand how to prove such results.

Proposition 4.16. *The wqo $(\mathbb{N}, \leq)$ of natural numbers is ideally effective.*

More elaborate examples, e.g., recursive ordinals or Rado's structure, are considered in the exercises section.

Now to the proof of Proposition 4.16. Recall that *Idl*$(\mathbb{N})$ consists of all $\downarrow n$ for $n \in \mathbb{N}$, together with the set $\mathbb{N}$ itself. Ordered with inclusion, this is isomorphic to $(\omega+1)\setminus\{0\}$, i.e., the set of non-null ordinals up to and including ω. A natural data structure for *Idl*$(\mathbb{N})$ is thus to use the number "n" to denote $\downarrow n$ and the symbol "ω" to denote $\mathbb{N}$.[1]

There remains to check that the axioms for ideal effectiveness are satisfied. First, (XR) and (OD) are obvious since they are about the data structure for $(\mathbb{N}, \leq)$ itself. Also, the data structure we just provided for *Idl*$(\mathbb{N})$ obviously satisfies (IR), and regarding (ID) (inclusion test for the ideals), we note that it reduces to comparing naturals complemented with $I \subseteq \omega$ for all $I \in$ *Idl*$(\mathbb{N})$ and $\omega \not\subseteq n$ for all $n \in \mathbb{N}$.

To show that the remaining axioms are satisfied, we rely on Proposition 4.13 and only have exhibit four procedures:

(CF) the complement of $\uparrow n$ for $n > 0$ is simply $\downarrow(n-1)$ and is empty otherwise;

(II) the intersection of two ideals amounts to computing greatest lower bounds in $\mathbb{N} \cup \{\omega\}$;

(PI) obtaining $\downarrow n$ from n is immediate with our choice of a data structure;

[1] From the ordinal viewpoint where α is exactly $\downarrow_< \alpha$, it would make more sense to use $n+1$ to denote $\downarrow n$, but this would go against the established practice in the counter systems literature.

(XI) the set $\mathbb{N}$ itself is represented by ω.

The same easy techniques can be used to show that any finite qo and any recursive ordinal is an ideally effective wqo. Or that Rado's structure is, see Exercise 4.8.

4.3.2 Sums and Products

Recall that, when A_1 and A_2 are wqos, the disjoint sum $A_1 + A_2$—see (2.5–2.7) for the definition—is a wqo. The following proposition is easy to prove (see Exercise 4.4):

Proposition 4.17 (Disjoint Sum)**.** *The ideals of $A_1 + A_2$ are all sets of the form $\{i\} \times I$ for $i = 1, 2$ and I an ideal of A_i. Thus $Idl(A_1 + A_2) \equiv Idl(A_1) + Idl(A_2)$.*
Furthermore, if A_1 and A_2 are ideally effective then $A_1 + A_2$ is.

Cartesian products behave equally nicely:

Proposition 4.18 (Cartesian Product)**.** *The ideals of $A_1 \times A_2$ are all sets of the form $I_1 \times I_2$ for $I_1 \in Idl(A_1)$ and $I_2 \in Idl(A_2)$. Thus $Idl(A_1 \times A_2) \equiv Idl(A_1) \times Idl(A_2)$.*
Furthermore, if A_1 and A_2 are ideally effective then $A_1 \times A_2$ is.

Proof Sketch. We leave the characterisation of $Idl(A_1 \times A_2, \leq_\times)$ as an exercise.

Regarding effectiveness, we assume that we are given an ideal presentation of each basic A_i set and use these procedures to implement an ideal presentation of $X \stackrel{\text{def}}{=} A_1 \times A_2$. Clearly, elements of X, and ideals of $Idl(X)$, can be represented as pairs of basic elements and as pairs of basic ideals respectively.

All the required procedures are very easy to provide. For example (CF) for X relies on

$$\neg(F_1 \times F_2) = (\neg F_1) \times A_2 \cup A_1 \times (\neg F_2)\,. \tag{4.3}$$

We now use (CF) at the basic level to replace each $\neg F_i$ in (4.3) by an ideal decomposition $\neg F_i = I_{i,1} \cup \cdots \cup I_{i,\ell_i}$, and also (XF) to replace each A_i by some ideal decomposition. Once these replacements are done, there only remains to invoke distributivity of cartesian product over unions. (We let the reader check that none of the other required procedures is more involved than this implementation for (CF).) □

With the above, one sees why $\mathbb{N} \cup \omega$ is such an ubiquitous set in the VAS and Petri Net literature: handling order-closed sets of configurations is like handling list of tuples in $(\mathbb{N} \cup \omega)^k$.

The following proposition shows two other simple instances of hierarchical constructions that accommodate our ideally effective wqos.

Proposition 4.19 (More Sums and Products). *If A_1 and A_2 are ideally effective, then the* lexicographic sum $A_1 +_{\text{lex}} A_2$ *and the* lexicographic product $A_1 \times_{\text{lex}} A_2$ *are ideally effective.*

For completeness, we should recall [2] that $A_1 +_{\text{lex}} A_2$ and $A_1 \times_{\text{lex}} A_2$ have the same support set as, respectively, the disjoint sum $A_1 + A_2$ and the cartesian product $A_1 \times A_2$, but their ordering is more permissive:

$$\langle i, a\rangle \leq_{A_1+_{\text{lex}}A_2} \langle j, b\rangle \stackrel{\text{def}}{\Leftrightarrow} i < j \text{ or } i = j \wedge a \leq_{A_i} b\,, \tag{4.4}$$

$$\langle a, b\rangle \leq_{A_1\times_{\text{lex}}A_2} \langle a', b'\rangle \stackrel{\text{def}}{\Leftrightarrow} a <_{A_1} a' \text{ or } a \equiv_{A_1} a' \wedge b \leq_{A_2} b'\,. \tag{4.5}$$

We refer to Exercises 4.6 and 4.7 for a proof of Proposition 4.19: the reader should easily work out a characterisation of the ideals of $A_1 +_{\text{lex}} A_2$ and $A_1 \times_{\text{lex}} A_2$ in terms of the basic ideals in A_1 and A_2.

4.3.3 Sequence Extensions

Let us start with some finite alphabet, say $\Sigma = \{a, b, c, d\}$ and consider the ideals of $(\Sigma^*, \leq_*)$. As usual, they include all principal ideals, of the form $\downarrow w$ for a word $w \in \Sigma^*$. We note that all these $\downarrow w$ are finite languages so these cannot be all the ideals, see Exercise 4.1.

For starters, Σ^* itself is an infinite ideal: one only has to check that this is a directed subset.

Another infinite subset is, say, $S = \{\varepsilon, ab, abab, ababab, \ldots\}$, the language denoted by the regular expression $(ab)^*$. The reader can check that S is directed but it is not an ideal (not downward-closed). However its downward-closure $\downarrow S$ obviously is. Note that $\downarrow S$ coincide with $(a + b)^*$, the set of words that can be written using only a's and b's. One can generalise this observation: for any subset $\Gamma \subseteq \Sigma$ of letters, Γ^* is an ideal of Σ^*.

Let us continue our exploration. Downward-closed sets can be obtained by taking the complement of an upward-closed set. So let us look at, say, $D = \Sigma^* \setminus \uparrow ab$, i.e., all words that do not have ab as a subword. One sees that D consists of all words with no a's, i.e., $(b + c + d)^*$, all words with no b's, i.e., $(a + c + d)^*$, and all words possibly having a's and b's but where the a's are after the b's, i.e., $(b+c+d)^*(a+c+d)^*$. Actually, that third part contains the earlier two and we can summarise with

$$D \stackrel{\text{def}}{=} \Sigma^* \setminus \uparrow ab = (b + c + d)^*(a + c + d)^*\,. \tag{4.6}$$

Now D is an ideal. One way to show this is to recall that $(b+c+d)^*$ and $(a+c+d)^*$, being some Γ^*, are ideals and apply the following lemma.

Lemma 4.20. *For any wqo $(X, \leq)$, if I, J are ideals of $(X^*, \leq_*)$, their concatenation $I \cdot J$ is an ideal too.*

[2] See also Exercise 1.4.

Proof Sketch. Let us check that $I \cdot J$ is directed. For this consider two words $u, v \in I \cdot J$. They can be factored under the form $u = u_1 u_2$ and $v = v_1 v_2$ for some $u_1, v_1 \in I$ and $u_2, v_2 \in J$. Since I is directed, it contains some w_1 with $u_1 \leq_* w_1$ and $u_2 \leq_* w_2$. Similarly J contains some w_2 above u_2 and v_2. We deduce $u \leq_* w_1 w_2 \geq_* v$. And since $w_1 w_2 \in I \cdot J$, we have proved that $I \cdot J$ is directed. One shows similarly that $I \cdot J$ is downward-closed because I and J are, and that it is nonempty since I and J are. □

Observe that Lemma 4.20 applies to $(X^*, \leq_*)$ with an arbitrary underlying wqo $(X, \leq)$, not just a finite alphabet with trivial ordering. This suggests generalising our earlier observation that Γ^* is an ideal of $(\Sigma^*, \leq_*)$:

Lemma 4.21. *For any wqo $(X, \leq)$, if $D \subseteq X$ is downward-closed then D^* is an ideal of $(X^*, \leq_*)$.*

We are now ready to characterise the set of ideals for arbitrary sequence extensions. Fix a wqo $(X, \leq)$.

atoms

Proposition 4.22 (Ideals of $(X^*, \leq_*)$)**.** *The ideals of $(X^*, \leq_*)$ are exactly the concatenations $A_1 \cdots A_n$ of* atoms*, where an atom of X^* is any language of the form $A = D^*$ for some downward-closed $D \in \mathit{Down}(X)$, or of the form $A = I + \varepsilon$ for some ideal I of X.*

Here "$I + \varepsilon$" is our denotation for the set of all sequences of length 1 that consists of a "letter" from I, together with the empty sequence ε.

Proof Sketch. We let the reader check that any $I + \varepsilon$ atom is an ideal of X^*, and we saw that the D^* atoms too are ideals (Lemma 4.21), and that all concatenations (we also say *products*) of atoms are ideals (Lemma 4.20).

To show that these products generate all of $\mathit{Idl}(X^*)$, we prove that the complements of upward-closed sets can be expressed as a finite union of products of atoms. Since upward-closed sets have a finite basis, it suffices to show that, (1) for any sequence $w = x_1 \cdots x_n$ in X^*, the complement $X^* \setminus {\uparrow} w$ is a product, and that (2) finite unions of products are closed under intersection.

For (1) we assume $|w| \geq 1$ and use

$$X^* \setminus {\uparrow_{X^*}}(x_1 \cdots x_n) = (X \setminus {\uparrow_X} x_1)^* \cdots (X \setminus {\uparrow_X} x_n)^* \,, \tag{4.7}$$

generalising the earlier example in (4.6). We note that any $(X \setminus {\uparrow} x_i)^*$ is indeed D^* for some $D \in \mathit{Down}(X)$. The special case $|w| = 0$ leads to $X^* \setminus {\uparrow_{X^*}} \varepsilon = \emptyset$.

