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 ([Schneebelen, 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 ........................................... 1
   1.1.1 Alternative Definitions .................................... 1
   1.1.2 Upward-closed Subsets of wqos ............................ 2
   1.1.3 Constructing wqos ........................................... 3

1.2 Well-Structured Transition Systems ............................ 4
   1.2.1 Termination .................................................. 5
   1.2.2 Coverability ................................................ 5

1.3 Examples of Applications ...................................... 7
   1.3.1 Program Termination ....................................... 7
   1.3.2 Relevance Logic ............................................ 9
   1.3.3 Karp & Miller Trees ....................................... 12

Exercises ...................................................................... 14

Bibliographic Notes ................................................... 23

## 2 Complexity Upper Bounds

2.1 The Length of Controlled Bad Sequences ....................... 27
   2.1.1 Controlled Sequences ....................................... 27
   2.1.2 Polynomial wqos ............................................. 28
   2.1.3 Subrecursive Functions .................................... 30
   2.1.4 Upper Bounds for Dickson’s Lemma ......................... 31

2.2 Applications ......................................................... 32
   2.2.1 Termination Algorithm ..................................... 32
   2.2.2 Coverability Algorithm ..................................... 33

2.3 Bounding the Length Function ................................... 33
   2.3.1 Residual wqos and a Descent Equation ..................... 34
   2.3.2 Reflecting wqos ............................................. 36
   2.3.3 A Bounding Function ....................................... 38

2.4 Classification in the Grzegorczyk Hierarchy ..................... 40
   2.4.1 Maximal Order Types ....................................... 40
   2.4.2 The Cichoń Hierarchy ....................................... 42
   2.4.3 Monotonicity ................................................ 44
   2.4.4 Wrapping Up ................................................ 46

Exercises ..................................................................... 47

Bibliographic Notes ................................................... 51
## Problems of Enormous Complexity  

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>Fast-Growing Complexities</td>
<td>107</td>
</tr>
<tr>
<td>B.2</td>
<td>ACKERMANN-Complete Problems</td>
<td>112</td>
</tr>
<tr>
<td>B.2.1</td>
<td>Vector Addition Systems</td>
<td>112</td>
</tr>
<tr>
<td>B.2.2</td>
<td>Energy Games</td>
<td>113</td>
</tr>
<tr>
<td>B.2.3</td>
<td>Unreliable Counter Machines</td>
<td>114</td>
</tr>
<tr>
<td>B.2.4</td>
<td>Relevance Logics</td>
<td>114</td>
</tr>
<tr>
<td>B.2.5</td>
<td>Data Logics &amp; Register Automata</td>
<td>115</td>
</tr>
<tr>
<td>B.2.6</td>
<td>Metric Temporal Logic</td>
<td>115</td>
</tr>
<tr>
<td>B.2.7</td>
<td>Ground Term Rewriting</td>
<td>116</td>
</tr>
<tr>
<td>B.2.8</td>
<td>Interval Temporal Logics</td>
<td>116</td>
</tr>
</tbody>
</table>

### References  

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>119</td>
</tr>
</tbody>
</table>

### Index  

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
</tr>
</tbody>
</table>
1

BASICS OF WQOS AND APPLICATIONS

1.1 Well Quasi Orderings

A relation $\leq$ over a set $A$ is a quasi ordering (qo) iff it is reflexive and transitive. A quasi-ordering is a partial ordering (po) iff it also antisymmetric ($x \leq y$ and $y \leq x$ imply $x = y$). Any qo induces an equivalence relation $\equiv \overset{\text{def}}{=} \leq \cap \geq$, and gives rise to a canonical partial ordering between the equivalence classes, and to a strict ordering $< \overset{\text{def}}{=} \leq \setminus \geq \setminus$ 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:

**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. $x_i \leq x_j$.

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.).

**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. $x_{i_1} \leq x_{i_1} \leq x_{i_2} \leq \cdots$. 

1.2 Well-Structured Transition Systems

1.3 Examples of Applications
**Definition 1.4 (wqo.3)** A qo \((A, \leq)\) is a wqo iff

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

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 \(\Sigma^*\) 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**. 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).

### 1.1.2 Upward-closed Subsets of wqos

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

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

Equivalently, a qo \((A, \leq)\) is a wqo iff \((\text{Down}(A), \subseteq)\), 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 1.3 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.

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.

**Example 1.10 (Subword ordering).** We use $\varepsilon$ to denote the empty sequence. Over $(\Sigma, =)$, where $\Sigma = \{a, b, c\}$ is a 3-letter alphabet and where different letters are incomparable, the word $abb$ is a subword of $cabcab$, as witnessed by the underlined letters, and written $abb \leq_* cabcab$. On the other hand $bba \nleq_* cabcab$. 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 $4 \cdot 1 \cdot 3 \nleq_* 1 \cdot 5 \cdot 0 \cdot 3 \cdot 0 \cdot 0$. Over $(\mathbb{N}^2, \leq_\triangle)$, one checks that $\left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right) \cdot \left(\begin{smallmatrix} 2 \\ 0 \end{smallmatrix}\right) \nleq_* \left(\begin{smallmatrix} 2 \\ 0 \end{smallmatrix}\right) \cdot \left(\begin{smallmatrix} 0 \\ 2 \end{smallmatrix}\right) \cdot \left(\begin{smallmatrix} 2 \\ 2 \end{smallmatrix}\right) \cdot \left(\begin{smallmatrix} 2 \\ 0 \end{smallmatrix}\right) \cdot \left(\begin{smallmatrix} 0 \\ 4 \end{smallmatrix}\right)$.
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:

**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):

**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

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

$$s \rightarrow s' \land s \leq t \text{ implies } \exists t' \geq s', t \rightarrow t'.$$

Put together, this defines a WSTS $S = (S, \rightarrow, \leq)$. In other words, the states of $S$ 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.

**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 $V = (Q, \delta, q_0, x_0)$ 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 $x_0$ in $\mathbb{N}^d$ describes the initial counter contents.

The semantics of a VASS define a transition system $(Q \times \mathbb{N}^d, \rightarrow)$ where a transition $\rightarrow$ holds between two configurations $(q, x)$ and $(q', x')$ if and only if there exists a translation $(q, a, q')$ in $\delta$ with $x' = x + a$; note that this transition requires $x + a$ 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 $(q, x) \leq (q', x')$ iff $q = q'$ and $x(j) = x'(j)$ for all $j = 1, \ldots, d$. Indeed, whenever $(q, x) \rightarrow (q', x')$ and $x \leq y$, then there exists $(q, a, q')$ in $\delta$ s.t. $x' = x + a$, and $y' = y + a \geq x + a \geq 0$, thus $(q, y) \rightarrow (q', y')$.  

1.2. Well-Structured Transition Systems

1.2.1 Termination

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

**[Term]** Termination

*instance*: A transition system \( \langle S, \rightarrow \rangle \) and a state \( s_0 \) in \( S \).

*question*: Does \( \langle S, \rightarrow \rangle \) terminate from \( s_0 \)?

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

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

**Proof.** The direct implication follows from \( \text{wqo.1} \) applied to the infinite run \( s_0 \rightarrow s_1 \rightarrow \cdots \). 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. \( s_j \rightarrow s_{j+1} \), and so on and so forth.

There is therefore a simple procedure to decide **Term**, pending some effectiveness conditions: in a transition system \( \langle S, \rightarrow \rangle \), define the (existential) successor set

\[
Post(s) \overset{\text{def}}{=} \{ s' \in S \mid s \rightarrow s' \}
\]

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

**Proposition 1.15** (Decidability of Termination for WSTSs). Let \( S = \langle S, \rightarrow, \leq \rangle \) be a WSTS and \( s_0 \) be a state in \( S \). If \( S \) is image-finite, Post-effective, and \( \leq \) is decidable, then termination of \( S \) 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 \( S \) terminates from \( s_0 \), then every branch of this tree will be finite, and since \( S \) is image-finite this tree is also finitely branching, hence finite overall by **Konig’s Lemma**. The second one attempts to prove non-termination, and looks nondeterministically for a finite witness matching **Lemma 1.14**.

1.2.2 Coverability

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

**[Cover]** Coverability

*instance*: A transition system \( \langle S, \rightarrow \rangle \), a qo \( S, \leq \), and two states \( s, t \) in \( S \).

*question*: Is \( t \) coverable from \( s \), i.e. is there a run \( s = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n \geq t \)?
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 $x \leq_A x'$.

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. $s' \rightarrow^* t' \geq t$. For a set of states $U \subseteq S$, define its (existential) predecessor set

$$\text{Pre}_\exists(U) \overset{\text{def}}{=} \{ s \in S \mid \exists s' \in U, s \rightarrow s' \} .$$

(1.3)

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

$$U = U_0 \subseteq U_1 \subseteq \cdots \text{ where } U_{n+1} \overset{\text{def}}{=} U_n \cup \text{Pre}_\exists(U_n) .$$

(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 $\text{Pre}_\exists(U)$ is upward-closed.

**Proof.** Assume $s \in \text{Pre}_\exists(U)$. Then $s \rightarrow t$ for some $t \in U$. By compatibility of $S$, if $s' \geq s$, then $s' \rightarrow t'$ for some $t' \geq t$. Thus $t' \in U$ and $s' \in \text{Pre}_\exists(U)$.

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

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

**Proof.** Compute a finite basis $B$ for $\text{Pre}_\exists^*(\{t\})$ 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.

---

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

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

\[
\langle a_0, b_0, c_0 \rangle, \ldots, \langle a_j, b_j, c_j \rangle, \ldots
\]  

(1.5)

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

\[
\langle a_0, b_0 \rangle, \ldots, \langle a_j, b_j \rangle, \ldots
\]  

(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:

\[
\langle a, b, c \rangle R \langle a', b', c' \rangle \text{ iff } a > 0 \land b > 0 \land ((a' = a - 1 \land b' = b \land c' = 2c) \lor (a' = 2c \land b' = b - 1 \land c' = 1)).
\] (1.7)

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

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

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

This is roughly what we did in (1.6), by projecting away the third component and using \( \mathbb{N}^2 \) as codomain; this does not satisfy (1.8) for the product ordering \( (\mathbb{N}^2, \leq) \), but it does satisfy it for the lexicographic ordering \( (\mathbb{N}^2, \leq_{\text{lex}}) \). Equivalently, one could define \( \rho: \mathbb{Z}^3 \to \omega^2 \) 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 \( \rho \) to have a wqo \( (A, \leq) \) as co-domain, and check that the transitive closure \( R^+ \) of \( R \) verifies

\[
R^+ \subseteq \{ (x_i, x_j) \mid \rho(x_i) \nleq \rho(x_j) \} \tag{1.9}
\]

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

\[
R^+ \subseteq T_1 \cup \cdots \cup T_k. \tag{1.10}
\]

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

\[
T_1 = \{ (\langle a, b, c \rangle, \langle a', b', c' \rangle) \mid a > 0 \land a' < a \} \quad \text{(1.11)}
\]

\[
T_2 = \{ (\langle a, b, c \rangle, \langle a', b', c' \rangle) \mid b > 0 \land b' < b \} \quad \text{(1.12)}
\]
1.3. Examples of Applications

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 \( \rho_j \) into some wqo \( (A_j, \preceq_j) \) (see Exercise 1.23 for a generic proof that only requires each \( T_j \) to be well-founded). Then with a sequence

\[
x_0, x_1, \ldots
\]

of program configurations one can associate the sequence of tuples

\[
\langle \rho_1(x_0), \ldots, \rho_k(x_0) \rangle, \langle \rho_1(x_1), \ldots, \rho_k(x_1) \rangle, \ldots
\]

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 \( \rho_j(x_{i_1}) \not\preceq_j \rho_j(x_{i_2}) \) 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 \( \text{wf}(R_j) \supseteq R_j \), provides an initial candidate termination argument

\[
\text{wf}(R_1) \cup \cdots \cup \text{wf}(R_k).
\]

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

\[
R_1 = \{ ((a, b, c), (a', b', c')) | a > 0 \land b > 0 \land a' = a - 1 \land b' = b \land c' = 2c \}
\]

(1.16)

\[
R_2 = \{ ((a, b, c), (a', b', c')) | a > 0 \land b > 0 \land a' = 2c \land b' = b - 1 \land c' = 1 \},
\]

(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 \( \text{wf}(\text{wf}(R_{j_1}) \uplus R_{j_2}) \) of the composition of \( R_{j_1} \) with \( R_{j_2} \), then \( \text{wf}(\text{wf}(\text{wf}(R_{j_1}) \uplus R_{j_2}) \uplus R_{j_3}) \) 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 \( A \supset (B \supset A) \), \( (A \land \neg A) \supset B \), 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 formulae and $\alpha, \beta, \gamma, \ldots$ for possibly empty sequences of formulae; a sequent is an expression $\alpha \vdash A$. The rules for $\text{R}_\supset$ are:

\[
\frac{\alpha \vdash A \quad \beta A \vdash B}{\alpha \beta \vdash B} \quad \text{(Cut)}
\]

\[
\frac{\alpha AB \beta \vdash C}{\alpha BA \beta \vdash C} \quad \text{(Ex)}
\]

\[
\frac{\alpha AA \vdash B}{\alpha A \vdash B} \quad \text{(Con)}
\]

\[
\frac{\alpha \vdash A \quad \beta B \vdash C}{\alpha \beta (A \supset B) \vdash C} \quad \text{(\supset_L)}
\]

\[
\frac{\alpha A \vdash B \quad \alpha \vdash A \supset B}{\alpha \vdash A \supset (B \supset A)} \
\]

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

\[
\frac{\alpha \vdash A}{\alpha \vdash \alpha B \vdash A} \quad \text{(W)}
\]

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

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

\[
\frac{\vdots \quad \gamma A \vdash B}{\gamma \vdash A \supset B} \quad \frac{\vdots \quad \alpha \vdash A \quad \beta B \vdash C}{\alpha \beta (A \supset B) \vdash C} \quad \text{(Cut)}
\]

\[
\frac{\vdots \quad \gamma A \vdash B \quad \alpha \vdash A \quad \beta B \vdash C}{\alpha \beta \gamma \vdash C} \quad \text{(Cut)}
\]

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

**Lemma 1.18 (Subformula Property).** If $\alpha \vdash A$ is a derivable sequent in $\text{R}_\supset$, 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 $\alpha A$. 
1.3. Examples of Applications

The Decision Problem we are interested in solving is whether a formula $A$ is a theorem of $R$; it is readily generalised to whether a sequent $\alpha \vdash A$ is derivable using $(\text{Ax} - R)$.