For (2) we use induction on the length of products, distributivity of concatenation over union, and basic language-theoretical properties like

$$\begin{aligned} D^*\cdot P \cap D'^*\cdot P' &= \begin{array}{l}(D\cap D')^*\cdot[D^*\cdot P \cap P'] \\ \cup\,(D\cap D')^*\cdot[P \cap D'^*\cdot P']\,,\end{array} \\ (I+\varepsilon)\cdot P \cap (I'+\varepsilon)\cdot P' &= [(I\cap I')+\varepsilon]\cdot[P\cap P']\,, \qquad (4.8)\\ (I+\varepsilon)\cdot P \cap D'^*\cdot P' &= [(I\cap D')+\varepsilon]\cdot[P\cap D'^*\cdot P']\,. \qquad \square \end{aligned}$$

We are now ready to state and prove the main result of this section.

Theorem 4.23. *If $(X,\leq)$ is ideally effective then $(X^*,\leq_*)$ is.*

To prove that $(X^*,\leq_*)$ is ideally effective, we have to provide a series of procedures and data structures. As data structure for X^*, we shall use finite sequences of elements of X, relying on the fact that X is recursive by assumption. Deciding $\leq_*$ is easy once we know, by assumption, how to compare elements occurring in sequences.

The data structure for $\mathit{Idl}(X^*)$ is only slightly more involved. We rely on Proposition 4.22 and represent products of atoms like $A_1\cdots A_n$ by sequences, using the representation of ideals of X for atoms of the form $I+\varepsilon$, and finite unions of ideals of X for the downward-closed subsets of X that generate atoms of the form D^*.

The next step is to show that inclusion between ideals of X^* is decidable (assuming the above representation). First we see that inclusion tests *between atoms* of X^* reduce to inclusion tests between ideals and downward-closed subsets of X, thanks to the following equivalences.

$I+\varepsilon\subseteq I'+\varepsilon$	iff $I\subseteq I'$	$D^*\subseteq I+\varepsilon$	iff $D\subseteq\emptyset$
$I+\varepsilon\subseteq D^*$	iff $I\subseteq D$	$D^*\subseteq D'^*$	iff $D\subseteq D'$

Building on this, one can compare products via the following equivalences (proofs omitted), where P, P' denote arbitrary products, and where ε is the empty product.

$\varepsilon\subseteq P$	always
$(I+\varepsilon).P\subseteq\varepsilon$	never
$D^*.P\subseteq\varepsilon$	iff $D=\emptyset\wedge P\subseteq\varepsilon$
$(I+\varepsilon).P\subseteq(I'+\varepsilon).P'$	iff $\big[I\subseteq I'\wedge P\subseteq P'\big]\vee\big[(I+\varepsilon).P\subseteq P'\big]$
$(I+\varepsilon).P\subseteq D^*.P'$	iff $\big[I\subseteq D\wedge P\subseteq P'\big]\vee\big[(I+\varepsilon).P\subseteq P'\big]$
$D^*.P\subseteq(I'+\varepsilon).P'$	iff $\big[D=\emptyset\wedge P\subseteq(I+\varepsilon).P'\big]\vee\big[D^*.P\subseteq P'\big]$
$D^*.P\subseteq D'^*.P'$	iff $\big[D\subseteq D'\wedge P\subseteq D'^*.P'\big]\vee\big[D^*.P\subseteq P'\big]$

Note that the above equalities directly lead to a dynamic programming procedure that will perform at most n^2 tests between atoms when comparing two ideals that

are denoted by products of at most n atoms each.

Thus far, we have provided an effective presentation for $(X^*, \leq_*)$ and $(\mathit{Idl}(X^*), \subseteq)$. We continue with

(CF): Equation (4.7) can be turned into an algorithm for expressing the complement $X^* \smallsetminus \uparrow w$ of any principal filter $\uparrow w$ as a union of ideals. Here the assumption that $(X, \leq)$ is ideally effective is used to express $X \smallsetminus x$ as a downward-closed subset of X.

(II): Eq. (4.8) can be turned into an algorithm for computing the intersection of ideals of X^*. It only uses the primitive for the intersection of downward-closed sets in X, and simple procedures for distributing concatenations over unions.

(PI): For an arbitrary $w = x_1 \cdots x_n$, the (principal) ideal $\downarrow w$ is represented by the product $(\downarrow x_1 + \varepsilon) \cdots (\downarrow x_n + \varepsilon)$, where now a downward-closure of the form $\downarrow x_i$ returns an ideal of X. We thus rely on the procedure that witnesses (PI) for $(X, \leq)$.

(XI): Expressing X^* with ideals is easy since this is just one single D^* atom: recall that $X \in \mathit{Down}(X)$. We only need to express X itself as a finite union of filters of X, i.e., rely on (XI) for X.

The proof of Theorem 4.23 is completed since, thanks to Proposition 4.13, we have provided enough procedures to know that an ideal presentation of $(X^*, \leq_*)$ can be obtained from an ideal presentation of $(X, \leq)$.

4.4 Some Algorithmic Applications of Ideals

We present in this section two applications of wqo ideals to the Cover problem for WSTS.

4.4.1 Forward Coverability

Our first ideal-based algorithm for coverability is a *forward coverability* one: it relies on the effectiveness of *Post* computations to avoid requiring effective pred-bases. Compared to the backward algorithm of Section 1.2.2 on page 5, one advantage is that it is often easier in practice to compute successor states than predecessor states. Another interest is that, since the algorithm proceeds forward from the initial state, it is likely to prune the search space more efficiently.

Consider an instance of Cover: we are given a WSTS $\langle S, \to, \leq \rangle$ and a source state s and a target state t from S. We shall assume that $\langle S, \leq \rangle$ is effective in the sense of Definition 4.11 plus (PI), and that it is *ideally Post-effective*, meaning that

ideally *Post*-effective

Post computations as defined in (1.2) on page 5 can be carried over ideals. More precisely, if we define

$$\mathit{Post}_\exists(I) \stackrel{\text{def}}{=} \{s_2 \in S \mid \exists s_1 \in I, s_1 \to s_2\}\ , \tag{4.9}$$

then we require the canonical decomposition of $\downarrow \mathit{Post}_\exists(I)$ to be computable for all $I \in \mathit{Idl}(S)$.

Proposition 4.24. *Coverability is decidable for ideally Post-effective WSTS with (PI).*

We proceed by showing the problem to be both recursive and co-recursive and thus decidable. We thus exhibit two procedures that will be run in parallel: the first one attempts to show that s cover t, while the second attempts to prove that s does not cover t.

Proof of recursivity. Our first procedure computes a sequence of downward-closed sets

$$D_0 \stackrel{\text{def}}{=} \downarrow s \qquad\qquad D_{n+1} \stackrel{\text{def}}{=} \downarrow \mathit{Post}_\exists(D_n) \tag{4.10}$$

until $t \in D_n$. These operations are effective using (PI), (ID), and the computability of $\downarrow \mathit{Post}_\exists(D) = \downarrow \mathit{Post}_\exists(I_1) \cup \cdots \cup \downarrow \mathit{Post}_\exists(I_\ell)$ for any downward-closed $D = I_1 \cup \cdots \cup I_\ell$. Regarding correctness, if $s = s_0 \to s_1 \to \cdots \to s_n \geq t$, then for all $0 \leq i \leq n$, $s_i \in D_i$ and thus the procedure will terminate; conversely, if $t \in D_n$ for some n, then there exists $s_n \in D_n$ and $s'_{n-1} \in D_{n-1}$ with $s'_{n-1} \to s_n \geq t$, which in the same way is such that there exists $s_{n-1} \in D_{n-1}$ and $s'_{n-2} \in D_{n-2}$ such that $s'_{n-2} \to s_{n-1} \geq s'_{n-1}$, etc. We can therefore exhibit a sequence $s_0 \geq s'_0 \to s_1 \geq s'_1 \to s_2 \geq s'_2 \to \cdots \to s_n \geq t$ with $s_i \in D_i$ for all i. By compatibility, we can delay all the $\geq$ relations above until the very end and exhibit a coverability witness. □

Proof of co-recursivity. Our second procedure enumerates downward-closed sets $D \in \mathit{Down}(S)$ until we find a (forward) *inductive invariant*, i.e. D such that $\downarrow \mathit{Post}_\exists(D) \subseteq D$, $s \in D$, and $t \notin D$. All these operations are effective using (IR), (PI), (CI), and the computability of the canonical decomposition of $\downarrow \mathit{Post}_\exists(D)$.

The procedure halts if and only if s cannot cover t. Indeed, if it halts, then $\mathit{Post}_\exists(D) \subseteq D$. Defining $\mathit{Post}^*_\exists(D) \stackrel{\text{def}}{=} \{s_2 \in S \mid \exists s_1 \in D, s_1 \to^* s_2\}$, this shows that $\downarrow \mathit{Post}^*_\exists(D) \subseteq D$. Therefore $t \notin \downarrow \mathit{Post}^*_\exists(s) \subseteq D$, which is the equivalent to s not covering t. Conversely, it suffices to show that $\downarrow \mathit{Post}^*_\exists(s)$ is an inductive invariant, and thus it suffices to show that $\downarrow \mathit{Post}_\exists(\downarrow \mathit{Post}^*_\exists(s)) \subseteq \downarrow \mathit{Post}^*_\exists(s)$. Consider for this some $s'_2 \in \downarrow \mathit{Post}_\exists(\downarrow \mathit{Post}^*_\exists(s))$: there exist s_2, s_1, and s'_1 such that $s \to^* s_1 \geq s'_1 \to s_2 \geq s'_2$. By compatibility, there exists $s''_2 \geq s_2$ such that $s_1 \to s''_2$, i.e. such that $s \to^* s''_2 \geq s'_2$, which shows that s'_2 belongs to $\downarrow \mathit{Post}^*_\exists(s)$. □

Example 4.25 (VAS)**.** By propositions 4.16 and 4.18, $\mathbb{N}^d$ along with the product ordering is ideally effective, with ideal representations in $(\mathbb{N} \cup \{\omega\})^d$. In order to apply Proposition 4.24 to VASS, we merely have to show that they are ideally *Post*-effective. Observe for this that, for any $\mathbf{u}$ in $(\mathbb{N} \cup \{\omega\})^d$,

$$\downarrow \mathit{Post}_\exists(\mathbf{u}) = \{\mathbf{u} + \mathbf{a} \mid \mathbf{a} \in \mathbf{A}\} \tag{4.11}$$

where $\omega + k = \omega$ for all $k \in \mathbb{Z}$. This illustrates the fact that $\mathit{Post}_\exists(I)$ is often easy to compute "by continuity."

4.4.2 Dual Backward Coverability

Our second algorithm for Cover proceeds like the backward algorithm of Section 1.2.2 on page 5, but manipulates the *complements* $D_0 \supseteq D_1 \supseteq \cdots$ of the upward-closed sets $U_0 \subseteq U_1 \subseteq \cdots$ defined in (1.4) on page 6. In other words, given a Cover instance comprising a WSTS $\langle S, \to, \leq \rangle$ and two states s and t, the algorithm computes

$$D_0 \stackrel{\text{def}}{=} S \setminus \uparrow t \qquad\qquad D_{n+1} \stackrel{\text{def}}{=} D_n \cap \mathit{Pre}_\forall(D_n)\ , \tag{4.12}$$

where

$$\mathit{Pre}_\forall(D) \stackrel{\text{def}}{=} \{s_1 \in S \mid \forall s_2 \in S, s_1 \to s_2 \text{ implies } s_2 \in D\}. \tag{4.13}$$

In order to carry the operations in (4.12), we rely on (CF) to obtain the canonical decomposition for D_0, and on the fact that

$$\mathit{Pre}_\forall(D) = S \setminus \mathit{Pre}_\exists(S \setminus D) \tag{4.14}$$

along (CI) and (IF) with the effective pred-basis, followed by (CF) and (II) to obtain the decomposition for D_{n+1} given that of D_n. Finally, the descending sequence (4.12) must eventually stabilise to $\mathit{Pre}^*_\forall(D_0)$ over the wqo $(S, \leq)$, and s covers t if and only if $s \notin \mathit{Pre}^*_\forall(D_0)$, which can be decided by (IM). Hence

Proposition 4.26. *Cover is decidable for ideally effective WSTS with effective pred-basis.*

VAS Coverability

Proposition 4.26 does not improve over Proposition 1.17, since it has more stringent requisites. However, this dual view of the backward coverability algorithm sometimes exhibits properties that were hidden in its usual presentation. As we shall see, in the case of VAS, it allows to lower the Ackermannian upper bounds of Section 2.2.2 on page 33 down to 2ExpTime. We begin by investigating the general properties of the dual backward algorithm, before focusing on the case of VAS.

PROPER IDEALS. Consider a computation $D_0 \supsetneq D_1 \supsetneq \cdots \supsetneq D_n = D_{n+1}$ of the dual backward algorithm, where each of the D_k is represented by its canonical decomposition. By similar arguments to those of Section 2.2.2, at each refinement step $D_k \supsetneq D_{k+1}$ of the algorithm, some of the ideals from the canonical decomposition of D_k—which we call *proper*—might be removed, while others might remain untouched in the decomposition of D_{k+1}. Sequences of such proper ideals $I_0, I_1, \ldots, I_{n-1}$ are necessarily bad for inclusion.

proper ideal

Put differently, an ideal is proper in D_k if and only if it intersects the set of elements *excluded* between steps k and $k+1$: by basic set operations, first observe that (4.12) can be restated using

$$D_{k+1} = D_k \setminus \{s_1 \in D_k \mid \exists s_2 \notin D_k \,.\, s_1 \to s_2\} \,. \tag{4.15}$$

Moreover, noting $D_{-1} \stackrel{\text{def}}{=} S$, s_2 in (4.15) must belong to D_{k-1}, or s_1 would have already been excluded before step k. We have therefore $D_{k+1} = D_k \setminus E_k$ where

$$E_{-1} \stackrel{\text{def}}{=} \uparrow t \,, \qquad E_k \stackrel{\text{def}}{=} \{s_1 \in D_k \mid \exists s_2 \in E_{k-1}, s_1 \to s_2\} \,, \tag{4.16}$$

and an ideal I_k is proper in D_k if and only if $I_k \cap E_k \neq \emptyset$.

Lemma 4.27 (Proper Transition Sequences)**.** *If I_{k+1} is a proper ideal in D_{k+1}, then there exists J in the canonical decomposition of $\downarrow \mathit{Post}_\exists(I_{k+1})$ and I_k a proper ideal of D_k such that $J \subseteq I_k$.*

Proof. By the previous remark, $I_{k+1} \cap E_{k+1} \neq \emptyset$. Thus $\downarrow \mathrm{Post}_\exists(I_{k+1}) \cap E_k \neq \emptyset$ by (4.16), and there exists J from the canonical decomposition of $\downarrow \mathit{Post}_\exists(I_{k+1})$ such that $J \cap E_k \neq \emptyset$.

Since $I_{k+1} \subseteq D_{k+1} \subseteq \mathrm{Pre}_\forall(D_k)$ by (4.12), we also know that $\mathrm{Post}_\exists(I_{k+1}) \subseteq D_k$. By ideal irreducibility, it means that $J \subseteq I_k$ for some ideal I_k from the decomposition of D_k. Observe finally that $I_k \cap E_k \supseteq J \cap E_k \neq \emptyset$, i.e. that I_k is proper. □

ω-MONOTONICITY. Let us turn now to the specific case of VAS. Our instance of Cover is thus a d-dimensional VAS $\langle \mathbf{x}_0, \mathbf{A} \rangle$ and a target configuration $\mathbf{t}$ in $\mathbb{N}^d$.

As mentioned in Example 4.25, the ideals of VASS configurations can be represented through elements of $(\mathbb{N} \cup \{\omega\})^d$. For $\mathbf{u}$ in $(\mathbb{N} \cup \{\omega\})^d$, its *$\omega$-set* is the subset $\omega(\mathbf{u})$ of $\{1, \ldots, d\}$ such that $\mathbf{u}(i) = \omega$ if and only if $i \in \omega(\mathbf{u})$. We say that a descending chain $D_0 \supsetneq D_1 \supsetneq \cdots \supsetneq D_\ell$ of downward-closed subsets of $\mathbb{N}^d$ is *ω-monotone* if for all $0 \leq k < \ell - 1$ and all proper ideals $\mathbf{v}_{k+1}$ in the canonical decomposition of D_{k+1}, there exists a proper ideal $\mathbf{v}_k$ in the canonical decomposition of D_k such that $\omega(\mathbf{v}_{k+1}) \subseteq \omega(\mathbf{v}_k)$.

ω-set

ω-monotone

Lemma 4.28 (VAS Descending Chains are ω-Monotone)**.** *The descending chains computed by the dual backward coverability algorithm for VAS are ω-monotone.*

Proof. Let $D_0 \supsetneq D_1 \supsetneq \cdots \supsetneq D_\ell$ be the descending chain computed for a VASS. Suppose $0 \leq k < \ell - 1$ and $\mathbf{v}_{k+1}$ is a proper ideal in the decomposition of D_{k+1}. By Lemma 4.27 and (4.11), there exists a proper ideal $\mathbf{v}_k$ in the decomposition of D_k such that $\mathbf{v}_{k+1} + \mathbf{a} \leq_\times \mathbf{v}_k$ for some $\mathbf{a}$ in $\mathbf{A}$. We conclude that $\omega(\mathbf{v}_{k+1}) \subseteq \omega(\mathbf{v}_k)$. □

Control. As in Chapter 2, in order to derive combinatorial statements, we need to restrict our attention to *controlled* sequences of downward-closed subsets. We use as in Section 2.1 the infinity norm for ideal representations in $(\mathbb{N} \cup \{\omega\})^d$: $\|\mathbf{v}\| \stackrel{\text{def}}{=} \max_{1 \leq i \leq d | \mathbf{v}(i) < \omega} \mathbf{v}(i)$, and the maximal norm $\|D\| \stackrel{\text{def}}{=} \max \|I_j\|$ over the ideals in the canonical decomposition of downward-closed sets $D = \bigcup_j I_j$. We can observe that the sequence $D_0, D_1, \ldots$ constructed by the dual backward algorithm according to (4.12) is controlled. Let $\|\mathbf{A}\| \stackrel{\text{def}}{=} \max_{\mathbf{a} \in \mathbf{A}} \|\mathbf{a}\|$; we rely for this on **????**:

Lemma 4.29 (Control for VAS)**.** *The descending chain $D_0 \supsetneq D_1 \supsetneq \cdots$ is (g, n)-controlled for $g(x) \stackrel{\text{def}}{=} x + \|\mathbf{A}\|$ and $n \stackrel{\text{def}}{=} \|\mathbf{t}\|$.*

Proof. The fact that $\|D_0\| \leq \|\mathbf{t}\|$ follows from analysing (CF). Regarding the control function g, observe that taking unions and intersections of ideals and filters does not increase the norm. The only source of growth stems from *pb* computations: $\|D_{k+1}\| \leq \|D_k\| + \|\mathbf{A}\|$. □

Upper Bound. We are now in position to state a refinement of **??** for ω-monotone controlled descending chains. For a control function $g \colon \mathbb{N} \to \mathbb{N}$, define the function $\widetilde{g} \colon \mathbb{N}^2 \to \mathbb{N}$ by induction on its first argument:

$$\widetilde{g}(0, n) \stackrel{\text{def}}{=} 1 \,, \qquad \widetilde{g}(m+1, n) \stackrel{\text{def}}{=} \widetilde{g}(m, n) + (g^{\widetilde{g}(m,n)}(n) + 1)^{m+1} \,. \tag{4.17}$$

Theorem 4.30 (Length Function Theorem for ω-Monotone Descending Chains)**.** *Any (g, n)-controlled ω-monotone descending chain $D_0 \supsetneq D_1 \supsetneq \cdots$ of downward-closed subsets of $\mathbb{N}^d$ is of length at most $\widetilde{g}(d, n)$.*

Proof. Note that D_ℓ the last element of the chain has the distinction of not having any proper ideal, hence it suffices to bound the index k of the last set D_k with a proper ideal $\mathbf{v}_k$, and add one to get a bound on ℓ. We are going to establish by induction on $d - |I|$ that if $\mathbf{v}_k$ is a proper ideal from the canonical decomposition of D_k and its ω-set is I, then $k < \widetilde{g}(d - |I|, n)$, which by monotonicity of $\widetilde{g}$ in its first argument entails $k < \widetilde{g}(d, n)$ as desired.

For the base case, $|I| = d$ implies that $\mathbf{v}_k$ is the vector with ω's in every coordinate, which can only occur in D_0. The inductive step is handled by the following claim, when setting $k < \widetilde{g}(d - |I| - 1, n)$ by induction hypothesis for the last index with a proper ideal whose ω-set strictly includes I:

Claim 4.30.1. Let I and $k < k'$ be such that:

(i) for all $j \in \{k+1, \ldots, k'-1\}$, the decomposition of D_j does not contain a proper ideal whose ω-set strictly includes I;

(ii) the decomposition of $D_{k'}$ contains a proper ideal whose ω-set is I.