[RI] Relevant Implication

instance: A formula $A$ of $R$.

question: Can the sequent $\vdash A$ be derived in $R$?

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 $S = \langle S, \rightarrow, \leq \rangle$, a target state $t$ of $S$, and an initial state $s$ in $S$ s.t. $t$ can be covered in $S$ from $s$ if and only if $A$ is a theorem of $R$.

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

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

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

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

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

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

For a multiset $\alpha$, define its multiset support $\sigma(\alpha)$ as its underlying set of elements $\sigma(\alpha) = \{ B \mid \alpha(B) > 0 \}$. We define the contraction $q_0 \ll$ over sequents by $\alpha \vdash B \ll \alpha' \vdash B'$ iff $\alpha \vdash B$ can be obtained from $\alpha' \vdash B'$ by some finite, possibly null, number of contractions. Over $\text{Seq}(A)$, this is equivalent to having $\alpha \leq \alpha'$ (for the product ordering over $N^{\text{Sub}(A)}$), $\sigma(\alpha) = \sigma(\alpha')$, and $B = B'$; $\ll$ over $\text{Seq}(A)$ is thus defined as a product ordering between the three wqos $(N^{\text{Sub}(A)}, \leq)$, $(\mathcal{P}(\text{Sub}(A)), =)$, and $(\text{Sub}(A), =)$, and therefore by Dickson’s Lemma:

**Lemma 1.19 (Kripke’s Lemma).** The $q_0 (\text{Seq}(A), \ll)$ is a wqo.

Then, by Exercise 1.15, the $q_0 (S, \leq)$, where $\leq$ is Hoare’s ordering applied to $\ll$, is a wqo, and we easily see that $S = \langle S, \rightarrow, \leq \rangle$ is a WSTS with effective pred-basis and a decidable ordering (see Exercise 1.25), thus the coverability problem for

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

is decidable by Proposition 1.17.
It remains to check that coverability of \( \langle S, s, t \rangle \) is indeed equivalent to derivability of \( \vdash A \). Clearly, if \( s = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n \), then any sequent appearing in any \( s_i \) along this run is derivable in \( R_D \), and if \( t \leq s_n \) — which is equivalent to the existence of a sequent \( \alpha \vdash B \) in \( s_n \), s.t. \( A \ll \alpha \vdash B \), 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 \( R_D \). Conversely, if \( \vdash A \) is derivable by a cut-free proof in \( R_D \), then we can reconstruct a run in \( S \) 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 \( \rightarrow \) along the rules \( (\text{Ax – R}) \) 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 \( S \), since \( s_0 \leq s \), there exists a run \( s' \rightarrow \cdots \rightarrow s'' \) such that \( t = \{\vdash A\} \subseteq s' \leq s'' \), proving that \( t \) is indeed coverable from \( s \) in \( S \).

1.3.3 Karp & Miller Trees

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

Formally, a \( d \)-dimensional VAS is a pair \( V = \langle x_0, A \rangle \) where \( x_0 \) is an initial configuration in \( \mathbb{N}^d \) and \( A \) is a finite set of translations in \( \mathbb{Z}^d \). A translation \( a \) in \( A \) can be applied to a configuration \( x \) in \( \mathbb{N}^d \) if \( x' = x + a \) is in \( \mathbb{N}^d \), i.e. non-negative. The resulting configuration is then \( x' \), and we write \( x \xrightarrow{a} V x' \). A \( d \)-dimensional VAS \( V \) clearly defines a WSTS \( \langle \mathbb{N}^d, \rightarrow, \leq \rangle \) where \( \rightarrow = \bigcup_{a \in A} \xrightarrow{a} V \) and \( \leq \) is the product ordering over \( \mathbb{N}^d \). A configuration \( x \) is reachable, denoted \( x \in \text{Reach}(V) \), if there exists a sequence

\[
x_0 \xrightarrow{a_1} x_1 \xrightarrow{a_2} x_2 \xrightarrow{a_3} \cdots \xrightarrow{a_n} x_n = x .
\]

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 \( \mathbb{N}_\omega \equiv \mathbb{N} \cup \{\omega\} \) of \( \mathbb{N} \) and equip it with the obvious ordering. Tuples \( y \in \mathbb{N}_\omega^d \), 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 \( \omega \)-markings are not proper configurations, it is convenient to extend the notion of steps and write \( y \xrightarrow{a} y' \) when \( y' = y + a \) (assuming \( n + \omega = \omega + n = \omega \) for all \( n \)).

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

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

2. any \( y \in C \) is in the adherence of the reachability set, i.e. \( y = \lim_{i=1,2,...} x_i \) for some infinite sequence of configurations \( x_1, x_2, ... \) in \( \text{Reach}(\mathcal{V}) \).

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

The Karp & Miller Tree constructs a particular covering of \( \mathcal{V} \). Formally, this tree has nodes labelled with \( \omega \)-markings in \( \mathbb{N}^d_\omega \) and edges labelled with translations in \( \mathcal{A} \). The root \( s_0 \) is labelled with \( x_0 \) and the tree is grown in the following way:

Assume a node \( s \) of the tree is labelled with some \( y \) and let \( y_0, y_1, \ldots, y_n \) be the sequence of labels on the path from the root \( s_0 \) to \( s \), with \( x_0 = y_0 \) and \( y_n = y \).

For any translation \( a \in \mathcal{A} \) such that there is a step \( y \xrightarrow{a} y' \), we consider whether to grow the tree by adding a child node \( s' \) to \( s \) with a \( a \)-labelled edge from \( s \) to \( s' \):

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

2. Otherwise, if \( y' > y_i \) for some \( i = 0, \ldots, n \), we build \( y'' \) from \( y' \) by setting, for all \( j = 1, \ldots, d \),

\[
y''(j) = \begin{cases} 
\omega & \text{if } y'(j) > y_i(j) \\
y'(j) & \text{otherwise}
\end{cases}
\]  

(1.22)

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

3. Otherwise, \( y' \) is not comparable with any \( y_i \): we simply add the edge and label \( s' \) with \( y' \).
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 $|A|$ children), thus by König’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 $y_0, y_1, \ldots$. By (wqo.2) applied to $\mathbb{N}^d$, we can exhibit an infinite subsequence $y_{i_0} \leq y_{i_1} \leq \cdots$ with $i_0 < i_1 < \cdots$. Any successive pair $y_{i_k} \leq y_{i_{k+1}}$ requires $y_{i_{k+1}}$ to be inserted at step 2 of the algorithm, hence $y_{i_{k+1}}$ has more $\omega$-components than $y_{i_k}$. Finally, since an $\omega$-marking has at most $d \omega$-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 $a | b$ means that $a$ divides $b$;
3. given a linearly ordered finite alphabet $\Sigma$ — e.g., $a < b < \cdots < z$ — the set of finite sequences $\Sigma^*$ with prefix ordering $\leq_{\text{pref}}$ or lexicographic ordering $\leq_{\text{lex}}$;
4. $(\mathcal{P}(\mathbb{N}), \subseteq)$, the subsets of $\mathbb{N}$ ordered with inclusion;
5. $(\mathcal{P}(\mathbb{N}), \subseteq_S)$, where $\subseteq_S$, called Smyth’s ordering, is given by $U \subseteq_S V \iff \forall n \in V, \exists m \in U, n \leq m$;
6. $(\mathcal{P}_f(\mathbb{N}), \subseteq)$ and $(\mathcal{P}_f(\mathbb{N}), \subseteq_S)$, 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 $\prod_{i=1}^m (A_i, \leq_i) = (A, \leq_\times)$ given by $A = A_1 \times \cdots \times A_m$, and

$$(x_1, \ldots, x_m) \leq_\times (x'_1, \ldots, x'_m) \iff x_1 \leq_1 x'_1 \land \cdots \land x_m \leq_m x'_m.$$  

1. Show that $\prod_{i=1}^m (A_i, \leq_i)$ is well-founded when each $(A_i, \leq_i)$ is.
2. Show that $\prod_{i=1}^m (A_i, \leq_i)$ 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 $(A_1 + A_2, \leq_{\text{lex}})$ with same support set $\{1\} \times A_1 \cup \{2\} \times A_2$ as for the disjoint sum but with an ordering defined with
\[ \langle i, x \rangle \leq_{+\leq_{\text{lex}}} \langle j, y \rangle \iff i < j \vee i = j \land x \leq_1 y. \]
Similarly, the lexicographic product $A_1 \times A_2$ has the same support set as for the disjoint product but ordered with
\[ \langle x, y \rangle \leq_{\times \leq_{\text{lex}}} \langle x', y' \rangle \iff x <_{1} x' \vee x \equiv_1 x' \land y \leq_2 y'. \]

1. Show that $A_1 + A_2$ is a linear ordering iff $A_1$ and $A_2$ are.
2. Show that $A_1 \times A_2$ is a linear ordering when $A_1$ and $A_2$ are but that the reciprocal does not hold.
3. Show that $A_1 + A_2$ and $A_1 \times A_2$ 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 qo 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 Erdős et al. (1950):

1. Consider an infinite sequence $x_0, x_1, x_2, \ldots$ over $A$ and write $M$ for the set $\{i \in \mathbb{N} | x_i \nleq x_j \text{ for all } j > i\}$. 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. (Lemma 1.7) Any upward-closed subset $U$ of a wqo $(A, \leq)$ can be written under the form $U = \langle x_1, \ldots, x_n \rangle$.

Exercise 1.9 (Linear WQOs).

1. Prove that a linear ordering is a wqo iff it is well-founded.
(2) (wqo.6) Prove that a qo is a wqo iff all its linearisations are well-founded (a result from \(\textbf{Wolk}, 1967\)), where a \textit{linearisation} of \((A, \leq)\) is any linear qo \((A, \preceq)\) with same support set and such that \(x \leq y\) implies \(x \preceq y\) (and such that \(x < y\) implies \(x \prec y\)). Here one may assume the \textit{Order-extension principle}: “every qo has a linearisation”.

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

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

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

\[
(a, b) \leq_R (a', b') \overset{\text{def}}{\iff} (a = a' \land b \leq b') \lor b < a'.
\]

This structure \((R, \leq_R)\) appeared in \(\textbf{Rado}, 1954\) and is called \textit{Rado’s structure}, see exer-
Exercises 1.13 and 1.15 for applications.

Show that $(\tilde{R}, \leq_{\tilde{R}})$ 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 $\leq_s$ defined with (1.1). We shall prove Higman’s Lemma $(A^*, \leq_s)$ is wqo iff $(A, \leq)$ is.

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

(2) Assume, by way of contradiction, that $(A, \leq)$ is wqo but $(A^*, \leq_s)$ 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 $w_i \not\preceq w_j$ 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 $\omega$-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 $A^\omega$, we let

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

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

(2) With a bounded $\omega$-word $v \in \mathbb{N}^\omega$, of the form $v = x_0, x_1, x_2, \ldots$, we associate $L(v)$, defined as $L(v) \overset{\text{def}}{=} \limsup_{i\to\omega} x_i = \lim_{k \to \infty} \max_{i \geq k} x_i$ (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 $\hat{v} = x_0, \ldots, x_{M(v)-1}$ is the shortest prefix of $v$ such that $v$ can be written $v = \hat{v} \breve{v}$ with $\breve{v}$ an $\omega$-length suffix having all its elements bounded by $L(v)$.
Assume that $w = y_0, y_1, y_2, \ldots$ is a second bounded $\omega$-word and show that

$$L(v) \leq L(w) \implies \hat{v} \leq \omega \hat{w} , \quad (E)$$

$$\left( L(v) \leq L(w) \land \hat{v} \leq_{*} \hat{w} \right) \implies v \leq_{\omega} w . \quad (E')$$

(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 $\omega$-sequence extension $(\Sigma^\omega, \leq_{\omega})$ is a wqo.

(7) Show that $(R^\omega, \leq_{R, \omega})$, the $\omega$-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 $\text{Mat}[A]$ —the set of rectangular matrices $M, N, \ldots$ with elements from $A$— by letting $M \leq_{\text{Mat}} N$ 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 $(\text{Mat}[A], \leq_{\text{Mat}})$ wqo?

**Exercise 1.15** (Ordering Powersets). Recall from Exercise 1.1 the definition of Smyth’s ordering on the powerset $P(A)$: if $(A, \leq)$ is a qo and $U, V \subseteq A$ we let:

$$U \sqsubseteq_S V \iff \forall m \in V, \exists n \in U, n \leq m . \quad (*)$$

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

$$U \sqsubseteq_H V \iff \forall n \in U, \exists m \in V, n \leq m . \quad (\dagger)$$

(1) Show that $(P(A), \sqsubseteq_H)$ and $(P(A), \sqsubseteq_S)$ 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 $\sqsubseteq_S$ and by $\sqsubseteq_H$?

(3) Express $\sqsubseteq_S$ in terms of $\sqsubseteq_H$ (and reciprocally), using set-theoretic operations like upward-closure, intersection, etc.

(4) Prove the following characterisation of wqos:

A qo $(A, \leq)$ is wqo if, and only if, $(P(A), \sqsubseteq_H)$ is well-founded. \hspace{1cm} (wqo.7)

(5) Further show that $(\mathcal{P}_f(A), \sqsubseteq_H)$ is wqo iff $(A, \leq)$ is wqo—recall that $\mathcal{P}_f(A)$ only contains the finite subsets of $A$. 

---

**Hoare ordering**

**Egli-Milner ordering**
(6) Show that \((P_f(R), \subseteq_S)\) and \((P(R), \subseteq_H)\) 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 \(\subseteq_T\), the homeomorphic embedding that extends \(\leq\). The definition of \(u \leq_T t\) is by induction on the structure of \(t\), with

\[
a(u_1, \ldots, u_m) \leq_T b(t_1, \ldots, t_k) \iff \left\{ \begin{array}{l}
a \leq b \text{ and } (u_1, \ldots, u_m) \leq_{T,*} (t_1, \ldots, t_k) \\
or \exists i \in \{1, \ldots, k\} : a(u_1, \ldots, u_m) \leq_T t_i.
\end{array} \right.
\]

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 \(U \overset{\text{def}}{=} \{t_{i,j} \mid i \in \mathbb{N} \land 1 \leq j \leq m_i\}\). 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 \(\text{term}\) remains decidable in this setting. Consider the following replacement for (compatibility):

\[
s \rightarrow s' \land s \leq t \implies s' \leq t' \lor \exists t' \geq s', t \rightarrow^+ t',
\]

where \(\rightarrow^+\) is the transitive closure of \(\rightarrow\).

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

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

\[
s \rightarrow s' \land s \leq t \implies s' \leq t' \lor \exists t' \geq s', t \rightarrow^* t',
\]

where \(\rightarrow^*\) is the reflexive transitive closure of \(\rightarrow\). We assume throughout this exercise that \(S = \langle S, \rightarrow, \leq \rangle\) is a WSTS under (rtc).
(1) Show that, if \( U \) is upward-closed, then \( \text{Pre}_U^* (U) \) is also upward-closed. Does Lemma 1.16 still hold?

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

\[
K_0 \subseteq K_1 \subseteq \cdots \text{ where } K_{n+1} \overset{\text{def}}{=} K_n \cup \text{pb}(K_n)
\]

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

(3) Show that \( \uparrow K = \text{Pre}_U^* (U) \).

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

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

\[
s \rightarrow s' \land s < t \text{ implies } \exists t', s' < t', t \rightarrow t'. \quad \text{(sc)}
\]

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 \( s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \) 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 \( \text{Post}^* (s_0) \) 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, \( \text{Post} \)-effective WSTSs with strict compatibility and decidable and antisymmetric \( \leq \):

**[Fin]** Finiteness

\textit{instance}: A transition system \( \langle S, \rightarrow \rangle \), a qo \( (S, \leq) \), and a state \( s_0 \) in \( S \).

\textit{question}: Is \( \text{Post}^* (s_0) \) finite?

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

**Exercise 1.20** (Downward WSTSs). Let \( \langle S, \rightarrow \rangle \) be a transition system and \( (S, \leq) \) be a qo. The definition of \( \text{compatibility} \) is also known as “upward-compatibility”, by contrast with its dual \textit{reflexive downward compatibility}:

\[
s \rightarrow s' \land s \geq t \text{ implies } s' \geq t \lor \exists t' \leq s', t \rightarrow t'. \quad \text{(rdc)}
\]

that defines a \textit{downward WSTS} \( S = \langle S, \rightarrow, \leq \rangle \).

Show that the following problem is decidable for image-finite, \( \text{Post} \)-effective downward WSTSs with decidable \( \leq \):

**[SCover]** Sub-Coverability
instance: A transition system ⟨S, →⟩, a qo (S, ≤), and two states s, t in S.
question: Is there a run s = s_0 → s_1 → · · · → s_n ≤ t?

Exercise 1.21 (WSTSs Everywhere). We consider a transition system S = (S, →) where S is a recursive (but otherwise arbitrary) set of configurations. For s ∈ S, let maxtime(s) be the length of the longest run starting from s. We let maxtime(s) = ∞ when arbitrary long runs exist. Define s ≤_T t when maxtime(s) ≤ maxtime(t) assuming the obvious total ordering over N ∪ {∞}.

(1) Show that (S, →, ≤_T) is a WSTS.

(2) Can we use WSTS theory and decide whether 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

\[ R \subseteq T_1 \cup \cdots \cup T_k \]

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 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 R. The key idea in this algorithm is to remove (Cor) from R_> and apply contractions only when needed, i.e. modify the rules (L) and (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 R'_> that includes the unmodified (Ax) and (R), and a modified version of (L):

\[
\frac{\alpha \vdash A \quad \beta \vdash B \quad C}{\gamma \vdash C} \quad (L')
\]

where γ ⊢ C ≪ α(Δ B) ⊢ C is such that, for all formulae D, γ(D) ≥ α(D) + β(D) − 1.

(1) Show how any derivation of a sequent α ⊢ B in R_> ∪ R'_> can be transformed into a derivation in R'_> of no larger height.
(2) Deduce that $R'_\land$ and $R_\land$ 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 $R'_\land$, then $\alpha \vdash B$ has a derivation of height at most $n$ in $R_\land$.

(4) We work now in the modified calculus $R'_\land$. We say that a derivation in $R'_\land$ 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 $R_I$ is decidable by proof search using König’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 $\text{neg}(a)$ be the vector in $\mathbb{N}^d$ defined by

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

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

(2) In order to prove that $C$ satisfies Definition 1.20, we will prove a stronger statement. For an $\omega$-marking $y$ in $\mathbb{N}^d_\omega$, first define

$$\Omega(y) \overset{\text{def}}{=} \{ j = 1, \ldots, d \mid y(j) = \omega \} \quad (\dagger \dagger)$$

the set of $\omega$-components of $y$, and

$$\overline{\Omega}(y) \overset{\text{def}}{=} \{ 1, \ldots, d \} \setminus \Omega(y) \quad (\dagger \dagger)$$

its set of finite components. We introduce for this question a variant of the construction found in the main text, which results in a Karp & Miller graph $G$ instead of a tree: in step 2 we rather add an edge $s \xrightarrow{u} G s_i$. Observe that this does not change $C$ nor the termination of the algorithm.

Show that, if $x_0 \xrightarrow{u}_G x$ for some translation sequence $u$ in $A^*$, then there exists a node $s$ in $G$ labelled by $y$ such that $x(j) = y(j)$ for all $j$ in $\overline{\Omega}(y)$ 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 $\gamma$ that agree with $y$ on its finite components, and that can be made arbitrarily high on its $\omega$-components. For this, we focus on the graph nodes where new $\omega$ values are introduced by step 2 which we call $\omega$-nodes.

Prove that, if $s_0 \xrightarrow{u} G s$ labelled $y$ for some $u$ in $A^*$ in the tree and $z$ in $\mathbb{N}^{\Omega(y)}$ is a partial configuration on the components of $\Omega(y)$, then there are
• $n$ in $\mathbb{N}$,
• a decomposition $u = u_1u_2 \cdots u_{n+1}$ with each $u_i$ in $A^*$ where the nodes $s_i$ reached by $s_0 \xrightarrow{u_1} \cdots \xrightarrow{u_{n+1}} s_i$ for $i \leq n$ are $\omega$-nodes,
• sequences $w_1, \ldots, w_n$ in $A^+$,
• numbers $k_1, \ldots, k_n$ in $\mathbb{N}$,

such that $x_0 \xrightarrow{u_1^{k_1} u_2 \cdots u_n^{k_n} u_{n+1}} \gamma x$ with $x(j) = y(j)$ for all $j$ in $\Omega(y)$ and $x(j) \geq z(j)$ for all $j$ in $\Omega(y)$. 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 (1962). The reader will find more information in the survey of Milner (1989), 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 $\theta(A, \subseteq)$ and/or $(A^{\omega}, \subseteq_{\omega})$ 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.

**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 $R_{\lor, \land}$, 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 $R_{\lor, \land}$ to be Ackermannian (see CRI on page 114, while $R_1$ was shown to be 2ExpTime-complete only very recently (Schmitz, 2016a). The decidability of implicative ticket logic $T_{\lor}$ 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.
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
\[ \langle a_0, b_0 \rangle, \ldots, \langle a_j, b_j \rangle, \ldots, \] (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
\[ \langle 2, 3, 2^0 \rangle \xrightarrow{r} \langle 1, 3, 2^1 \rangle \xrightarrow{r} \langle 2^2, 2, 2^0 \rangle \xrightarrow{r^{2^2-1}} \langle 2^{2^2}, 1, 1 \rangle \xrightarrow{r^{2^{2^{2^2}}-1}} \langle 2^{2^{2^2}}, 0, 1 \rangle, \]
of length
\[ 2 + 2^2 + 2^{2^2}, \] (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 \((\mathcal{F}_k)_{k \in \mathbb{N}}\) 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 \(\mathcal{F}_3\), which is the lowest level to contain non-elementary functions. Chapter 3 will be devoted to further matching complexity lower bounds for decision problems on monotonic counter systems.

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

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

2.1.1 Controlled Sequences

Our first issue with our program is that one can construct arbitrarily long bad sequences, even when starting from a fixed first element. Consider $\mathbb{N}^2$ and fix $x_0 = \langle 0, 1 \rangle$. Then the following

$$ \langle 0, 1 \rangle, \langle L, 0 \rangle, \langle L - 1, 0 \rangle, \langle L - 2, 0 \rangle, \ldots, \langle 2, 0 \rangle, \langle 1, 0 \rangle $$

is a bad sequence of length $L + 1$. What makes such examples possible is the “uncontrolled” jump from an element like $x_0$ to an arbitrarily large next element, here $x_1 = \langle L, 0 \rangle$. 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 $|.|_A: A \to \mathbb{N}$ on the elements to represent their size. We always assume $A_{\leq n} \overset{\text{def}}{=} \{ a \in A \mid |a|_A \leq n \}$ 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 $|\langle m, n \rangle|_{\mathbb{N}^2} \overset{\text{def}}{=} |\langle m, n \rangle|_{\infty} = \max(m, n)$;

2. the second is a control function $g: \mathbb{N} \to \mathbb{N}$ 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 $x$. 

![Figure 2.2: The Grzegorczyk hierarchy of primitive-recursive functions.](image)
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 \( n \in \mathbb{N} \) if

\[
\forall i = 0, 1, 2, \ldots : |a_i|_A \leq g^i(n) \overset{\text{def}}{=} g(g(\cdots g(n)))) . \tag{2.4}
\]

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 \textsc{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 \(|j|_{[k]} \overset{\text{def}}{=} j\) for all \(0 \leq j < k\), and \(\Gamma_k\) the generic \(k\)-elements nwqo \(\{a_0, \ldots, a_{k-1}\}, =, |.|_{\Gamma_k}\) where \(|a_j|_{\Gamma_k} \overset{\text{def}}{=} 0\) for all \(0 \leq j < k\).

**LENGTH FUNCTION.** The outcome of these definitions is that, unlike in the uncontrolled case, there is a longest \((g, n)\)-controlled bad sequence over any nwqo \((A, \leq_A, |.|_A)\): indeed, we can 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 \textit{König’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 \( L_{g,A} \).

**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.

**Isomorphisms.** 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.
2.1. The Length of Controlled Bad Sequences

**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 \(\Gamma_2\), 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 \(\Gamma_1\) 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.

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
\[
A_1 + A_2 \overset{\text{def}}{=} \{\langle i, a \rangle \mid i \in \{1, 2\} \text{ and } a \in A_i\},
\]
\[
\langle i, a \rangle \leq_{A_1+A_2} \langle j, b \rangle \overset{\text{def}}{\iff} i = j \text{ and } a \leq_{A_i} b,
\]
\[
|\langle i, a \rangle|_{A_1+A_2} \overset{\text{def}}{=} |a|_{A_i}.
\]

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

The cartesian product of two nwqos \((A_1, \leq_{A_1}, |.|_{A_1})\) and \((A_2, \leq_{A_2}, |.|_{A_2})\) is the nwqo \((A_1 \times A_2, \leq_{A_1 \times A_2}, |.|_{A_1 \times A_2})\) defined by
\[
A_1 \times A_2 \overset{\text{def}}{=} \{\langle a_1, a_2 \rangle \mid a_1 \in A_1, a_2 \in A_2\},
\]
\[
\langle a_1, a_2 \rangle \leq_{A_1 \times A_2} \langle b_1, b_2 \rangle \overset{\text{def}}{\iff} a_1 \leq_{A_1} b_1 \text{ and } a_2 \leq_{A_2} b_2,
\]
\[
|\langle a_1, a_2 \rangle|_{A_1 \times A_2} \overset{\text{def}}{=} \max_{i \in \{1, 2\}} |a_i|_{A_i}.
\]

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

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

**Definition 2.4.** The set of polynomial nwqos is the smallest set of nwqos containing \(\Gamma_0\), \(\Gamma_1\), and \(\mathbb{N}\) and closed under the + and \(\times\) operations.