Then we have $k' - k \leq (\|D_{k+1}\| + 1)^{(d-|I|)}$.

For a proof, from assumption (ii), by applying the ω-monotonicity for $j = k'-1, k'-2, \ldots, k+1$ and due to assumption (i), there exists a proper ideal $\mathbf{v}_j$ in the decomposition of D_j and such that $\omega(\mathbf{v}_j) = I$ for all $j \in \{k+1, \ldots, k'\}$. Since they are proper, those $k' - k$ vectors are mutually distinct.

Consider any such $\mathbf{v}_j$. Since $D_{k+1} \supseteq D_j$, by ideal irreducibility there exists a vector $\mathbf{u}_j$ in the decomposition of D_{k+1} such that $\mathbf{v}_j \leq_\times \mathbf{u}_j$. We have that $\omega(\mathbf{u}_j) = I$, since otherwise $\mathbf{u}_j$ would be proper at $D_{j'}$ for some $j' \in \{k+1, \ldots, j-1\}$, which would contradict assumption (i). Hence $\|\mathbf{v}_j\| \leq \|\mathbf{u}_j\| \leq \|D_{k+1}\|$.

To conclude, note that there can be at most $(\|D_{k+1}\| + 1)^{(d-|I|)}$ mutually distinct vectors in $\mathbb{N}^d_\omega$ with I as ω-set and norm bounded by $\|D_{k+1}\|$. □

Finally, putting together Lemma 4.29 (control for VAS), Lemma 4.28 (ω-monotonicity), and Theorem 4.30 (lengths of controlled ω-monotone descending chains), we obtain that the backward coverability algorithm for VAS runs in 2ExpTime, and in pseudo-polynomial time if the dimension d is fixed.

Corollary 4.31. *For any d-dimensional VAS $\mathbf{A}$ and target vector $\mathbf{t}$, the backward coverability algorithm terminates after at most $((\|\mathbf{A}\| + 1)(\|\mathbf{t}\| + 2))^{(d+1)!}$ steps.*

Proof. Let $h(m,n) = \widetilde{g}(m,n)(\|\mathbf{A}\| + 1)(n+2)$ where $g(x) = x + \|\mathbf{A}\|$. We have $h(m+1,n) \leq (h(m,n))^{m+2}$, so $\widetilde{g}(m,n) \leq h(m,n) \leq ((\|\mathbf{A}\| + 1)(n+2))^{(m+1)!}$. □

Exercises

Exercise 4.1 (Finite ideals)**.** Let $(X, \leq)$ be a wqo and $I \subseteq X$ one of its ideals.

(1) Show that if I is finite then I is a principal ideal.

(2) Is the reciprocal true?

(3) Show that if X is infinite some of its ideals are infinite too.

Exercise 4.2 (Ideals of ordinals)**.** What are the ideals of α, an arbitrary ordinal?

Exercise 4.3 (Rado's Structure)**.** (1) What are the ideals of $(R, \leq_R)$, Rado's structure seen in Exercise 1.11 ?

(2) Is $(Idl(R), \subseteq)$ a wqo?

Exercise 4.4 (Ideals of Disjoint Sums)**.** We prove Proposition 4.17 in steps.

(1) Show that the ideals of $A_1 + A_2$ are all sets of the form $\{i\} \times I$ for $i = 1, 2$ and I an ideal of A_i.

(2) Show that $(A_1 + A_2, \leq_+)$ is effective when $(A_1, \leq_1)$ and $(A_2, \leq_2)$ are.

(3) Further show that $(A_1 + A_2, \leq_+)$ is ideally effective when $(A_1, \leq_1)$ and $(A_2, \leq_2)$ are. One may rely on Proposition 4.13.

Exercise 4.5 (Ideals of Cartesian Products)**.** (1) Prove the first part of Proposition 4.18: the ideals of $(A_1 \times A_2, \leq_\times)$ are exactly the sets of the form $I_1 \times I_2$ for $I_1 \in \mathit{Idl}(A_1, \leq_1)$ and $I_2 \in \mathit{Idl}(A_2, \leq_2)$.

(2) Complete the proof of the second part of Proposition 4.18: assume that A_1 and A_2 are ideally effective and show that $A_1 \times A_2$ has procedures witnessing (II), (PI), and (XI).

⋆ **Exercise 4.6** (Ideals of Lexicographic Sums)**.** In this exercise we prove the first half of Proposition 4.19.

(1) Show that $\mathit{Idl}(A_1 +_{\text{lex}} A_2, \subseteq)$ is isomorphic to $\mathit{Idl}(A_1, \subseteq) +_{\text{lex}} \mathit{Idl}(A_2, \subseteq)$.

(2) Show that $A_1 +_{\text{lex}} A_2$ is effective when A_1 and A_2 are.

(3) Show that $A_1 +_{\text{lex}} A_2$ is ideally effective when A_1 and A_2 are.

⋆ **Exercise 4.7** (Ideals of Lexicographic Products)**.** In this exercise we prove the second half of Proposition 4.19.

(1) What are the ideals of $A_1 \times_{\text{lex}} A_2$?

(2) Show that $A_1 \times_{\text{lex}} A_2$ is effective when A_1 and A_2 are.

(3) Show that $A_1 \times_{\text{lex}} A_2$ is ideally effective when A_1 and A_2 are.

Exercise 4.8 (More Ideally Effective WQOs)**.** In the following, one can use Proposition 4.13 to shorten the proof.

(1) Show that if α is a recursive ordinal then it is fully effective (in the sense of Definition 4.12).

(2) Show that if $(Q, \leq)$ is any *finite* qo then it is fully effective. For this question you are required to provide *uniform algorithms*, parameterized by a canonical representation of $(Q, \leq)$, e.g., where one gives $\leq$ via a Boolean $n \times n$ matrix for $n = |Q|$.

(3) Show that Rado's structure is ideally effective.

Exercise 4.9 (Ideals of $(X^*, \leq_*)$)**.** We complete the proof of Proposition 4.22.

(1) Show that if I is an ideal of $(X, \leq)$, and $D \subseteq X$ is a downward-closed subset of X, then $I + \varepsilon$ and D^* are ideals of $(X^*, \leq_*)$.

(2) Prove Eq. (4.7): for $w \in X^*$ of the form $w = x_1 \cdots x_n$, the complement $\neg\uparrow_{X^*} w$ is the product $D^*_{x_1} \cdots D^*_{x_n}$ where D_x denotes the downward-closed subset $\downarrow_< x$, i.e., $\neg\uparrow_X x$, of X.

Bibliographic Notes

This chapter heavily borrows from (Goubault-Larrecq et al., 2016).

Ideals. The structural properties of wqo ideals we recall in Section 4.1 are classic, see e.g. (Fraïssé, 1986, chapter 10). Exercise 4.3 is from (Finkel and Goubault-Larrecq, 2012, Sect. 4).

Effective Ideal Representations. The proof of Proposition 4.13 relies on the so-called *Generalised Valk-Jantzen Lemma* from (Goubault-Larrecq, 2009). The algorithms for $(\mathbb{N}^k, \leq_\times)$ extend results from (Karp and Miller, 1969) and (Valk and Jantzen, 1985). The algorithms for $(\Sigma^*, \leq_*)$ extend results from (Kabil and Pouzet, 1992) and (Abdulla et al., 2004).

Forward Coverability. Section 4.4.1 from Blondin et al. (2014) (Blondin et al., 2016)

Dual Backward Coverability. Section 4.4.2 from (Lazić and Schmitz, 2015a) (Lazić and Schmitz, 2016)

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 ord$(0) = 0$, ord$(\alpha + 1)$ is a successor ordinal, and ord(λ) 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 $s(x) = x + 1$.

[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.

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. Observe that, if $s(x) > 0$, then $\lambda_x > 0$.

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

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

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

$$P_x(\gamma+\alpha) = \gamma + P_x(\alpha)\,, \tag{A.3}$$
$$P_x(\omega^\alpha) = \omega^{P_x(\alpha)} \cdot (\omega_x - 1) + P_x(\omega^{P_x(\alpha)})\,. \tag{A.4}$$

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

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

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

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

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

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

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

$$\begin{aligned} P_x(\omega^\lambda) &\stackrel{\text{(A.2)}}{=} P_x((\omega^\lambda)_x) \stackrel{\text{(A.1)}}{=} P_x(\omega^{\lambda_x}) \stackrel{ih}{=} \omega^{P_x(\lambda_x)} \cdot (\omega_x - 1) + P_x(\omega^{P_x(\lambda_x)}) \\ &\stackrel{\text{(A.2)}}{=} \omega^{P_x(\lambda)} \cdot (\omega_x - 1) + P_x(\omega^{P_x(\lambda)})\,. \end{aligned} \qquad \square$$

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

$$\Big(\bigcup_{x\in\mathbb{N}} \prec_x\Big) = < . \tag{A.10}$$

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

$$\begin{aligned} &\alpha' \prec_x \beta + 1 \text{ implies } \alpha' \preccurlyeq_x \beta && \text{(by (A.8))} \\ &\qquad \text{implies } \gamma + \alpha' \preccurlyeq_x \gamma + \beta \overset{\text{(A.5)}}{\prec_x} \gamma + \beta + 1 \,. && \text{(by ind. hyp.)} \end{aligned}$$

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

$$\begin{aligned} &\alpha' \prec_x \lambda \text{ implies } \alpha' \preccurlyeq_x \lambda_x && \text{(by (A.8))} \\ &\qquad \text{implies } \gamma + \alpha' \preccurlyeq_x \gamma + \lambda_x \overset{\text{(A.1)}}{=} (\gamma + \lambda)_x \overset{\text{(A.5)}}{\prec_x} \gamma + \lambda \,. && \text{(by ind. hyp.)} \end{aligned}$$

(A.12)**:** For the successor case $\alpha = \beta + 1$, we go through

$$\begin{aligned} &\alpha' \prec_x \beta + 1 \text{ implies } \alpha' \preccurlyeq_x \beta && \text{(by (A.8))} \\ &\qquad \text{implies } \omega^{\alpha'} \preccurlyeq_x \omega^{\beta} = \omega^{\beta} + 0 && \text{(by ind. hyp.)} \\ &\qquad \text{implies } \omega^{\alpha'} \preccurlyeq_x \omega^{\beta} + \omega^{\beta} \cdot (\omega_x - 1) \\ &\qquad\qquad && \text{(by equations (A.6) and (A.11))} \\ &\qquad \text{implies } \omega^{\alpha'} \preccurlyeq_x \omega^{\beta} \cdot \omega_x = (\omega^{\beta+1})_x \overset{\text{(A.5)}}{\prec_x} \omega^{\beta+1} \,. \end{aligned}$$

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

$$\begin{aligned} &\alpha' \prec_x \lambda \text{ implies } \alpha' \preccurlyeq_x \lambda_x && \text{(by (A.8))} \\ &\quad\text{implies } \omega^{\alpha'} \preccurlyeq_x \omega^{\lambda_x} \overset{\text{(A.1)}}{=} (\omega^{\lambda})_x \overset{\text{(A.5)}}{\prec_x} \omega^{\lambda}\,. && \text{(by ind. hyp.)} \end{aligned}$$

□

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 $i = 1, ..., m$. 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 characterisation 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 $\mathrm{CNF}(\varepsilon_0)$ be $(\omega_x - 1)$-lean. Then:*

$$\alpha < \gamma \text{ iff } \alpha \prec_x \gamma \text{ iff } \alpha \preccurlyeq_x P_x(\gamma) \text{ iff } \alpha \leq P_x(\gamma)\ . \tag{A.13}$$

One sees $\left(\bigcup_{x\in\mathbb{N}} \prec_x\right) = <$ over terms in $\mathrm{CNF}(\varepsilon_0)$ as a result of Lemma A.5. The proof relies on the syntactic characterisation of the ordinal ordering over terms in $\mathrm{CNF}(\varepsilon_0)$ 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 $\mathrm{CNF}(\varepsilon_0)$ 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 characterisation 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:

$$h^0(x) \stackrel{\text{def}}{=} x, \qquad h^{\alpha+1}(x) \stackrel{\text{def}}{=} h^\alpha(h(x)), \qquad h^\lambda(x) \stackrel{\text{def}}{=} h^{\lambda_x}(x). \tag{A.15}$$

An example of inner iteration hierarchy is the *Hardy hierarchy* $(H^\alpha)_{\alpha\in\Omega}$ obtained from (A.15) in the special case of $h(x) = x + 1$:

$$H^0(x) \stackrel{\text{def}}{=} x, \qquad H^{\alpha+1}(x) \stackrel{\text{def}}{=} H^\alpha(x + 1), \qquad H^\lambda(x) \stackrel{\text{def}}{=} H^{\lambda_x}(x). \tag{A.16}$$

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:

$$h_0(x) \stackrel{\text{def}}{=} 0, \qquad h_{\alpha+1}(x) \stackrel{\text{def}}{=} 1 + h_\alpha(h(x)), \qquad h_\lambda(x) \stackrel{\text{def}}{=} h_{\lambda_x}(x). \tag{A.17}$$

As before, in the case where $h(x) = x + 1$ is the successor function, this yields

$$H_0(x) \stackrel{\text{def}}{=} 0, \qquad H_{\alpha+1}(x) \stackrel{\text{def}}{=} 1 + H_\alpha(x + 1), \qquad H_\lambda(x) \stackrel{\text{def}}{=} H_{\lambda_x}(x). \tag{A.18}$$

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

$$f_0(x) \stackrel{\text{def}}{=} h(x), \qquad f_{\alpha+1}(x) \stackrel{\text{def}}{=} f_\alpha^{\omega_x}(x), \qquad f_\lambda \stackrel{\text{def}}{=} f_{\lambda_x}(x), \tag{A.19}$$

while its standard version (for $h(x) = x + 1$) is defined by

$$F_0(x) \stackrel{\text{def}}{=} x + 1, \qquad F_{\alpha+1}(x) \stackrel{\text{def}}{=} F_\alpha^{\omega_x}(x), \qquad F_\lambda(x) \stackrel{\text{def}}{=} F_{\lambda_x}(x). \tag{A.20}$$

Several properties of these functions can be proved by rather simple induction arguments.

Lemma A.6. *For all $\alpha > 0$ in Ω and x in $\mathbb{N}$ with $\omega_X > 0$,*

$$h_\alpha(x) = 1 + h_{P_x(\alpha)}(h(x)) \ , \tag{A.21}$$

$$h^\alpha(x) = h^{P_x(\alpha)}(h(x)) = h^{P_x(\alpha)+1}(x) \ , \tag{A.22}$$

$$f_\alpha(x) = f^{\omega_x}_{P_x(\alpha)}(x) = f_{P_x(\alpha)+1}(x) \ . \tag{A.23}$$

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, $h_\lambda(x) = h_{\lambda_x}(x) \overset{ih}{=} 1 + h_{P_x(\lambda_x)}(h(x)) \overset{(A.2)}{=} 1 + h_{P_x(\lambda)}(h(x))$, where the ind. hyp. can applied since $0 < \lambda_x < \lambda$. □

Lemma A.7. *Let $h(x) > x$ for all x. Then for all α in Ω and x in $\mathbb{N}$ with $\omega_x > 0$,*

$$h_\alpha(x) \leq h^\alpha(x) - x \ .$$

Proof. By induction over α. For $\alpha = 0$, $h_0(x) = 0 = x - x = h^0(x) - x$. For $\alpha > 0$,

$$\begin{aligned} h_\alpha(x) &= 1 + h_{P_x(\alpha)}(h(x)) && \text{(by Lemma A.6)} \\ &\leq 1 + h^{P_x(\alpha)}(h(x)) - h(x) && \text{(by ind. hyp. since } P_x(\alpha) < \alpha) \\ &\leq h^{P_x(\alpha)}(h(x)) - x && \text{(since } h(x) > x) \\ &= h^\alpha(x) - x \ . && \text{(by (A.22))} \end{aligned}$$

□

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.8. *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.9. Some care should be taken with Lemma A.8: $\gamma+\alpha$ is not necessarily a term in CNF. See Remark A.15 on page 102 for a related discussion.

Lemma A.10. *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.8 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.11 (Cichoń and Tahhan Bittar, 1998)**.** *Let h be an expansive monotone function. Then, for all α, α' in Ω and x, y in $\mathbb{N}$,*

$$x < y \text{ implies } h_\alpha(x) \leq h_\alpha(y) \;, \tag{A.25}$$

$$\alpha' \prec_x \alpha \text{ implies } h_{\alpha'}(x) \leq h_\alpha(x) \;. \tag{A.26}$$

Proof. Let us first deal with $\alpha' = 0$ for (A.26). Then $h_0(x) = 0 \leq h_\alpha(x)$ 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), $h_{\alpha+1}(x) = 1 + h_\alpha(h(x)) \overset{ih(A.25)}{\leq} 1 + h_\alpha(h(y)) = h_{\alpha+1}(y)$ 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 $h_{\alpha'}(x) \overset{ih(A.26)}{\leq} h_\alpha(x) \overset{ih(A.25)}{\leq} h_\alpha(h(x)) \overset{(A.17)}{=} h_{\alpha+1}(x)$ where the ind. hyp. on (A.25) can be applied since $h(x) \geq x$.

For a limit λ. For (A.25), $h_\lambda(x) = h_{\lambda_x}(x) \overset{ih(A.25)}{\leq} h_{\lambda_x}(y) \overset{ih(A.26)}{\leq} h_{\lambda_y}(y) = h_\lambda(y)$ 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 $h_{\alpha'}(x) \overset{ih(A.26)}{\leq} h_{\lambda_x}(x) = h_\lambda(x)$. □

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.12 (Löb and Wainer, 1970)**.** *Let h be a function with $h(x) \geq x$. Then, for all α, α' in Ω, x, y in $\mathbb{N}$ with $\omega_x > 0$,*

$$f_\alpha(x) \geq h(x) \geq x \;. \tag{A.27}$$

$$\alpha' \prec_x \alpha \text{ implies } f_{\alpha'}(x) \leq f_\alpha(x) \;, \tag{A.28}$$

$$x < y \text{ and } h \text{ monotone imply } f_\alpha(x) \leq f_\alpha(y) \;. \tag{A.29}$$

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 $n > 0$, $f_\alpha^n(x) \geq h(x)$: for $n = 1$ it holds since $f_\alpha(x) \geq h(x)$, and for $n+1$ since $f_\alpha^{n+1}(x) = f_\alpha(f_\alpha^n(x)) \geq f_\alpha(x)$ by ind. hyp. on n. Therefore $f_{\alpha+1}(x) = f_\alpha^{\omega_x}(x) \geq x$ 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 $f_{\alpha+1}(x) = f_\alpha^{\omega_x - 1}(f_\alpha(x)) \overset{\text{(A.27)}}{\geq} f_\alpha(x) \overset{ih}{\geq} f_{\alpha'}(x)$. 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, $f_{\alpha+1}(x) = f_\alpha^{\omega_x}(x) \overset{\text{(A.27)}}{\leq} f_\alpha^{\omega_y}(x) \overset{ih}{\leq} f_\alpha^{\omega_y}(y) = f_{\alpha+1}(y)$ 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 $\omega_x = x + 1$ in the rest of the notes, but the reader might desire to compare this to bounds using e.g. $\omega_x = x$—as seen in Lemma A.13, 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.13. *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, $h_{s,\lambda}(x) = h_{s,\lambda_x}(x) \overset{ih}{\leq} h_{id,\lambda_x}(s(x)) \overset{(A.26)}{\leq} h_{id,\lambda_{s(x)}}(s(x)) = h_{id,\lambda}(s(x))$ 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.13 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.14. *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.12.

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 $x \geq x_0$,

$$\begin{aligned}
f_\alpha^0(x) &= x = F_{\gamma+\alpha}^0(x) \\
f_\alpha^{n+1}(x) &= f_\alpha(f_\alpha^n(x)) \\
&\leq f_\alpha\big(F_{\gamma+\alpha}^n(x)\big) \qquad \text{(by (A.29) on } f_\alpha \text{ and the ind. hyp. on } n) \\
&\leq F_{\gamma+\alpha}\big(F_{\gamma+\alpha}^n(x)\big) \\
&\quad \text{(since by (A.27) } F_{\gamma+\alpha}(x) \geq x \geq x_0 \text{ and by ind. hyp. on } \alpha) \\
&= F_{\gamma+\alpha}^{n+1}(x) \; .
\end{aligned}$$

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.15. Observe that the statement of Lemma A.14 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.17):

Lemma A.16. *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.16.1. 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 $F_{id,\gamma+\alpha+1}(x) = F^x_{id,\gamma+\alpha}(x) \overset{(A.27)}{\leq} F^{x+\gamma}_{id,\gamma+\alpha}(x) \leq F^{x+\gamma}_{s,\alpha}(x) = F_{s,\alpha+1}(x)$,

limit case $\lambda > \omega$: $F_{id,\gamma+\lambda}(x) = F_{id,\gamma+\lambda_x}(x) \overset{ih}{\leq} F_{s,\lambda_x}(x) \overset{(A.28)}{\leq} F_{s,\lambda x+\gamma}(x) = F_{s,\lambda}(x)$ where (A.28) can be applied since $\lambda_x \preccurlyeq_x \lambda_{x+\gamma}$ 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.13, thus for all x,

$$\begin{aligned} F_{id,\gamma+\alpha}(x) &\leq F_{s,\alpha}(x) && \text{(by the previous claim)} \\ &= H_s^{\omega^\alpha}(x) && \text{(by Lemma A.10)} \\ &\leq H_{id}^{\omega^\alpha}(s(x)) && \text{(by Lemma A.13 and (A.24))} \\ &= F_{id,\alpha}(s(x)) \,. && \text{(by Lemma A.10)} \end{aligned}$$

□

A.8 Classes of Subrecursive Functions

We finally consider how some natural classes of recursive functions can be characterised 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 $\omega_x \stackrel{\text{def}}{=} x$.