**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*: \( \Gamma_0 \) is neutral for + and absorbing for \( \times \):
\[
\Gamma_0 + A \equiv A + \Gamma_0 \equiv A \quad \quad \Gamma_0 \times A \equiv A \times \Gamma_0 \equiv \Gamma_0,
\]
(2.11)
\( \Gamma_1 \) is neutral for \( \times \):
\[
\Gamma_1 \times A \equiv A \times \Gamma_1 \equiv A,
\]
(2.12)
+ is associative and commutative:
\[
A + (B + C) \equiv (A + B) + C \quad \quad A + B \equiv B + A,
\]
(2.13)
\( \times \) is associative and commutative:
\[
A \times (B \times C) \equiv (A \times B) \times C \quad \quad A \times B \equiv B \times A,
\]
(2.14)
and \( \times \) distributes over +:
\[
(A + B) \times C \equiv (A \times C) + (B \times C).
\]
(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)
\[
A \equiv \mathbb{N}^{d_1} + \cdots + \mathbb{N}^{d_m}
\]
(2.16)
for \( m, d_1, \ldots, d_m \geq 0 \). In particular, we denote the PNF of \( \Gamma_0 \) 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 \( 2^{2^{\cdots^{2}}} \) \( b \) 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** \( (\mathcal{R}_k)_{k<\omega} \) 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:
\[
\bigcup_{k<\omega} \mathcal{R}_k = \text{FPR}.
\]
(2.17)
The lower levels correspond to reasonable classes, \( \mathcal{R}_0 = \mathcal{R}_1 \) being the class of linear functions, and \( \mathcal{R}_2 \) that of elementary functions. Starting at level 1, the hierarchy is strict in that \( \mathcal{R}_k \subset \mathcal{R}_{k+1} \) for \( k > 0 \) (see Figure 2.2 on page 27).

At the heart of each \( \mathcal{R}_k \) lies the \( k \)th *fast-growing function* \( F_k : \mathbb{N} \to \mathbb{N} \), which
is defined for finite \( k \) by
\[
F_0(x) \overset{\Delta}{=} x + 1, \quad F_{k+1}(x) \overset{\Delta}{=} F_k^{x+1}(x) = F_k(F_k(\cdots F_k(x)))). \tag{2.18}
\]
This hierarchy of functions continues with ordinal indices, e.g.
\[
F_\omega(x) \overset{\Delta}{=} F_x(x). \tag{2.19}
\]
Observe that
\[
F_1(x) = 2x + 1, \quad F_2(x) = 2^{x+1}(x + 1) - 1, \tag{2.20}
\]
\[
F_3(x) > 2^{2^\cdot\cdot\cdot^x} \times \text{etc.} \tag{2.21}
\]
For \( k \geq 2 \), each level of the Grzegorczyk hierarchy can be characterised as
\[
\mathcal{F}_k = \{ f \mid \exists i, \text{ \( f \) 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 \( F_k \), \( F_k \in \mathcal{F}_k \) for all \( k \). On the other hand, every function \( f \) in \( \mathcal{F}_k \) 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 \( k > 0 \),
\[
F_{k+1} \not\in \mathcal{F}_k. \tag{2.23}
\]
In particular, \( F_\omega \) is (akin to) the diagonal Ackermann function: it is not primitive-recurive and eventually bounds every primitive recursive function.

We delay more formal details on \( (\mathcal{F}_k)_k \) 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 \( \mathcal{F}_\gamma \) 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 \( \mathcal{F}_{\gamma+d} \).

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 \textsc{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 \( \mathcal{F}_1 \), the **Length Function Theorem** with \( p = \gamma = 1 \) entails the existence of a bounding function in \( \mathcal{F}_3 \) on the length of any run of \textsc{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 $F_1$, with $d$ counters, and with $p$ states, the Length Function Theorem yields an upper bound in $F_{d+1}$ on the length of controlled bad sequences. This is improved to $F_d$ by Corollary 2.36 on page 46. When $b$ or $p$ is part of the input, this rises to $F_{d+1}$, and when $d$ is part of the input, to $F_d$, which asymptotically dominates every primitive-recursive function.

2.2.1 Termination Algorithm

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

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

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 $S$ of length $\ell \triangleq f(n) + 1$ is necessarily good, we can replace the algorithm in the proof of Proposition 1.15 by an algorithm that looks for a finite witness of non-termination of form

$$s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_\ell. \quad (2.25)$$

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 $g^\ell$. 
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 8 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

\[
\uparrow\{t\} = I_0 \subset I_1 \subset \cdots \subset I_\ell, \text{ where } s \in I_\ell \text{ but } s \not\in I_i \text{ for any } i < \ell. \tag{2.26}
\]

For each \( i = 1, \ldots, \ell \), let \( s_i \) be a minimal element in the non-empty set \( I_i \setminus I_{i-1} \); 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 \( s_j \not\in s_i \) for \( j > i \), since \( s_j \not\in I_i \) 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 \( s_i \not\leadsto s_{i+1} \), 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 \( I_i \setminus I_{i-1} \). Assume again that \( \mathcal{S} = (S, \rightarrow, \leq) \) 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

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

(2.29)

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

$$n, n-1, \ldots, 1, 0$$

(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

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

(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:

**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 $A/a = \{a' \in A \mid a \not\leq a'\}$ of elements that are not above $a$.

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

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

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

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 nwqs: a sequence of successive residuals \( A \supset A/a_0 \supset A/a_0/a_1 \supset \cdots \) 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

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

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 \( L_{g,N} \): by the Descent Equation

\[
L_{g,N}(n) = \max_{k \in N_{\leq n}} \left\{ 1 + L_{g,N/k}(g(n)) \right\}
= \max_{k \in N_{\leq n}} \left\{ 1 + L_{g,[k]}(g(n)) \right\}
= \max_{k \in N_{\leq n}} \{ 1 + k \}
= n + 1
\]
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 \( \Gamma_n \) and to deduce the value of \( L_{g,\mathbb{N}}(n) \). However, as already mentioned in Example 2.3 on page 29, \( \Gamma_n \neq [n] \) 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 \( [n] \) than in \( \mathbb{N} \). 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.

**Definition 2.14.** A nwqo reflection is a mapping \( h: A \rightarrow B \) between two nwqos that satisfies the two following properties:

\[
\forall a, a' \in A : h(a) \leq_B h(a') \implies a \leq_A a', \hspace{1cm} (2.36)
\]

\[
\forall a \in A : |h(a)|_B \leq |a|_A. \hspace{1cm} (2.37)
\]

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

We write \( h: A \hookrightarrow B \) 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 \( h: A \hookrightarrow B \) and \( h': B \hookrightarrow C \) entails \( h' \circ h: A \hookrightarrow C \). It is also reflexive, hence reflection is a quasi-ordering. Any nwqo reflects its induced substructures since \( \text{Id}: X \hookrightarrow A \) 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 \( h: A \hookrightarrow B \), 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 \( A \)).

\[
A \hookrightarrow B \text{ implies } L_{g,A}(n) \leq L_{g,B}(n) \text{ for all } g, n. \hspace{1cm} (2.38)
\]
This is the last missing piece for deducing (2.29) from (2.31): since \([k] \leftrightarrow \Gamma_k\), \(L_{g,[k]}(n) \leq L_{g,\Gamma_k}(n)\) 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:

\[ A \leftrightarrow A' \text{ and } B \leftrightarrow B' \text{ imply } A + B \leftrightarrow A' + B' \text{ and } A \times B \leftrightarrow A' \times B'. \quad (2.39) \]

**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 \(A/a_0/a_1/\ldots/a_n\) 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 \(\Gamma_k\), and

\[ \mathbb{N}/k \leftrightarrow \Gamma_k \]  
for the naturals \(\mathbb{N}\)—because \(\mathbb{N}/k = [k]\) by (2.32), and then \([k] \leftrightarrow \Gamma_k\) 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

\[ (A + B)/\langle 1, a \rangle = (A/a) + B, \quad (A + B)/\langle 2, b \rangle = A + (B/b), \]  
(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 \(\mathbb{N}^2\)

\[ \langle 3, 2 \rangle, \langle 5, 1 \rangle, \langle 0, 4 \rangle, \langle 17, 0 \rangle, \langle 1, 1 \rangle, \langle 16, 0 \rangle, \langle 0, 3 \rangle. \]  
(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 \text{ or } 2 \not\leq b. \text{ Thus we can partition the elements of this suffix into two groups: the pairs where the first coordinate is in } \mathbb{N}/3, \text{ and the pairs where the second coordinate is in } \mathbb{N}/2—\text{ an element might fulfill both conditions, in which case we choose an arbitrary group for it. Thus the elements of the suffix can be either from } (\mathbb{N}/3) \times \mathbb{N} \text{ or from } \mathbb{N} \times (\mathbb{N}/2), \text{ 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)

\[ \langle 3, 2 \rangle, \{ \langle 5, 1 \rangle, \langle 17, 0 \rangle, \langle 1, 1 \rangle, \langle 16, 0 \rangle, \ldots \} \in \mathbb{N} \times (\mathbb{N}/2) \]
\[ \ldots \langle 0, 4 \rangle, \ldots \ldots \langle 0, 3 \rangle \in (\mathbb{N}/3) \times \mathbb{N} \] (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:

\[ (A \times B)/\langle a, b \rangle \leftrightarrow ((A/a) \times B) + (A \times (B/b)) \] (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

It is time to wrap up our analysis of \( L \). We first combine the inductive residuation and reflection operations into derivation relations \( \partial_n \): intuitively, the relation \( A \partial_n A' \) is included in the relation “\( A/a \leftrightarrow A' \) 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

\[ \partial_n 0 \defeq \emptyset, \] (2.45)
\[ \partial_n N^0 \defeq \{0\}, \] (2.46)
\[ \partial_n N^d \defeq \{N^{d-1} \cdot nd\}, \] (2.47)
\[ \partial_n (A + B) \defeq ((\partial_n A) + B) \cup (A + (\partial_n B)) \] (2.48)
for \( d > 0 \) and \( A, B \) in PNF; in these definitions the + operations are lifted to act upon nwqo sets \( S \) by \( A + S \overset{df}{=} \{ A + A' \mid A' \in S \} \) and symmetrically. Note that (2.46) can be seen as a particular case of (2.47) if we ignore the undefined \( \mathbb{N}^{0-1} \) and focus on its coefficient 0.

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

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

The definition of \( \partial_n \) 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. \( A/a \mapsto A' \).

**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 \( A/a \mapsto \Gamma_a \) by (2.40), and then \( \Gamma_a \mapsto \Gamma_n \equiv \mathbb{N}^0 \cdot n \) 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),

\[
A/a \mapsto (\mathbb{N}/k) \times \mathbb{N}^{d-1} + (\mathbb{N} \times (\mathbb{N}^{d-1}/b)).
\]

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

\[
\mapsto (\mathbb{N}^0 \cdot n) \times \mathbb{N}^{d-1} + (\mathbb{N} \times (\mathbb{N}^{d-1}/b))
\]

\[
\equiv (\mathbb{N}^{d-1} \cdot n) + (\mathbb{N} \times (\mathbb{N}^{d-1})).
\]

Using the ind. hyp. on \( \mathbb{N}/b \) along with Remark 2.17,

\[
\mapsto (\mathbb{N}^{d-1} \cdot n) + (\mathbb{N} \times (\mathbb{N}^{d-2} \cdot n(d - 1)))
\]

\[
\equiv \mathbb{N}^{d-1} \cdot nd,
\]

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) \( A/a \mapsto (B/b) + C \). By ind. hyp., there exists \( B' \in \partial_n B \) s.t. \( B/b \mapsto B' \), thus \( A/a \mapsto B' + C \) 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):

\[
\partial_n A = \{ B + \partial_n \mathbb{N}^d \mid A \equiv B + \mathbb{N}^d, d \geq 0 \}.
\]
The bounding function $M_{g,A}$ for $A$ a polynomial nwqo in PNF is defined by

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

$$L_{g,A}(n) \leq M_{g,A}(n).$$  \hspace{1cm} (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. $A/a \leftrightarrow A'$, 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 $M$. \hfill \square

### 2.4 * Classification in the Grzegorczyk Hierarchy

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

#### 2.4.1 Maximal Order Types

Consider a wqo $(A, \leq)$: it defines an associated strict ordering $< \overset{\text{def}}{=} \{(a, a') \in A^2 \mid a \leq a' \text{ and } a' \not\leq a\}$. There are many possible linearisations $\prec$ of $<$, i.e. linear orders with $< \overset{\text{def}}{=} \prec$, 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.

**Example 2.21** (Maximal Order Types). In a finite set $\Gamma_k$, the strict ordering is empty and the $k!$ different linear orders over $\Gamma_k$ 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 \(\omega\)):

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

**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 \(\varepsilon_0\) be the supremum of the family of ordinals \(\{0, 1, \omega, \omega^\omega, \omega^{\omega^\omega}, \ldots\}\) (in other words \(\varepsilon_0\) is the smallest solution of the equation \(\omega^x = x\)). It is well-known that ordinals below \(\varepsilon_0\) can be written down in a canonical way as ordinal terms in Cantor Normal Form (CNF), i.e. sums

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

with \(\alpha > \beta_1 \geq \cdots \geq \beta_m \geq 0\) and each \(\beta_i\) itself a term in CNF. We write 1 for \(\omega^0\) and \(\alpha \cdot n\) for \(\underbrace{\alpha + \cdots + \alpha}_{n\text{ times}}\). 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 \(\alpha\) of form \(\gamma + 1\) is called a successor ordinal. Otherwise, if not 0, it is a limit ordinal, usually denoted \(\lambda\). We write \(\text{CNF}(\alpha)\) for the set of ordinal terms \(\alpha' < \alpha\) in CNF (which is in bijection with the ordinal \(\alpha\), and we use ordinal terms in CNF and set-theoretic ordinals interchangeably).