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 characterises 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.15) are alleviated when working with the $(\mathscr{F}_\alpha)_\alpha$:

Lemma A.17. *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.14 and A.16, $f_\alpha(x) \overset{(A.29)}{\leq} f_\alpha(x + x_0) \leq F_\alpha(x + x_0 + \gamma + 1)$ 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 characterising 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 is based on (Schmitz, 2016b) and 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 Grzegorczyk 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 generalise these classes of problems to the *exponential hierarchy*

$$k\text{-ExpTime} \stackrel{\text{def}}{=} \bigcup_c \text{DTime}\left(\underbrace{2^{\cdot^{\cdot^{\cdot 2^{n^c}}}}}_{k \text{ times}}\right),$$

with the nondeterministic and space-bounded variants defined accordingly. The union of the classes in this hierarchy is the class of *elementary* problems:

$$\text{Elementary} \stackrel{\text{def}}{=} \bigcup_k k\text{-ExpTime} = \bigcup_c \text{DTime}\left(\underbrace{2^{\cdot^{\cdot^{\cdot 2^{n}}}}}_{c \text{ times}}\right).$$

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

$$F_0(x) \stackrel{\text{def}}{=} x+1\,, \qquad F_{\alpha+1}(x) \stackrel{\text{def}}{=} F_\alpha^{x+1}(x) = \overbrace{F_\alpha(F_\alpha(\cdots F_\alpha}^{x+1 \text{ times}}(x)))\,,$$
$$F_\lambda(x) \stackrel{\text{def}}{=} F_{\lambda_x}(x)\,,$$

where $\lambda_x < \lambda$ is the xth element of the *fundamental sequence* for the limit ordinal λ, defined by

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

For instance,

$$F_1(x) = 2x+1\,, \qquad F_2(x) = 2^{x+1}(x+1)-1\,,$$
$$F_3(x) > 2^{\cdot^{\cdot^{\cdot^{2}}}\}_{x \text{ times}}}\,,$$

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 characterised as a class of functions computable with bounded resources

$$\mathscr{F}_\alpha = \bigcup_c \text{FDTime}\left(F_\alpha^c(n)\right), \tag{B.1}$$

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

$$\begin{aligned}
\mathscr{F}_2 &= \text{FElementary} \,, \\
\bigcup_k \mathscr{F}_k &= \text{FPrimitive-Recursive} \,, \\
\bigcup_k \mathscr{F}_{\omega^k} &= \text{FMultiply-Recursive} \,, \\
\bigcup_{\alpha<\varepsilon_0} \mathscr{F}_\alpha &= \text{FOrdinal-Recursive} \,.
\end{aligned}$$

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.

We therefore introduce *smaller* classes:

$$\mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{p \in \bigcup_{\beta<\alpha} \mathscr{F}_\beta} \text{DTime}\left(F_\alpha(p(n))\right) \,. \tag{B.2}$$

[1] See Ritchie (1963) for a characterisation 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.

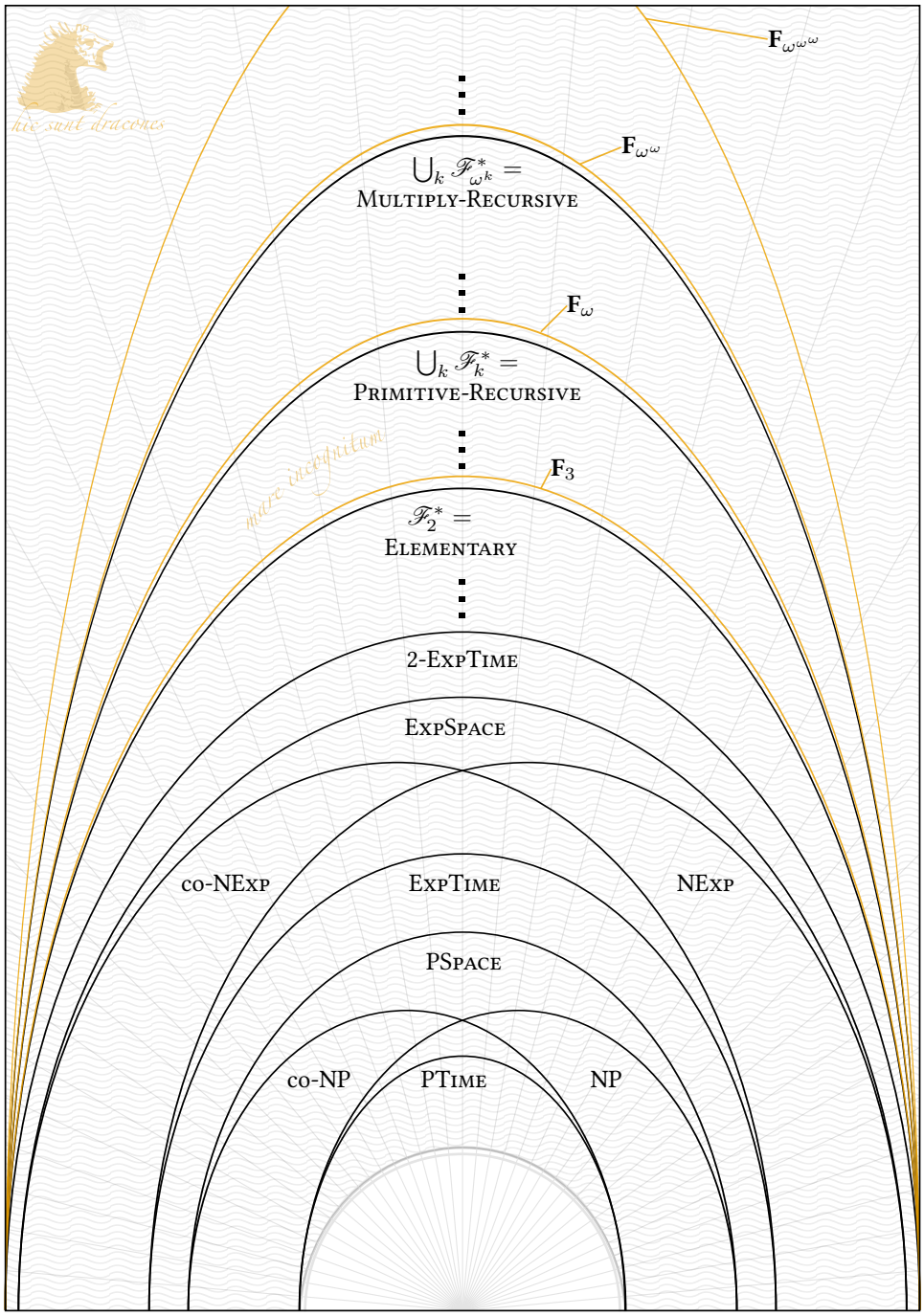

Figure B.1: Some complexity classes.

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:

$$\begin{aligned}\mathbf{F}_\omega &= \textsc{Ackermann}\ ,\\ \mathbf{F}_{\omega^\omega} &= \textsc{Hyper-Ackermann}\ .\end{aligned}$$

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 characterisations: 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

star-free expression

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

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

$$\begin{array}{ccc}[\![a]\!] \stackrel{\text{def}}{=} \{a\} & [\![\varepsilon]\!] \stackrel{\text{def}}{=} \{\varepsilon\} & [\![\emptyset]\!] \stackrel{\text{def}}{=} \emptyset\\ [\![e_1 + e_2]\!] \stackrel{\text{def}}{=} [\![e_1]\!] \cup [\![e_2]\!] & [\![e_1 e_2]\!] \stackrel{\text{def}}{=} [\![e_1]\!] \cdot [\![e_2]\!] & [\![\neg e]\!] \stackrel{\text{def}}{=} \Sigma^* \smallsetminus [\![e]\!]\ .\end{array}$$

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)

[2] An alternative class for $\alpha \geq 3$ is

$$\mathbf{F}'_\alpha \stackrel{\text{def}}{=} \bigcup_c \textsc{DTime}\left(F_\alpha(n+c)\right)\ ,$$

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 $\bigcup_{\beta<\omega^\alpha\cdot 2} \mathscr{H}^*_\beta$.

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 recognising $[\![e_1]\!]$ 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 Ackermann-Complete Problems

We gather here some decision problems that can be proven decidable in $\mathbf{F}_\omega$. The common trait of all these problems is their reliance on Dickson's Lemma over $\mathbb{N}^d$ for some d for decidability, and on the associated length function theorems (McAloon, 1984; Clote, 1986; Figueira et al., 2011; Abriola et al., 2015) for Ackermann upper bounds. The reader will find examples of $\mathbf{F}_\alpha$-complete problems for higher α in (Schmitz, 2016b).

B.2.1 Vector Addition Systems

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

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

[FCP] Finite Containment Problem

instance: Two VAS $\mathcal{V}_1$ and $\mathcal{V}_2$ known to have finite sets $\mathrm{Reach}(\mathcal{V}_1)$ and $\mathrm{Reach}(\mathcal{V}_2)$ of reachable configurations.

question: Is $\mathrm{Reach}(\mathcal{V}_1)$ included in $\mathrm{Reach}(\mathcal{V}_2)$?

lower bound: Mayr and Meyer (1981), from an F_ω-bounded version of Hilbert's Tenth Problem. A simpler reduction is given by Jančar (2001) from F_ω-MM the halting problem of F_ω-bounded Minsky machines.

upper bound: Originally McAloon (1984) and Clote (1986), or more generally using length function theorems for Dickson's Lemma (Figueira et al., 2011; Abriola et al., 2015).

comment: Testing whether the set of reachable configurations of a VAS is finite is ExpSpace-complete (Lipton, 1976; Rackoff, 1978). FCP has been generalised by Jančar (2001) to a large range of behavioural relations between two VASs. Without the finiteness condition, these questions are undecidable (Hack, 1976; Jančar, 1995, 2001).