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

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

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 \(\omega^\omega\) 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 nwqos \(A\) and \(B\)

\[
o(A + B) = o(A) \oplus o(B), \quad o(A \times B) = o(A) \otimes o(B). \tag{2.55}
\]

**Example 2.24.** Given a polynomial nwqo in PNF \(A \equiv \sum_{i=1}^{m} N^{d_i}\), its associated maximal order type is \(o(A) = \bigoplus_{i=1}^{m} \omega^{d_i}\), which is in \(\omega^\omega\). It turns out that \(o\) is a bijection between polynomial nwqos and \(\omega^\omega\) (see Exercise 2.7).
It is more convenient to reason with ordinal arithmetic rather than with polynomial nwqs, and we lift the definitions of \( \partial \) and \( M \) to ordinals in \( \omega^\omega \). Define for all \( \alpha \) in \( \omega^\omega \) and all \( d, n \) in \( \mathbb{N} \)

\[
\partial_n \omega^d \begin{cases} 
0 & \text{if } d = 0 \\
\omega^{d-1} \cdot (nd) & \text{otherwise}
\end{cases}
\] (2.56)

\[
\partial_n \alpha \equiv \{ \gamma \oplus \partial_n \omega^d \mid \alpha = \gamma \oplus \omega^d \}
\] (2.57)

\[
M_{g,\alpha}(n) \equiv \max_{\alpha' \in \partial_n \alpha} \{ 1 + M_{g,\alpha'}(g(n)) \} .
\] (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 \( \omega^\omega \); 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 \( n \),

\[
L_{g,A}(n) \leq M_{g,o(A)}(n) .
\] (2.59)

A benefit of ordinal notations is that the well-foundedness of \( \partial \) 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

\[
M_{g,k}(n) = k \\
M_{g,\omega}(n) = n + 1 .
\] (2.60)

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

\[
\partial_n k = \begin{cases} 
\emptyset & \text{if } k = 0 \\
\{k-1\} & \text{otherwise}
\end{cases} , \\
\partial_n \omega = n .
\] (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 \( \lambda \), 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 \( \omega_x \) for \( \omega \) and to define on ordinal terms in CNF

\[
(\gamma + \omega^x)^{+1} = \gamma + \omega^x \cdot \omega_x , \\
(\gamma + \omega^\lambda)^x = \gamma + \omega^\lambda .
\] (2.62)

We always assume the standard assignment \( \omega_x \equiv x + 1 \) in the remainder of the chapter. Note that this assignment implies \( \lambda_x > 0 \) for all \( x \).
2.4. Classification in the Grzegorczyk Hierarchy

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

\[
P_x(\alpha + 1) \overset{\text{def}}{=} \alpha, \quad P_x(\lambda) \overset{\text{def}}{=} P_x(\lambda_x).
\]

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)

\[
P_x(\gamma + \alpha) = \gamma + P_x(\alpha).
\]

Observe that predecessors of ordinals in \(\omega^\omega\) 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 \(\omega^d\) 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 \(h : \mathbb{N} \to \mathbb{N}\). We define the Cichoń hierarchy \((h_\alpha)_{\alpha \in \varepsilon_0}\) by

\[
h_0(x) \overset{\text{def}}{=} 0, \quad h_{\alpha + 1}(x) \overset{\text{def}}{=} 1 + h_\alpha(h(x)), \quad h_\lambda(x) \overset{\text{def}}{=} h_{\lambda_x}(x).
\]

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

**Lemma 2.27.** For all \(\alpha > 0\) in \(\varepsilon_0\) and \(x\),

\[
h_\alpha(x) = 1 + h_{P_x(\alpha)}(h(x)).
\]

**Proof.** By transfinite induction over \(\alpha > 0\). 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 \(\lambda\), \(h_\lambda(x) = h_{\lambda_x}(x)\) is equal to \(1 + h_{P_x(\lambda_x)}(h(x))\) by ind. hyp. since \(0 < \lambda_x < \lambda\), which is the same as \(1 + h_{P_x(\lambda)}(h(x))\) by definition of \(P_x(\lambda)\). \(\square\)

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

\[
H_{\omega^2}(x) = H_{\omega^2 + 1}(x) = H_\omega(2x + 1) + x = 3x + 1, \quad (2.67)
\]

\[
H_{\omega^2}(x) = (2^{x+1} - 1)(x + 1). \quad (2.68)
\]

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 \( h_\alpha \) 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 \( \mathcal{F}_\gamma \) and \( \alpha < \omega^{d+1} \), then \( h_\alpha \) is bounded by a function in \( \mathcal{F}_{\gamma+d} \).

2.4.3 Monotonicity

One obstacle subsists before we can finally prove the Length Function Theorem: the functions \( M_{g,\alpha} \) and \( h_\alpha \) are not monotone in the parameter \( \alpha \). 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 \( h_\alpha \), 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 \( \omega^\omega \) by

\[
\omega^{d_1} + \cdots + \omega^{d_m} \sqsubseteq \omega^{d_1'} + \cdots + \omega^{d_n'} \quad \iff \quad m \leq n \text{ and } \forall 1 \leq i \leq m, \ d_i \leq d_i'
\]

for ordinal terms in \( \text{CNF}(\omega^\omega) \), 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):

\[
\alpha \sqsubseteq \alpha' \text{ and } \gamma \sqsubseteq \gamma' \implies \alpha \oplus \gamma \sqsubseteq \alpha' \oplus \gamma' .
\]

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

\[
M_{g,\alpha}(n) = \max_{\alpha = \gamma \oplus \omega^d} \{ 1 + M_{g,\partial_{d,n} \alpha}(g(n)) \} .
\]

Lemma 2.31 (Structural Monotonicity). Let \( \alpha, \alpha' \) be in \( \omega^\omega \) 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

\[
M_{g,\alpha}(x) = 1 + M_{g,\partial_{d_i,x} \alpha}(g(x)) .
\]
As \( i \leq n \) and \( d_i \leq d'_i \), observe that \( \partial_{d_i,x} \alpha \subseteq \partial_{d'_i,x} \alpha' \), and by \textbf{Fact 2.18}, we can apply the induction hypothesis:

\[
M_{g,\alpha}(x) \leq 1 + M_{g,\partial_{d'_i,x} \alpha'}(g(x)) \\
\leq M_{g,\alpha'}(x).
\]

An important consequence of \textbf{Lemma 2.31} is that there is a maximising strategy for \( M \), which is to always derive along the smallest term:

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

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

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

\[
\partial_{d,n'} \partial_{d',n} \alpha \subseteq \partial_{d',n'} \partial_{d,n} \alpha. \quad (2.73)
\]

The lemma follows immediately from the claim, \textbf{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 \( d' = 0 \),

\[
\partial_{d,n'} \partial_{d',n} \alpha = \gamma \oplus (\omega^{d'-1} \cdot nd' + \omega^{d-1} \cdot n'd) \\
\partial_{d',n'} \partial_{d,n} \alpha = \gamma \oplus (\omega^{d'-1} \cdot n'd' + \omega^{d-1} \cdot nd).
\]

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 \( \omega^{d'-1} \) terms, thus fulfilling the second condition of (2.69). \Box

Let us conclude with a comparison between derivatives and predecessors:

\textbf{Corollary 2.33.} If \( 0 < \alpha < \omega^{d+1} \), then

\[
M_{g,\alpha}(n) \leq 1 + M_{g,P_{n\alpha}}(g(n)).
\]

\textbf{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 \textbf{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

\[
\gamma + \partial_n 1 = P_{n\alpha} = \gamma
\]

and the statement holds. Otherwise, by (2.70)

\[
\gamma + \partial_n \omega^{d'} = \gamma + \omega^{d'-1} \cdot nd'
\]
\[
\subseteq \gamma + \omega^{d'-1} \cdot nd + P_{n\omega^{d'-1}}
\]
\[
= P_{n\alpha},
\]

from which we deduce the result by \textbf{Lemma 2.31}. \Box
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

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

(2.74)

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

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

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

$$M_{g,\alpha}(n) \leq 1 + h_{P_{\alpha_p}(\alpha)}(g(nd))$$

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

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

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

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

(2.75)

which is optimal (recall examples 2.26 and 2.28).

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

$$L_{g,\alpha}(n) \leq M_{g,\alpha_p}(n) \leq h_{\alpha_p}(nd) \leq F_{dp}(nd) - nd ,$$

(2.76)

the latter being a function in $\mathcal{F}_d$ 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 $L_{g,\alpha}(n) \leq M_{g,\alpha_p}(n) \leq h_{\alpha_p}(nd) \leq F_{dp}(nd) - nd$, so $L_{g,\alpha}(n)$ is bounded by a function in $\mathcal{F}_d$. 
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 \( \mathcal{F}_\gamma \) for some \( \gamma \geq 1 \) and \( d, p \geq 0 \). Then \( L_{g, \mathbb{N}^d} \) is bounded by a function in \( \mathcal{F}_\gamma + d \).

**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 \( \mathcal{F}_\gamma \).

For \( d > 0 \) we first use Corollary 2.25:

\[
L_{g, A}(n) \leq M_{g, o(A)}(n)
\]

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

\[
L_{g, A}(n) \leq h_{o(A)}(nd)
\]

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 \( \mathcal{F}_\gamma \) if and only if \( h \) is bounded in \( \mathcal{F}_\gamma \), and thus by Fact 2.30 \( L_{g, A} \) is bounded in \( \mathcal{F}_{\gamma + d} \).

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 \textsc{simple} was an example of a bad sequence with length function in \( \mathcal{F}_3 \). Exercise 2.15 generalises \textsc{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 $\mathcal{F}_k$ of the Grzegorczyk hierarchy is formally defined as the closure of the constant zero function $0$, the sum function $+ : x_1, x_2 \mapsto x_1 + x_2$, the projections $\pi^i_n : x_1, \ldots, x_n \mapsto x_i$ for all $0 < i \leq n$, and the fast-growing function $F_k$, under two basic operations:

- **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**: if $h_1, h_2$, and $h_3$ belong to the class, then so does $f$ if
  \[ f(0, x_1, \ldots, x_n) = h_1(x_1, \ldots, x_n), \]
  \[ f(y + 1, x_1, \ldots, x_n) = h_2(y, x_1, \ldots, x_n, f(y, x_1, \ldots, x_n)), \]
  \[ f(y, x_1, \ldots, x_n) \leq h_3(y, x_1, \ldots, x_n). \]

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

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

- predecessor : $x \mapsto x - 1$,
- cut-off subtraction : $x, y \mapsto x \div y$,
- odd: $x \mapsto x \mod 2$.

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

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

(4) Show that if a function $f(x_1, \ldots, x_n)$ belongs to $\mathcal{F}_k$ 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 $\mathcal{F}_k$ 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 3, where we synthesised linear ranking functions $\rho_j$ into $\mathbb{N}$ for each $T_j$.

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

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

\[ A \times B / (a, b) \neq (A/a \times \uparrow_B b) \cup (A/a \times B / b) \cup (\uparrow_A a \times B / b) \], \hspace{1cm} (\ast) \]

\[ A \times B / (a, b) \neq (A/a \times B) \cup (A \times B / b). \hspace{1cm} (\beta) \]
Exercise 2.6 (Derivatives). Prove \( \partial_n A = \{ B + \partial_n \mathbb{N}^d \mid A \equiv B + \mathbb{N}^d \} \).

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(\( \omega^\omega \)).

Exercise 2.8 (Well-Foundedness of \( \partial \)). 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:

\[
\sum_{i=1}^m \beta_i < \sum_{i=1}^n \beta_i' \iff \begin{cases} m < n \text{ and } \forall 1 \leq i \leq m, \beta_i = \beta_i', & \\
\exists 1 \leq j \leq \min(m, n), \beta_j < \beta_j' \text{ and } \forall 1 \leq i < j, \beta_i = \beta_i'. & \end{cases}
\]

Prove Fact 2.18: The relation \( \partial \) defined using a unary function is well-founded.

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

Exercise 2.10 (Structural Monotonicity). Prove Equation (2.70) \( \sqsubseteq \) is a precongruence for \( \oplus \).

Exercise 2.11 (Structural Monotonicity). Let \( \alpha, \alpha' \) be in \( \omega^\omega \) 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.

By wqo.2 (stated on page 41), 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:

\[
L_{g,r,A}(n) = L_{g,A \times \Gamma_r}(n). \tag{8}
\]

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

(2) Given a sequence \( a_0, a_1, \ldots, a_r \) 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 \( L_{g,r,A}(n) \leq L_{g,A \times \Gamma_p}(n) \).

(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 \( h: \mathbb{N} \to \mathbb{N} \) by

\[
h^0(x) \overset{\text{def}}{=} x, \quad h^{\alpha+1}(x) \overset{\text{def}}{=} h^\alpha(h(x)), \quad h^\lambda(x) \overset{\text{def}}{=} h^{\lambda^+(x)}(x).
\]
Observe that $h^\alpha$ is intuitively the $\alpha$th (transfinite) iterate of the function $h$. As with the Cichoń hierarchy, one case is of particular interest: that of $(H^\alpha)_\alpha$ for $H(x) \overset{\text{df}}{=} x + 1$. 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 $\alpha, x$. What about $h_\alpha(x)$ and $h^\alpha(x) - x$ if $h(x) > x$?

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

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

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

5. Show that $h_\alpha$ measures the finite length of the iteration in $h^\alpha$, i.e. that $h^\alpha(x) = h^{h_\alpha(x)}(x)$ for all $h, \alpha, x$—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 $(A, x_0)$ with maximal increment $b = \max_{a \in A} |a|$, and maximal initial counter value $n = |x_0|$. Show using Exercise 2.13 that the finite values in this tree are bounded by $h^{\omega^d}(nd)$ 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 $\preceq_{\text{lex}}$ (with most significant element last) defined by

$$ x \preceq_{\text{lex}} y \overset{\text{df}}{\iff} x(d) < y(d) \text{ or } (x(d) = y(d) \quad \text{and} \quad \langle x(1), \ldots, x(d-1) \rangle \prec_{\text{lex}} \langle y(1), \ldots, y(d-1) \rangle) . $$

This is a linearisation of the product ordering over $\mathbb{N}^d$; writing $\mathbb{N}^d_{\text{lex}}$ for the associated nwqo $(\mathbb{N}^d, \preceq_{\text{lex}}, |.|)$, 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 $\preceq_{\text{lex}}$ 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 $n$,

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

(1) Let $n > 0$, and write a program $\textsc{lex}_d(g, n)$ with $d$ counters $x(1), \ldots, x(d)$ 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 $c$ holding the current value of the control. The run of $\textsc{lex}_d(g, n)$ should be a sequence $(x_1, c_1), (x_2, c_2), \ldots, (x_\ell, c_\ell)$ of pairs $(x_i, c_i)$ composed of a vector $x_i$ in $\mathbb{N}^d$ and of a counter $c_i$.

(2) Let $(x_1, c_1), (x_2, c_2), \ldots, (x_\ell, c_\ell)$ be the unique run of $\textsc{lex}_d(g, n)$ for $n > 0$. Define

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

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

$$ g_{\omega^d}(n) = i + g_{\alpha(x_i)}(c_i) . $$

(\text{**})
Deduce (3).

Show that, if \((x_1, c_1), (x_2, c_2), \ldots, (x_\ell, c_\ell)\) is the run of \(\text{lex}_d(g, n)\) for \(n > 0\), then \(c_\ell = g^{\omega^d}(n)\).

**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 Cichon and Tahhan Bittar (1998) and Weiermann (1994); furthermore the latter provides upper bounds for the more general Kruskal’s Tree Theorem.

The hierarchy \((\mathcal{F}_k)_{k \geq 2}\) described as the Grzegorczyk hierarchy in Section 2.1.3 and Section 2.4 is actually due to L"ob and Wainer (1970); its relationship with the original Grzegorczyk hierarchy \((\mathcal{E}^k)_k\) (Grzegorczyk, 1953) is that \(\mathcal{F}_k = \mathcal{E}^{k+1}\) for all \(k \geq 2\). There are actually some difference between our definition of \((\mathcal{F}_k)_k\) and that of L"ob 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 Cichon hierarchy was first published in (Cichon 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 (4) in Exercise 2.15 relates the Cichon hierarchy with the length of bad sequences for the lexicographic ordering over \(\mathbb{N}^d\), i.e. with the length of decreasing sequences over \(\omega^d\); this holds more generally for any ordinal below \(\varepsilon_0\) along with an appropriate norm, see (Schmitz, 2014).
The previous chapter has established some very high complexity upper bounds on algorithms that rely on Dickson’s Lemma over $d$-tuples of natural numbers for termination. The Length Function Theorem shows that these bounds can be found in every level of the Grzegorczyk hierarchy when $d$ varies, which means that these bounds are Ackermannian when $d$ is part of the input.

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

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

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

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

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.
Chapter 3. Complexity Lower Bounds

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

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 \( M = (\text{Loc}, C, \Delta) \) where \( \text{Loc} = \{\ell_1, \ldots, \ell_p\} \) is a finite set of locations, \( C = \{c_1, \ldots, c_n\} \) is a finite set of counters, and \( \Delta \subseteq \text{Loc} \times OP(C) \times \text{Loc} \) 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:

\[
OP(C) \ni \text{op} ::= \quad \text{c}=0\? \quad /* \text{zero test} */ \quad | \quad \text{c}:=0 \quad /* \text{reset} */ \\
| \quad \text{c}>0\? \text{c}-- \quad /* \text{decrement} */ \quad | \quad \text{c}=\text{c}'\? \quad /* \text{equality test} */ \\
| \quad \text{c}++ \quad /* \text{increment} */ \quad | \quad \text{c}:=\text{c}' \quad /* \text{copy} */
\]

where \( \text{c}, \text{c}' \) are any two counters in \( C \). (We also allow a 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 \( M = (\text{Loc}, C, \Delta) \) is given under the form of transitions between its configurations. Formally, a configuration (written \( \sigma, \theta, \ldots \))
of \( M \) is a tuple \((\ell, a)\) with \( \ell \in \text{Loc} \) representing the “current” control location, and \( a \in \mathbb{N}^C \), a \( C \)-indexed vector of natural numbers representing the current contents of the counters. If \( C \) is some \( \{c_1, \ldots, c_n\} \), we often write \((\ell, a)\) 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. \( a = (0, \ldots, 0, c_k; 1, 0, \ldots, 0) \).

Regarding the behaviour induced by the rules from \( \Delta \), there is a transition (also called a step) \( \sigma \xrightarrow{\delta} \text{std} \sigma' \) if, and only if, \( \sigma \) is some \((\ell, a_1, \ldots, a_n)\), \( \sigma' \) is some \((\ell', a_1', \ldots, a_n')\), \( \Delta \ni \delta = (\ell, op, \ell') \) and either:

- \( op = c_k = 0 \) (zero test): \( a_k = 0 \), and \( a_i' = a_i \) for all \( i \neq k \), or
- \( op = c_k > 0 \) (decrement): \( a_k' = a_k - 1 \), and \( a_i' = a_i \) for all \( i \neq k \), or
- \( op = c_k ++ \) (increment): \( a_k' = a_k + 1 \), and \( a_i' = a_i \) for all \( i \neq k \), or
- \( op = c_k : = 0 \) (reset): \( a_k' = 0 \), and \( a_i' = a_i \) for all \( i \neq k \), or
- \( op = c_k = c_p \) (equality test): \( a_k = a_p \), and \( a_i' = a_i \) for all \( i = 1, \ldots, n \), or
- \( op = c_k : = c_p \) (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 \( \sigma \xrightarrow{\text{std}} \sigma' \), when \( \sigma \xrightarrow{\delta} \text{std} \sigma' \) for some \( \delta \in \Delta \). Chains \( \sigma_0 \xrightarrow{\text{std}} \sigma_1 \xrightarrow{\text{std}} \cdots \xrightarrow{\text{std}} \sigma_m \) of consecutive steps, also called runs, are denoted \( \sigma_0 \xrightarrow{\tau} \text{std} \sigma_m \), and also \( \sigma_0 \xrightarrow{\tau} \text{std} \sigma_m \) when \( m > 0 \). Steps may also be written more precisely under the form \( M \vdash \sigma \xrightarrow{\text{std}} \sigma' \) when several counter machines are at hand and we want to be explicit about where the steps take place.

For a vector \( a = (a_1, \ldots, a_n) \), or a configuration \( \sigma = (\ell, a) \), we let \( |a| = |\sigma| = \sum_{i=1}^{n} a_i \) denote its size. For \( N \in \mathbb{N} \), we say that a run \( \sigma_0 \xrightarrow{\text{std}} \sigma_1 \xrightarrow{\text{std}} \cdots \xrightarrow{\text{std}} \sigma_m \) 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”).

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 $M$:
\[(\ell, a_1, \ldots, a_n) \leq (\ell', a'_1, \ldots, a'_n) \iff \ell = \ell' \land a_1 \leq a_1' \land \cdots \land a_n \leq a_n'. \quad (3.1)\]

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

Now “lossy” steps, denoted $M \vdash \sigma \xrightarrow{\delta_{\text{lossy}}} \sigma'$, are given by the following definition:
\[
\sigma \xrightarrow{\delta_{\text{lossy}}} \sigma' \iff \exists \theta, \theta', (\sigma \geq \theta \land \theta \xrightarrow{\delta_{\text{std}}} \theta' \land \theta' \geq \sigma'). \quad (3.2)
\]

Note that reliable steps are a special case of lossy steps:
\[
M \vdash \sigma \rightarrow_{\text{std}} \sigma' \text{ implies } M \vdash \sigma \xrightarrow{\text{lossy}} \sigma'. \quad (3.3)
\]

### 3.1.4 Behavioural Problems on Counter Machines

We consider the following decision problems:

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

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

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

These problems are parameterized by the class of counter machines we consider and, more importantly, by the operational semantics that is assumed. Reachability and termination are decidable for lossy counter machines, i.e. counter machines assuming lossy steps, because they are well-structured. Observe that, for lossy machines, reachability and coverability coincide (except for runs of length 0). Coverability is often used to check whether a control location is reachable from some $\sigma_{\text{ini}}$. 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

The **Hardy hierarchy** $(H^\alpha: \mathbb{N} \to \mathbb{N})_{\alpha < \varepsilon_0}$ 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:
\[
H^0(n) \overset{\text{def}}{=} n, \quad H^{\alpha+1}(n) \overset{\text{def}}{=} H^\alpha(n + 1), \quad H^\lambda(n) \overset{\text{def}}{=} H^{\lambda n}(n). \quad (3.4)
\]
Observe that $H^1$ is the successor function, and more generally $H^\alpha$ is the $\alpha$th 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

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

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

$$H^\alpha(n) = H_\alpha(n) + n.$$  \hfill (3.6)

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

$$\text{Ack}(n) \overset{\text{def}}{=} F_\omega(n) = H^{\omega^\omega}(n) = H^{\omega^n}(n).$$  \hfill (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 $\alpha \subseteq \alpha'$, 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

$$\alpha_0; n_0 \rightarrow \alpha_1; n_1 \rightarrow \alpha_2; n_2 \rightarrow \cdots \rightarrow \alpha_\ell; n_\ell$$  \hfill (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 $H^{\alpha_\ell}(n_\ell)$ 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,

$$\omega^\omega; m \rightarrow \omega^m; m \rightarrow \omega^{m-1} \cdot m; m$$  \hfill (3.9)

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

Hardy computation
3.2.1 Encoding Hardy Computations

Ordinals below \( \omega^{m+1} \) 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 \( \omega^{m+1} \) as

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

(3.10)

Observe that ordinals below \( \omega^{m+1} \) and vectors in \( \mathbb{N}^{m+1} \) are in bijection through \( \alpha \).

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

\[
\langle a_m, \ldots, a_0 + 1; n \rangle \rightarrow \langle a_m, \ldots, a_0; n + 1 \rangle , 
\]

(D1)

\[
\langle a_m, \ldots, a_k + 1, 0, \ldots, 0; n \rangle \rightarrow \langle a_m, \ldots, a_k, n + 1, 0, \ldots, 0; n \rangle . 
\]

(D2)

The two rules (D1) and (D2) 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 \( \alpha(\mathbf{a}) \subseteq \alpha(\mathbf{a}') \) 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, \( \mathcal{H} \) terminates since \( \langle \mathbf{a}; n \rangle \rightarrow \langle \mathbf{a}'; n' \rangle \) implies \( \alpha(\mathbf{a}) > \alpha(\mathbf{a}') \). Furthermore, if \( \mathbf{a} \neq \mathbf{0} \), one of the rules among (D1) and (D2) can be applied to \( \langle \mathbf{a}; n \rangle \).

Hence for all \( \mathbf{a} \) and \( n \) there exists some \( n' \) such that \( \langle \mathbf{a}; n \rangle \rightarrow * \langle \mathbf{0}; n' \rangle \), and then \( n' = H^{\alpha(\mathbf{a})}(n) \). The reverse relation \( \mathcal{H}^{-1} \) terminates as well since, in a step \( \langle \mathbf{a}'; n' \rangle \rightarrow_{\mathcal{H}^{-1}} \langle \mathbf{a}; n \rangle \), 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 \( \mathcal{H} \) 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 \( M_H(m) = (\text{Loc}_H, C, \Delta_H) \), or \( M_H \) for short, with \( C = \{a_0, a_1, \ldots, a_m, n\} \). Its rules are defined pictorially in Figure 3.1: they implement \( \mathcal{H} \) as a loop around a central location \( \ell_H \), 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 \( \langle \mathbf{a}; n \rangle \rightarrow \mathcal{H} \langle \mathbf{a}'; n' \rangle \) then \( M_H \vdash (\ell_H, \mathbf{a}, n) \rightarrow_{\text{std}}^* (\ell_H, \mathbf{a}', n') \).
2. If $M_H \vdash (\ell_H, a, n) \rightarrow_{std}^* (\ell_H, a', n')$ then $H^\alpha(a)(n) = H^\alpha(a')(n')$.

3. If $M_H \vdash (\ell_H, a, n) \rightarrow_{lossy}^* (\ell_H, a', n')$ then $H^\alpha(a)(n) \geq H^\alpha(a')(n')$.

The rules (D1–D3) 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 $H^{-1}_H$ as a loop around a central location $\ell_{H^{-1}}$, as captured by Lemma 3.6. Note that $M_{H^{-1}}$ may deadlock if it makes the wrong guess as whether $a_i$ contains $n + 1$, but this is not a problem with the construction.

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

1. If $(a; n) \xrightarrow{H} (a'; n')$ then $M_{H^{-1}} \vdash (\ell_{H^{-1}}, a', n') \rightarrow_{std}^* (\ell_{H^{-1}}, a, n)$.

2. If $M_{H^{-1}} \vdash (\ell_{H^{-1}}, a, n) \rightarrow_{std}^* (\ell_{H^{-1}}, a', n')$ then $H^\alpha(a)(n) = H^\alpha(a')(n')$.

3. If $M_{H^{-1}} \vdash (\ell_{H^{-1}}, a, n) \rightarrow_{lossy}^* (\ell_{H^{-1}}, a', n')$ then $H^\alpha(a)(n) \geq H^\alpha(a')(n')$.

### 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 $\Delta$ so that any increment (resp. decrement) in the original counters is balanced by a corresponding decrement (resp. increment) on the new counter $B$,
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 $c_i$ 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 “$\ell, \ell' \in \text{Loc}$” 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 $M^b \vdash (\ell, B, a) \rightarrow^*_{\text{std}} (\ell', B', a')$ and $\ell, \ell' \in \text{Loc}$, then $B + |a| = B' + |a'|$. 