An arguably simpler problem on vector addition systems has recently been shown to be Ackermann-complete by Hofman and Totzke (2014). A *labelled vector addition system with states* (VASS) $\mathcal{V} = \langle Q, \Sigma, d, T, q_0, \mathbf{v}_0 \rangle$ is a VAS extended with a finite set Q of control states that includes a distinguished initial state q_0. The transitions in T of such systems are furthermore labelled with symbols from a finite alphabet Σ: transitions are then defined as quadruples $q \xrightarrow{a,\mathbf{u}} q'$ for a in Σ and $\mathbf{u}$ in $\mathbb{Z}^d$. Such a system defines an infinite labelled transition system $\langle Q \times \mathbb{N}^d, \to, (q_0, \mathbf{v}_0) \rangle$ where $(q, \mathbf{v}) \xrightarrow{a} (q', \mathbf{v} + \mathbf{u})$ if $q \xrightarrow{a,\mathbf{u}} q'$ is in T and $\mathbf{v} + \mathbf{u} \geq \mathbf{0}$. The *set of traces* of $\mathcal{V}$ is the set of finite sequences $L(\mathcal{V}) \stackrel{\text{def}}{=} \{a_1 \cdots a_n \in \Sigma^* \mid \exists (q, \mathbf{v}) \in Q \times \mathbb{N}^d . (q_0, \mathbf{v}_0) \xrightarrow{a_1 \cdots a_n} (q, \mathbf{v})\}$.

[1VASSU] One-Dimensional VASS Universality
instance: A one-dimensional labelled VASS $\mathcal{V} = \langle Q, \Sigma, 1, T, q_0, \mathbf{x}_0 \rangle$.
question: Does $L(\mathcal{V}) = \Sigma^*$, i.e. is every finite sequence over Σ a trace of $\mathcal{V}$?
lower bound: Hofman and Totzke (2014) by reduction from reachability in gainy counter machines, see LCM.
upper bound: Hofman and Totzke (2014) using length function theorems for Dickson's Lemma.
comment: One-dimensional VASS are also called "one counter nets" in the literature. More generally, the *inclusion* problem $L \subseteq L(\mathcal{V})$ for some rational language L is still Ackermann-complete.

B.2.2 Energy Games

A problem with a similar flavour to 1VASSU considers instead games played on (multi-)weighted graphs. Given a finite directed graph $G = (V, E)$, whose vertices $V = V_1 \uplus V_2$ are partitioned into Player 1 vertices and Player 2 vertices, and whose edges E are labelled by vectors in $\mathbb{Z}^d$ for some dimension d—representing discrete energy consumption and replenishment from d sources—, an initial vertex v of G, and an initial credit $\mathbf{u}$ in $\mathbb{N}^d$, we may consider an *energy objective* for Player 1: does she have a strategy ensuring an infinite play starting from v such that, at all times, $\mathbf{u}$ plus the vector labels of all the edges used so far is componentwise non-negative? (Note that the input to this problem could equivalently be seen as a labelled VASS with a partition of its control states into Player 1 and Player 2 states.) This problem is known to be 2ExpTime-complete (Jurdziński et al., 2015).

We are interested here in a variant where additionally Player 1 has only *partial observation*, meaning that she does not know the exact current vertex, but is only

given an equivalence class of vertices that contains it. This variant of the game is still decidable (Degorre et al., 2010), and was shown $\mathbf{F}_\omega$-complete by Pérez (2016):

[POE] Energy Games with Partial Observation
instance: A multi-weighted finite game graph $G = (V, E)$ with labelling $\lambda\colon E \to \mathbb{Z}^d$, $v \in V$, $\mathbf{u} \in \mathbb{N}^d$, and $\equiv \subseteq V \times V$ an equivalence relation on vertices.
question: Does Player 1 have a winning strategy for the energy objective and compatible with $\equiv$, when starting from v with initial credit $\mathbf{u}$?
lower bound: Pérez (2016) by reduction from reachability in gainy counter machines, see LCM.
upper bound: Pérez (2016) by applying length function theorems for Dickson's Lemma to the decision algorithm of Degorre et al. (2010).
comment: Hardness already holds in dimension $d = 1$ and in the case of a *blind* game, i.e. where $v \equiv v'$ for all $v, v' \in V$.

B.2.3 Unreliable 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 chapters 2 and 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?
lower bound: Schnoebelen (2010a), by a direct reduction from F_ω-bounded Minsky machines. The first proofs were given independently by Urquhart in 1999 and Schnoebelen in 2002.
upper bound: Length function theorems for Dickson's Lemma.
comment: Completeness also holds for terminating LCMs (meaning that every computation starting from the initial configuration terminates), coverability in Reset or Transfer Petri nets, and for reachability in *gainy* counter machines, where counter values can increase nondeterministically.

B.2.4 Relevance Logics

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

[CRI] Conjunctive Relevant Implication
instance: A formula A of $\mathbf{R}_{\to,\wedge}$.
question: Is A a theorem of $\mathbf{R}_{\to,\wedge}$?

lower bound: Urquhart (1999), from a variant of LCM: the emptiness problem of *alternating expansive counter machines*, for which he proved $\mathbf{F}_\omega$-hardness directly from F_ω-MM the halting problem in F_ω-bounded Minsky machines.
upper bound: Urquhart (1999) using length function theorem for Dickson's Lemma.
comment: Hardness also holds for any intermediate logic between $\mathbf{R}_{\to,\wedge}$ and $\mathbf{T}_{\to,\wedge}$, which might include some undecidable fragments. The related *contractive propositional linear logic* LLC and its additive-multiplicative fragment MALLC are also Ackermann-complete (Lazić and Schmitz, 2015b).

B.2.5 Data Logics & Register Automata

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

[A1RA] Emptiness of Alternating 1-Register Automata
instance: An A1RA $\mathcal{A}$.
question: Is the data language $L(\mathcal{A})$ empty?
lower bound: Demri and Lazić (2009), from reachability in gainy counter machines LCM.
upper bound: Demri and Lazić (2009), by reducing to reachability in gainy counter machines LCM.
comment: There exist many variants of the A1RA model, and hardness also holds for the corresponding data logics (e.g. Jurdziński and Lazić, 2007; Demri and Lazić, 2009; Figueira and Segoufin, 2009; Tan, 2010; Figueira, 2012; Tzevelekos and Grigore, 2013). The complexity rises to $\mathbf{F}_{\omega^\omega}$ in the case of linearly ordered data (Ouaknine and Worrell, 2007), and even to $\mathbf{F}_{\varepsilon_0}$ for data logics using multiple attributes with a hierarchical policy (Decker and Thoma, 2016).

B.2.6 Metric Temporal Logic

Metric Temporal Logic (MTL) allows 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 (Koymans, 1990). When considering infinite timed words, one usually focuses on *non-Zeno* words, where the timestamps are increasing and unbounded. MTL is an extension of linear temporal logic where temporal modalities are decorated with real intervals constraining satisfaction; for instance, a timed word w satisfies the formula $\mathsf{F}_{[3,\infty)}\varphi$ at position i, written $w, i \models \mathsf{F}_{[3,\infty)}\varphi$, only if φ holds at some

timed words

position $j > i$ of w with timestamp $\tau_j - \tau_i \geq 3$. The *safety* fragment of MTL restricts the intervals decorating "until" modalities to be right-bounded.

[SMTL] Satisfiability of Safety Metric Temporal Logic
instance: A safety MTL formula φ.
question: Does there exist an infinite non-Zeno timed word w s.t. $w, 0 \models \varphi$?
lower bound: Lazić et al. (2016), by a direct reduction from F_ω-bounded Turing machines.
upper bound: Lazić et al. (2016) by resorting to length function theorems for Dickson's Lemma.
comment: The complexity bounds are established through reductions to and from the *fair termination* problem for insertion channel systems, which Lazić et al. (2016) show to be Ackermann-complete.

B.2.7 Ground Term Rewriting

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

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

[SGTRS] State Reachability in Senescent Ground Term Rewrite Systems
instance: A senescent sGTRS $\langle Q, \Sigma, R\rangle$ with lifespan k, two states q_0 and q_f in Q, and an initial tree t_0 in $T(\Sigma)$.
question: Does there exist a tree t in $T(\Sigma)$ such that (q_f, t) is reachable from (q_0, t_0)?
lower bound: Hague (2014), from coverability in reset Petri nets, see LCM.
upper bound: Hague (2014), by reducing to coverability in reset Petri nets, see LCM.

B.2.8 Interval Temporal Logics

Interval Temporal Logics provide a formal framework for reasoning about temporal intervals. Halpern and Shoham (1991) define a logic with modalities expressing

the basic relationships that can hold between two temporal intervals, $\langle B \rangle$ for "begun by", $\langle E \rangle$ for "ended by", and their inverses $\langle \bar{B} \rangle$ and $\langle \bar{E} \rangle$. This logic, and even small fragments of it, has an undecidable satisfiability problem, thus prompting the search for decidable restrictions and variants. Montanari et al. (2010) show that the logic with relations $A\bar{A}B\bar{B}$—where $\langle A \rangle$ expresses that the two intervals "meet", i.e. share an endpoint—, has an $\mathbf{F}_\omega$-complete satisfiability problem over finite linear orders:

[ITL] Finite Linear Satisfiability of $A\bar{A}B\bar{B}$
instance: An $A\bar{A}B\bar{B}$ formula φ.
question: Does there exist an interval structure $\mathcal{S}$ over some finite linear order and an interval I of $\mathcal{S}$ s.t. $\mathcal{S}, I \models \varphi$?
lower bound: Montanari et al. (2010), from reachability in lossy counter machines (LCM).
upper bound: Montanari et al. (2010), by reducing to reachability in lossy counter machines (LCM).
comment: Hardness already holds for the fragments $\bar{A}B$ and $\bar{A}\bar{B}$ (Bresolin et al., 2012).

REFERENCES

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 23.

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 23.

Abdulla, P.A., Collomb-Annichini, A., Bouajjani, A., and Jonsson, B., 2004. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65. doi:10.1023/B:FORM.0000033962.51898.1a. Cited on page 89.

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 23.

Abriola, S., Figueira, S., and Senno, G., 2015. Linearizing well-quasi orders and bounding the length of bad sequences. *Theoretical Computer Science*, 603:3–22. doi:10.1016/j.tcs.2015.07.012. Cited on pages 51, 112.

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 68.

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 68.

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 109.

Bertrand, N. and Schnoebelen, Ph., 2013. Computable fixpoints in well-structured symbolic model checking. *Formal Methods in System Design*, 43(2):233–267. doi:10.1007/s10703-012-0168-y. Cited on page 23.

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 23.

Blockelet, M. and Schmitz, S., 2011. Model-checking coverability graphs of vector addition systems. In *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 24, 112.

Blondin, M., Finkel, A., and McKenzie, P., 2014. Handling infinitely branching WSTS. In *ICALP 2014*, volume 8573 of *Lecture Notes in Computer Science*, pages 13–25. doi:10.1007/978-3-662-43951-7_2. Cited on pages iv, 89.

Blondin, M., Finkel, A., and McKenzie, P., 2016. Well behaved transition systems. Preprint. http://arxiv.org/abs/1608.02636. Cited on page 89.

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 68.

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*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 199–204. IOS Press. doi:10.3233/978-1-61499-098-7-199.

Cited on pages 68, 117.

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 24.

Chambart, P. and Schnoebelen, Ph., 2008. The ordinal recursive complexity of lossy channel systems. In *LICS 2008*, pages 205–216. IEEE. doi:10.1109/LICS.2008.47. Cited on page 68.

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 111.

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 107.

Ciardo, G., 1994. Petri nets with marking-dependent arc cardinality: Properties and analysis. In *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 68.

Cichoń, E.A. and Wainer, S.S., 1983. The slow-growing and the Grzecorczyk hierarchies. *Journal of Symbolic Logic*, 48(2):399–408. doi:10.2307/2273557. Cited on pages 91, 97, 101.

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 51, 91, 93, 97, 99.

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 109.

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 51, 112.

Cobham, A., 1965. The intrinsic computational difficulty of functions. In *International Congress for Logic, Methodology and Philosophy of Science*, volume 2, pages 24–30. North-Holland. Cited on page 109.

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 23.

Dauchet, M. and Tison, S., 1990. The theory of ground rewrite systems is decidable. In *LICS '90*, pages 242–248. IEEE. doi:10.1109/LICS.1990.113750. Cited on page 116.

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 51.

Decker, N. and Thoma, D., 2016. On freeze LTL with ordered attributes. In Jacobs, B. and Löding, C., editors, *FoSSaCS 2016*, volume 9634 of *Lecture Notes in Computer Science*, pages 269–284. Springer. doi:10.1007/978-3-662-49630-5_16. Cited on page 115.

Degorre, A., Doyen, L., Gentilini, R., Raskin, J.F., and Toruńczyk, S., 2010. Energy and mean-payoff games with imperfect information. In *CSL 2010*, volume 6247 of *Lecture Notes in Computer Science*, pages 260–274. doi:10.1007/978-3-642-15205-4_22. Cited on page 114.

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 68.

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 68, 115.

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 92, 93.

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 23.

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 68.

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 23, 114.

Erdös, P., Lehman, R.S., Hedlund, G.A., and Buck, R.C., 1950. Solution to problem 4330. *The American Mathematical Monthly*, 57(7):493–494. doi:10.2307/2308318. Cited on page 15.

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 92, 93.

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 51, 91, 111.

Figueira, D. and Segoufin, L., 2009. Future-looking logics on data words and trees. In *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 68, 115.

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, 51, 112.

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 page 115.

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 23.

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 23.

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 pages iii, 23.

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 23.

Finkel, A. and Goubault-Larrecq, J., 2012. Forward analysis for WSTS, part II: Complete WSTS. *Logical Methods in Computer Science*, 8(4:28). doi:10.2168/LMCS-8(3:28)2012. Cited on pages 23, 24, 89.

Fraïssé, R., 1986. *Theory of Relations*, volume 118 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science. ISBN 978-0-444-87865-6. Cited on page 89.

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 107.

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 51.

Goubault-Larrecq, J., 2009. On a generalization of a result by Valk and Jantzen. Research Report LSV-09-09, Laboratoire Spécification et Vérification, ENS Cachan, France. http://www.lsv.ens-cachan.fr/Publis/publis.php?onlykey=LSV:09:09. Cited on page 89.

Goubault-Larrecq, J., Halfon, S., Karandikar, P., Narayan Kumar, K., and Ph. Schnoebelen, 2016. The ideal approach to computing closed subsets in well-quasi-orderings. In preparation. Cited on pages iv, 89.

Grzegorczyk, A., 1953. Some classes of recursive functions. *Rozprawy Matematyczne*, 4. http://matwbn.icm.edu.pl/ksiazki/rm/rm04/rm0401.pdf. Cited on page 51.

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 pages 112, 113.

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, 68.

Hague, M., 2014. Senescent ground tree rewriting systems. In *CSL-LICS 2014*, pages 48:1–48:10. ACM Press. doi:10.1145/2603088.2603112. Cited on page 116.

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 116.

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 23.

Hofman, P. and Totzke, P., 2014. Trace inclusion for one-counter nets revisited. In Ouaknine, J., Potapov, I., and Worrell, J., editors, *RP 2014*, volume 8762 of *Lecture Notes in Computer Science*, pages 151–162. Springer. doi:10.1007/978-3-319-11439-2_12. Cited on page 113.

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 51.

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 page 23.

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 pages 112, 113.

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 24, 112, 113.

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 68, 115.

Jurdziński, M., Lazić, R., and Schmitz, S., 2015. Fixed-dimensional energy games are in pseudo-polynomial time. In *ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 260–272. Springer. doi:10.1007/978-3-662-47666-6_21. Cited on page 113.

Kabil, M. and Pouzet, M., 1992. Une extension d'un théorème de P. Jullien sur les âges de mots. *RAIRO Theoretical Informatics and Applications*, 26(5):449–482. http://archive.numdam.org/article/ITA_1992__26_5_449_0.pdf. Cited on page 89.

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 pages 24, 89.

Ketonen, J. and Solovay, R., 1981. Rapidly growing Ramsey functions. *Annals of Mathematics*, 113 (2):27–314. doi:10.2307/2006985. Cited on page 51.

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 112.

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 115.

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 23.

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, 23.

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 112.

Lazić, R. and Schmitz, S., 2015a. The ideal view on Rackoff's coverability technique. In *RP 2015*, volume 9328 of *Lecture Notes in Computer Science*, pages 1–13. Springer. doi:10.1007/978-3-319-24537-9_8. Cited on pages iv, 89.

Lazić, R. and Schmitz, S., 2015b. Non-elementary complexities for branching VASS, MELL, and extensions. *ACM Transactions on Computational Logic*, 16(3:20):1–30. doi:10.1145/2733375. Cited on page 115.

Lazić, R. and Schmitz, S., 2016. The complexity of coverability in ν-Petri nets. In *LICS 2016*. ACM.

hal.inria.fr:hal-01265302. To appear. Cited on page 89.

Lazić, R., Ouaknine, J.O., and Worrell, J.B., 2016. Zeno, Hercules and the Hydra: Safety metric temporal logic is Ackermann-complete. *ACM Transactions on Computational Logic*, 17(3). doi:10.1145/2874774. Cited on page 116.

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 112.

Leroux, J. and Schmitz, S., 2015. Demystifying reachability in vector addition systems. In *LICS 2015*, pages 56–67. IEEE. doi:10.1109/LICS.2015.16. Cited on page 112.

Lipton, R.J., 1976. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University. http://www.cs.yale.edu/publications/techreports/tr63.pdf. Cited on pages 112, 113.

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 51, 99, 103, 104, 108.

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 23.

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 23.

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 112.

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 24, 112.

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 68.

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 51, 112.

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 111.

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*, volume 147 of *NATO ASI Series*, pages 487–502. D. Reidel Publishing. doi:10.1007/978-94-009-5315-4_14. Cited on page 23.

Montanari, A., Puppis, G., and Sala, P., 2010. Maximal decidable fragments of Halpern and Shoham's modal logic of intervals. In *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 117.

Nash-Williams, C.St.J.A., 1963. On well-quasi-ordering finite trees. *Math. Proc. Cambridge Philos. Soc.*, 59(4):833–835. doi:10.1017/S0305004100003844. Cited on page 23.

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 51, 91.

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 page 115.

Padovani, V., 2013. Ticket Entailment is decidable. *Mathematical Structures in Computer Science*, 23 (3):568–607. doi:10.1017/S0960129512000412. Cited on page 23.

Pérez, G.A., 2016. The fixed initial credit problem for energy games with partial-observation is ACK-complete. Preprint. http://arxiv.org/abs/1512.04255. Cited on page 114.

Podelski, A. and Rybalchenko, A., 2004. Transition invariants. In *LICS 2004*, pages 32–41. IEEE. doi:10.1109/LICS.2004.1319598. Cited on page 23.

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 24, 112, 113.

Rado, R., 1954. Partial well-ordering of sets of vectors. *Mathematika*, 1(2):89–95. doi:10.1112/S0025579300000565. Cited on pages 16, 23.

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 109.

Robertson, N. and Seymour, P.D., 2010. Graph minors. XXIII. Nash-Williams' immersion conjecture. *Journal of Combinatorial Theory, Series B*, 100(2):181–205. doi:10.1016/j.jctb.2009.07.003. Cited on page 4.

Rose, H.E., 1984. *Subrecursion: Functions and Hierarchies*, volume 9 of *Oxford Logic Guides*. Clarendon Press. Cited on pages 51, 91.

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, 51.

Schmitz, S., 2014. Complexity bounds for ordinal-based termination. In *RP 2014*, volume 8762 of *Lecture Notes in Computer Science*, pages 1–19. Springer. doi:10.1007/978-3-319-11439-2_1. Cited on page 51.

Schmitz, S., 2016a. Implicational relevance logic is 2-ExpTime-complete. *Journal of Symbolic Logic*, 81(2):641–661. doi:10.1017/jsl.2015.7. Cited on page 23.

Schmitz, S., 2016b. Complexity hierarchies beyond Elementary. *ACM Transactions on Computation Theory*, 8(1):1–36. doi:10.1145/2858784. Cited on pages iv, 107, 112.

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 68, 114.

Schnoebelen, Ph., 2010a. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In *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, iv, 68, 114.

Schnoebelen, Ph., 2010b. Lossy counter machines decidability cheat sheet. In *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 68.

Seidl, H., 1990. Deciding equivalence of finite tree automata. *SIAM Journal on Computing*, 19(3):424–437. doi:10.1137/0219027. Cited on page 107.

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 page 111.

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 68, 115.

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 23.

Tzevelekos, N. and Grigore, R., 2013. History-register automata. In Pfenning, F., editor, *FoSSaCS 2013*, volume 7794 of *Lecture Notes in Computer Science*, pages 273–288. doi:10.1007/978-3-642-37075-5_2. Cited on page 115.

Urquhart, A., 1984. The undecidability of entailment and relevant implication. *Journal of Symbolic Logic*, 49(4):1059–1073. doi:10.2307/2274261. Cited on pages 23, 114.

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 23, 68, 114, 115.

Valk, R. and Jantzen, M., 1985. The residue of vector sets with applications to decidability problems in Petri nets. *Acta Informatica*, 21(6):643–674. doi:10.1007/BF00289715. Cited on page 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 109, 111.

Wainer, S.S., 1972. Ordinal recursion, and a refinement of the extended Grzegorczyk hierarchy.

Journal of Symbolic Logic, 37(2):281–292. doi:10.2307/2272973. Cited on page 111.

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 page 51.

Wolk, E., 1967. Partially well ordered sets and partial ordinals. *Fundamenta Mathematicae*, 60(2): 175–186. http://eudml.org/doc/213955. Cited on page 16.

Zetzsche, G., 2015. An approach to computing downward closures. In *ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 440–451. Springer. doi:10.1007/978-3-662-47666-6_35.

INDEX