---

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

Figure 3.3: From $M$ to $M^b$ (schematically).
Observe that $M^b$ can only do what $M$ would do:

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

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

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

**Proof Idea.** The “$(\Rightarrow)$” direction is an immediate consequence of (3.3).

For the “$(\Leftarrow)$” direction, we consider the hypothesised run $M^b \vdash (\ell, B, a) = \sigma_0 \rightarrow_{\text{lossy}} \sigma_1 \rightarrow_{\text{lossy}} \cdots \rightarrow_{\text{lossy}} \sigma_n = (\ell', B', a')$. Coming back to (3.2), these lossy steps require, for $i = 1, \ldots, n$, some reliable steps $\theta_{i-1} \rightarrow_{\text{std}} \theta_i$ with $\sigma_{i-1} \geq \theta_{i-1}$ and $\theta_i \geq \sigma_i$, and hence $|\theta'_i| \geq |\theta_i|$ for $i < n$. Combining with $|\theta_{i-1}| = |\theta'_i|$ (by Lemma 3.7), and $|\sigma_0| = |\sigma_n|$ (from the assumption that $B + |a| = B' + |a'|$), 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 $M^b \vdash (\ell, B, 0) \rightarrow^*_{\text{lossy}} (\ell', B', a)$ with $\ell, \ell' \in \text{Loc}$. Then:

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


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

3. $M^b$ has a reliable run ($\ell_{ini}, Ack(m), 0) \rightarrow^*_{std} (\ell_{fin}, Ack(m), 0)$.

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

5. $M$ has a reliable run ($\ell_{ini}, 0) \rightarrow^*_{std} (\ell_{fin}, 0)$ 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 ($\ell_H, 1, 0, m, 0) \rightarrow^*_{lossy} (\ell_{H^{-1}}, 1, 0, m, 0)$. This run must go through $M^b$ and be in three parts of the following form:

\[
(\ell_H, 1, 0, m, 0) \xrightarrow{\Delta_{H^{-1}}^{*}_{lossy}} (\ell_{H^{-1}}, a_n: x, 0) \quad \text{(starts in $M_H$)}
\]
\[
\quad \rightarrow_{lossy} (\ell_{ini}, \ldots, B, 0) \xrightarrow{\Delta_{H^{-1}}^{*}_{lossy}} (\ell_{fin}, \ldots, B', c) \quad \text{(goes through $M^b$)}
\]
\[
\quad \rightarrow_{lossy} (\ell_{H^{-1}}, a', x', \ldots) \xrightarrow{\Delta_{H^{-1}}^{*}_{lossy}} (\ell_{H^{-1}}, 1, 0, m, 0). \quad \text{(ends in $M_{H^{-1}}$)}
\]

The first part yields $H^{a(x, 0)}(m) \geq H^{a(x)}(x)$ (by Lemma 3.5.3), the third part $H^{a(x')}(x') \geq H^{a(x)}(x)$ (by Lemma 3.6.3), and the middle part $B \geq B' + |c|$ (by Corollary 3.1.1). Lossiness further implies $x \geq B$, $B' \geq x'$ and $a \geq a'$. Now, the only way to reconcile $H^{a(x)}(x) \leq H^{a(x, 0)}(m) = Ack(m) \leq H^{a(x')}(x')$, $a' \leq a$, $x' \leq x$, and the monotonicity of $F$ (Lemma 3.4) is by concluding $x = B = B' = x' = Ack(m)$ and $c = 0$. Then the middle part of the run witnesses $M^b \models (\ell_{ini}, Ack(m), 0) \rightarrow^*_{lossy} (\ell_{fin}, Ack(m), 0)$.

- “2 ⇒ 5” is Corollary 3.1.1.
- “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 $M_{H^{-1}}$. 

---

**Figure 3.4:** Constructing $M(m)$ from $M^b$, $M_H$ and $\Delta H$. 

---

**Chapter 3. Complexity Lower Bounds**
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.

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 \( c = 0 \) with a corresponding reset \( c := 0 \). 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. \( M \vdash \sigma \rightarrow_{std} \sigma' \) implies \( R(M) \vdash \sigma \rightarrow_{std} \sigma' \).
2. \( R(M) \vdash \sigma \rightarrow_{std} \sigma' \) implies \( M \vdash \sigma \rightarrow_{lossy} \sigma' \).

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 \((\ell_H, a_m; 1, 0, n;m, 0) \rightarrow_{std}^* (\ell_{H-1}; 1, 0, m, 0)\).
2. \( R(M(m)) \) has a reliable run \((\ell_H, 1, 0, m, 0) \rightarrow^{*} \text{std} \theta \geq (\ell_{H^{-1}}, 1, 0, m, 0) \).

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

Proof. For 1 \( \Rightarrow \) 3: The reliable run in \( R(M(m)) \) gives a lossy run in \( M(m) \) (Lemma 3.13), 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), 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), hence also a lossy run in \( M(m) \) that reaches exactly \((\ell_{H^{-1}}, 1, 0, m, 0)\) (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). \( \square \)

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 \((M \text{ and } M^b \text{ already are Minsky machines})\) 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. Second, 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 \( M = (\text{Loc}, C, \Delta) \) a new machine \( M' = (\text{Loc}', C', \Delta') \) such that:

1. \( \text{Loc}' \) is \( \text{Loc} \) plus some extra “auxiliary” locations,
2. \( C' = C + \{\text{aux}\} \) is \( C \) extended with one extra counter,

3. \( M' \) only uses zero-tests, increments and decrements, hence it is a Minsky machine,

4. For any \( \ell, \ell' \in \text{Loc} \) and vectors \( c, c' \in \mathbb{N}^C \), the following holds:

\[
M \vdash (\ell, c) \rightarrow^{*}_\text{std} (\ell', c') \iff M' \vdash (\ell, c, 0) \rightarrow^{*}_\text{std} (\ell', c', 0), \quad (3.11)
\]
\[
M \vdash (\ell, c) \rightarrow^{*}_\text{lossy} (\ell', c') \iff M' \vdash (\ell, c, 0) \rightarrow^{*}_\text{lossy} (\ell', c', 0). \quad (3.12)
\]

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 \( \text{aux} \) 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 “\( \Rightarrow \)” direction. Its proof is done with the help of the following observations:

- \( c - c' \) is a numerical invariant in \( l \), and also in \( l' \),
- \( c + \text{aux} \) is a numerical invariant in \( l \), and also in \( l' \),
- when \( M' \) moves from \( \ell_0 \) to \( l \), \( \text{aux} \) contains 0; when it moves from \( l \) to \( l' \), both \( c \) and \( c' \) contain 0; when it moves from \( l' \) to \( \ell_1 \), \( \text{aux} \) contains 0.

![Figure 3.5: From \( M \) to \( M' \): eliminating equality tests.](image)
Then we also need the less standard notion of correctness from (3.12) for this transformation. The “⇐” direction is proved with the help of the following observations:

- \( c - c' \) can only decrease during successive visits of \( l \), and also of \( l' \),
- \( c + \text{aux} \) can only decrease during successive visits of \( l \), and also of \( l' \),
- when \( M' \) moves from \( \ell_0 \) to \( l \), aux contains 0; when it moves from \( l \) to \( l' \), both \( c \) and \( c' \) contain 0; when it moves from \( l' \) to \( \ell_1 \), aux contains 0.

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

### 3.6 HARDNESS FOR TERMINATION

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

Basically, \( M^b \) now uses two copies of the initial budget. One copy in \( B \) works as before: its purpose is to ensure that losses will be detected by a budget imbalance as in Lemma 3.10. The other copy, in a new counter \( T \), is a time limit that is 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 \( H \rightarrow^* \) and \( H \rightarrow^* 1 \) terminate, see Section 3.2), we now have a new \( M(m) \) that always terminates when started in \( \ell_H \) 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 \((\ell_H, 1, 0, n: m, 0) \rightarrow^*_{\text{lossy}} \theta \geq (\ell_{H-1}, 1, 0, m, 0)\).

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

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

Finally, we add a series of \( m + 1 \) transitions that leave from \( \ell_{H-1} \), and check that \( \sigma_{\text{goal}} \overset{\text{def}}{=} (\ell_{H-1}, 1, 0, m, 0) \) is covered, i.e., that \( a_m \) contains at least 1 and \( n \) at least \( m \). If this succeeds, one reaches a new location \( \ell_\omega \), 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 \( \sigma_{\text{goal}} \), i.e., if, and only if, \( M \) has a reliable run of length at most \( \text{Ack}(m) \), which is an Ackermann-hard problem.

**Exercises**

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

**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

\[
c_1 += c_2 ; c_2 := 0.
\]

/* transfer \( c_2 \) to \( c_1 \) */

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 \( M \vdash \sigma \rightarrow_{\text{inc}} \sigma' \), and defined by:

\[
\sigma \overset{\delta_{\text{inc}}} \rightarrow_{\text{inc}} \sigma' \overset{\text{def}}{=} \exists \theta, \theta', (\sigma \leq \theta \land \theta \overset{\delta_{\text{std}}} \rightarrow_{\text{std}} \theta' \land \theta' \leq \sigma').
\]  

(*)

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

\[
M \vdash \sigma_1 \rightarrow_{\text{std}} \sigma_2 \iff \text{if } M^{-1} \vdash \sigma_2 \rightarrow_{\text{std}} \sigma_1. 
\]

What does it entail for lossy runs of \( M \) and incrementing runs of \( M^{-1} \)? Conclude that reachability for incrementing counter machines is Ackermannian.
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 Lazic, 2009; Jurdzinski and Lazic, 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).
IDEALS

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.

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.

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 $\text{Down}$ denotes the set of its downward-closed subsets, with typical elements $D, D', \ldots$

### 4.1.1 Prime Decompositions of Order-Closed Subsets

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

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

2. Similarly, a nonempty $D \in \text{Down}(X)$ 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. $U \in \text{Up}(X)$ is prime if, and only if, for any $U_1, \ldots, U_n \in \text{Up}, U = U_1 \cup \cdots \cup U_n$ implies $U = U_i$ for some $i$.

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

**Proof.** We prove 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 $U_1, U_2 \in \text{Up}$. Then, letting $U'_i = U_i \cap U$ 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 $U \subseteq U_i$. \qed

We say that a collection $\{P_1, \ldots, P_n\}$ of up (resp. down) primes is a (finite) **prime decomposition of** $U \in \text{Up}(X)$ (resp., of $D \in \text{Down}(X)$) 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 $U \in \text{Up}(X)$ is a finite union of up primes.

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

**Proof of 2.** By well-founded induction on $D$. For this, recall that $(\text{Down}(X), \subseteq)$ 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. \qed
4.1. Ideals of WQOs

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. \( D \)).

**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 \( \ell_j \) such that \( P'_j \subseteq P_{\ell_j} \). These inclusions are equalities since \( P_i \subseteq P'_{k_i} \subseteq P_{\ell_k} \) requires \( i = \ell_k \) 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 \( x_2 \geq x \).

**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 \( X \).

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 \( \text{Idl}(X) \) 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: \( \text{Idl}(A) = \{\downarrow a \mid a \in A\} \).

In the case of \( \mathbb{N} \), the ideals are exactly the principal ideals and the whole set itself: \( \text{Idl}(\mathbb{N}) = \{\downarrow n \mid n \in \mathbb{N}\} \cup \{\mathbb{N}\} \).
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, \(P \subseteq (P \setminus \uparrow x_1) \cup (P \setminus \uparrow x_2)\). But \(P \setminus \uparrow x_i\) is downward closed for both \(i = 1, 2\), so \(P\), being prime, is included in one \(P \setminus \uparrow x_i\). 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 \(x_i \not\in \bigcup_{j\neq i} P_j\). 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 \(\mathcal{J} \subseteq \text{Idl}(X)\) be such that every \(D \in \text{Down}(X)\) is a finite union of ideals from \(\mathcal{J}\). Then \(\mathcal{J} = \text{Idl}(X)\).

*Proof.* Let \(I \in \text{Idl}(X)\). Since \(I \in \text{Down}(X)\), \(I = J_1 \cup \cdots \cup J_n\) 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). \(\text{Up}(X), \text{Down}(X)\) and \(\text{Idl}(X)\) are countable when \(X\) is a countable wqo.
4.2. Effective Well Quasi-Orders and Ideals

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

(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 $Idl(X)$, 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:

$$U \subseteq U' \text{ iff } \forall i : \exists j : F_i \subseteq F'_j.$$  \hfill (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:

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

Note also that the chosen representation for $Up(X)$ and $Down(X)$ 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 $X$, $Idl(X)$, and consequently also for $Up(X)$ and $Down(X)$, our next definition lists effectiveness assumptions on more set-theoretical operations.

**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: \( \neg U = \bigcup_{i=1}^{n} \uparrow x_i = \bigcap_{i=1}^{n} \neg \uparrow x_i \), 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 \( x \not\in D \) 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.

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
4.3 Some Ideally Effective WQOs

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 $\omega$. A natural data structure for $Idl(\mathbb{N})$ is thus to use the number “$n$” to denote $\downarrow n$ and the symbol “$\omega$” to denote $\mathbb{N}$.

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;

1From the ordinal viewpoint where $\alpha$ 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 $\omega$.

The same easy techniques can be used to show that any finite $\mathbb{Q}$ 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 $I_i$ for $i = 1, 2$ and $I$ an ideal of $A_i$. Thus $\text{Idl}(A_1 + A_2) \equiv \text{Idl}(A_1) + \text{Idl}(A_2)$.

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 $I_1 \in \text{Idl}(A_1)$ and $I_2 \in \text{Idl}(A_2)$. Thus $\text{Idl}(A_1 \times A_2) \equiv \text{Idl}(A_1) \times \text{Idl}(A_2)$.

Furthermore, if $A_1$ and $A_2$ are ideally effective then $A_1 \times A_2$ is.

**Proof Sketch.** We leave the characterisation of $\text{Idl}(A_1 \times A_2, \leq \times)$ 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 $X \equiv A_1 \times A_2$. Clearly, elements of $X$, and ideals of $\text{Idl}(X)$, 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

$$\neg(F_1 \times F_2) = (\neg F_1) \times A_2 \cup A_1 \times (\neg F_2).$$

(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.
4.3. Some 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 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:

\[
\langle i, a \rangle \leq_{A_1 +_{\text{lex}} A_2} \langle j, b \rangle \iff i < j \text{ or } i = j \land a \leq_{A_1} b, \tag{4.4}
\]

\[
\langle a, b \rangle \leq_{A_1 \times_{\text{lex}} A_2} \langle a', b' \rangle \iff a <_{A_1} a' \text{ or } a \equiv_{A_1} a' \land b \leq_{A_2} b'. \tag{4.5}
\]

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 $(\Sigma^*, \leq_\ast)$. 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, $\Sigma^*$ 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 $(ab)^\ast$. 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 $(a + b)^\ast$, 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, $\Gamma^\ast$ is an ideal of $\Sigma^*$.

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., $(b + c + d)^\ast$, all words with no $b$’s, i.e., $(a + c + d)^\ast$, and all words possibly having $a$’s and $b$’s but where the $a$’s are after the $b$’s, i.e., $(b + c + d)^\ast(a + c + d)^\ast$. Actually, that third part contains the earlier two and we can summarise with

\[
D \overset{\text{def}}{=} \Sigma^* \setminus \uparrow ab = (b + c + d)^\ast(a + c + d)^\ast. \tag{4.6}
\]

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

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

\[\text{See also Exercise 1.4}\]
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 $I$ 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 $(X, \leq)$.

**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 $D \in \text{Down}(X)$, 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 $\varepsilon$.

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 $\text{Idl}(X^*)$, 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

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

(4.7) generalising the earlier example in (4.6). We note that any $(X \setminus \uparrow_{X^*} x_i)^*$ is indeed $D^*$ for some $D \in \text{Down}(X)$. 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

\[ D^* \cdot P \cap D'^* \cdot P' = (D \cap D')^* \cdot [D^* \cdot P \cap P'] \cup (D \cap D')^* \cdot [P \cap D'^* \cdot P'], \]

\[(I + \varepsilon) \cdot P \cap (I' + \varepsilon) \cdot P' = [(I \cap I') + \varepsilon] \cdot [P \cap P'], \tag{4.8} \]

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 \(\leq_\ast\) is easy once we know, by assumption, how to compare elements occurring in sequences.

The data structure for \(Idl(X^*)\) 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.

\[ I + \varepsilon \subseteq I' + \varepsilon \quad \text{iff} \quad I \subseteq I' \]
\[ I + \varepsilon \subseteq D^* \quad \text{iff} \quad I \subseteq D \]
\[ D^* \subseteq I + \varepsilon \quad \text{iff} \quad D \subseteq \emptyset \]
\[ D^* \subseteq D'^* \quad \text{iff} \quad D \subseteq D' \]

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

\[ \varepsilon \subseteq P \quad \text{always} \]
\[ (I + \varepsilon).P \subseteq \varepsilon \quad \text{never} \]
\[ D^* \cdot P \subseteq \varepsilon \quad \text{iff} \quad D = \emptyset \land P \subseteq \varepsilon \]
\[ (I + \varepsilon).P \subseteq (I' + \varepsilon).P' \quad \text{iff} \quad [I \subseteq I' \land P \subseteq P'] \lor [(I + \varepsilon).P \subseteq P'] \]
\[ (I + \varepsilon).P \subseteq D^* \cdot P' \quad \text{iff} \quad [I \subseteq D \land P \subseteq P'] \lor [(I + \varepsilon).P \subseteq P'] \]
\[ D^* \cdot P \subseteq (I' + \varepsilon).P' \quad \text{iff} \quad [D = \emptyset \land P \subseteq (I + \varepsilon).P'] \lor [D^* \cdot P \subseteq P'] \]
\[ D^* \cdot P \subseteq D'^* \cdot P' \quad \text{iff} \quad [D \subseteq D' \land P \subseteq D'^* \cdot P'] \lor [D^* \cdot P \subseteq P'] \]

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 $(Idl(X^*), \subseteq)$. We continue with

(CF): Equation (4.7) can be turned into an algorithm for expressing the complement $X^* \setminus \uparrow w$ of any principal filter $\uparrow w$ as a union of ideals. Here the assumption that $(X, \leq)$ is ideally effective is used to express $X \setminus x$ 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 $X \in \text{Down}(X)$. 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 $\langle S, \rightarrow, \leq \rangle$ 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
Post computations as defined in (1.2) on page 8 can be carried over ideals. More precisely, if we define

\[ Post_\exists(I) \overset{\text{def}}{=} \{ s_2 \in S \mid \exists s_1 \in I, s_1 \rightarrow s_2 \}, \]

then we require the canonical decomposition of \( \downarrow Post_\exists(I) \) to be computable for all \( I \in Idl(S) \).

**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

\[ D_0 \overset{\text{def}}{=} \downarrow s \quad D_{n+1} \overset{\text{def}}{=} \downarrow Post_\exists(D_n) \]

until \( t \in D_n \). These operations are effective using (PI), (ID), and the computability of \( \downarrow Post_\exists(D) = \downarrow Post_\exists(I_1) \cup \cdots \cup \downarrow Post_\exists(I_\ell) \) for any downward-closed \( D = I_1 \cup \cdots \cup I_\ell \). Regarding correctness, if \( s = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n \geq t \), 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 \( s'_{n-1} \rightarrow s_n \geq t \), 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 \( s'_{n-2} \rightarrow s_{n-1} \geq s'_{n-1} \), etc. We can therefore exhibit a sequence \( s_0 \geq s'_0 \rightarrow s_1 \geq s'_1 \rightarrow s_2 \geq s'_2 \rightarrow \cdots \rightarrow s_n \geq t \) 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. \( \square \)

**Proof of co-recursivity.** Our second procedure enumerates downward-closed sets \( D \in \text{Down}(S) \) until we find a (forward) inductive invariant, i.e. \( D \) such that \( \downarrow Post_\exists(D) \subseteq D \), \( s \in D \), and \( t \not\in D \). All these operations are effective using (IR), (PI), (CI), and the computability of the canonical decomposition of \( \downarrow Post_\exists(D) \).

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

\[
\downarrow Post_\exists (u) = \{ u + a \mid a \in A \}
\]

(4.11)

where \( \omega + k = \omega \) for all \( k \in \mathbb{Z} \). This illustrates the fact that \( Post_\exists (I) \) 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

\[
D_0 \overset{\text{def}}{=} S \setminus \uparrow t \quad D_{n+1} \overset{\text{def}}{=} D_n \cap Pre_\forall (D_n),
\]

(4.12)

where

\[
Pre_\forall (D) \overset{\text{def}}{=} \{ s_1 \in S \mid \forall s_2 \in S, s_1 \to s_2 \text{ implies } s_2 \in D \}.
\]

(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

\[
Pre_\forall (D) = S \setminus Pre_\exists (S \setminus D)
\]

(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 \( Pre_\forall (D_0) \) over the wqo \( (S, \leq) \), and \( s \) covers \( t \) if and only if \( s \not\in Pre_\forall (D_0) \), which can be decided by (IM). Hence

Proposition 4.26. \textbf{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 \( D_0 \supseteq D_1 \supseteq \cdots \supseteq D_n = D_{n+1} \) 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 \( D_k \supseteq D_{k+1} \) 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.

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

\[
D_{k+1} = D_k \setminus \{ s_1 \in D_k \mid \exists s_2 \not\in D_k . s_1 \rightarrow s_2 \}. \tag{4.15}
\]

Moreover, noting \( D_{k-1} \uparrow t \uparrow \uparrow t \), \( s_2 \) 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

\[
E_1 = \uparrow t , \quad E_k = \{ s_1 \in D_k \mid \exists s_2 \in E_{k-1} . s_1 \rightarrow s_2 \}. \tag{4.16}
\]

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 \( \downarrow \text{Post}_\exists (I_{k+1}) \) 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 \( \downarrow \text{Post}_\exists (I_{k+1}) \cap E_k \neq \emptyset \) by (4.16), and there exists \( J \) from the canonical decomposition of \( \downarrow \text{Post}_\exists (I_{k+1}) \) such that \( J \cap E_k \neq \emptyset \).

Since \( I_{k+1} \subseteq D_{k+1} \subseteq \text{Pre}_\exists D_k \) by (4.12), we also know that \( \text{Post}_\exists (I_{k+1}) \subseteq D_k \). 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.

\( \omega \)-MONOTONICITY. Let us turn now to the specific case of VAS. Our instance of Cover is thus a \( d \)-dimensional VAS \( \langle x_0, A \rangle \) and a target configuration \( t \) 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 \( u \) in \( (\mathbb{N} \cup \{ \omega \})^d \), its \( \omega \)-set is the subset \( \omega(u) \) of \( \{1, \ldots, d\} \) such that \( u(i) = \omega \) if and only if \( i \in \omega(u) \). We say that a descending chain \( D_0 \supseteq D_1 \supseteq \cdots \supseteq D_t \) of downward-closed subsets of \( \mathbb{N}^d \) is \( \omega \)-monotone if for all \( 0 \leq k < \ell - 1 \) and all proper ideals \( v_{k+1} \) in the canonical decomposition of \( D_k+1 \), there exists a proper ideal \( v_k \) in the canonical decomposition of \( D_k \) such that \( \omega(v_{k+1}) \subseteq \omega(v_k) \).

**Lemma 4.28** (VAS Descending Chains are \( \omega \)-Monotone). The descending chains computed by the dual backward coverability algorithm for VAS are \( \omega \)-monotone.
Proof. Let $D_0 \supseteq D_1 \supseteq \cdots \supseteq D_\ell$ be the descending chain computed for a VASS. Suppose $0 \leq k < \ell - 1$ and $v_{k+1}$ is a proper ideal in the decomposition of $D_{k+1}$. By Lemma 4.27 and (4.11), there exists a proper ideal $v_k$ in the decomposition of $D_k$ such that $v_{k+1} + a \leq v_k$ for some $a$ in $A$. We conclude that $\omega(v_{k+1}) \subseteq \omega(v_k)$.

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 $(\mathbb{N} \cup \{\omega\})^d$: $\|v\| \overset{\text{def}}{=} \max_{1 \leq i \leq d} |v(i)| < \omega v(i)$, and the maximal norm $\|D\| \overset{\text{def}}{=} \max \|I_j\|$ 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 $\|A\| \overset{\text{def}}{=} \max_{a \in A} \|a\|$; we rely for this on ????

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

Proof. The fact that $\|D_0\| \leq \|t\|$ 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: $\|D_{k+1}\| \leq \|D_k\| + \|A\|$.

UPPER BOUND. We are now in position to state a refinement of ?? for $\omega$-monotone controlled descending chains. For a control function $g : \mathbb{N} \rightarrow \mathbb{N}$, define the function $\tilde{g} : \mathbb{N}^2 \rightarrow \mathbb{N}$ by induction on its first argument:

$$
\tilde{g}(0, n) \overset{\text{def}}{=} 1, \quad \tilde{g}(m + 1, n) \overset{\text{def}}{=} \tilde{g}(m, n) + (\tilde{g}(m, n)(n) + 1)^{m+1}.
$$

(4.17)

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

Proof. Note that $D_{\ell}$ 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 $v_k$, and add one to get a bound on $\ell$. We are going to establish by induction on $d - |I|$ that if $v_k$ is a proper ideal from the canonical decomposition of $D_k$ and its $\omega$-set is $I$, then $k < \tilde{g}(d - |I|, n)$, which by monotonicity of $\tilde{g}$ in its first argument entails $k < \tilde{g}(d, n)$ as desired.

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

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

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

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

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

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 \( \omega \)-set and norm bounded by \( \|D_{k+1}\| \).

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

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

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

**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 \( \omega \), 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 $(A_1 + A_2, \leq_+)\leq_+$ is effective when $(A_1, \leq_1)$ and $(A_2, \leq_2)$ are.

3. Further show that $(A_1 + A_2, \leq_+)\leq_+$ 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 $I_1 \in Idl(A_1, \leq_1)$ and $I_2 \in Idl(A_2, \leq_2)$.

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 $Idl(A_1 +_{lex} A_2, \subseteq)$ is isomorphic to $Idl(A_1, \subseteq) +_{lex} Idl(A_2, \subseteq)$.

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

3. Show that $A_1 +_{lex} A_2$ 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 $A_1 \times_{lex} A_2$?

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

3. Show that $A_1 \times_{lex} A_2$ 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 $\alpha$ 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 $(X^*, \leq_\ast)$). We complete the proof of Proposition 4.22.
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 $(X^*, \leq^*)$.

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 $\downarrow^*_\leq x$, 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 $(\mathbb{N}^k, \leq_k)$ extend results from (Karp and Miller, 1969) and (Valk and Jantzen, 1983). 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 (Lažić and Schmitz, 2015a) (Lažić and Schmitz, 2016)
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 Cichoki and Wainer (1983) and Cichoki 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 \((\mathcal{R}_\alpha)\). 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 \((F_\alpha)\) or the Hardy functions \((H^\alpha)\), 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 \(\varepsilon_0\), which allows to write any ordinal as an ordinal term \(\alpha\) 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 $\Omega$ of all ordinal terms. Each ordinal term $\alpha$ is a syntactic object, and denotes a unique ordinal $\text{ord}(\alpha)$ 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 $\text{ord}(\alpha') \leq \text{ord}(\alpha)$. 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 $\beta_i$ 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 $\beta_i$ in strict form—we call the $c_i$'s coefficients. Terms $\alpha$ in CNF are in bijection with their denoted ordinals $\text{ord}(\alpha)$. We write $\text{CNF}(\alpha)$ for the set of ordinal terms $\alpha' < \alpha$ in CNF; thus $\text{CNF}(\varepsilon_0)$ is a subset of $\Omega$ in our view. Having a richer set $\Omega$ will be useful later in Section A.8.

1 We write $\omega^0$ for $\omega^0_0$ and $\alpha \cdot n$ for $\alpha + \cdots + \alpha$. We work modulo associativity $((\alpha + \beta) + \gamma = \alpha + (\beta + \gamma))$ and idempotence $(\alpha + 0 = \alpha = 0 + \alpha)$ of $+$. An ordinal term $\alpha$ of form $\gamma + 1$ is called a successor ordinal term. Otherwise, if not 0, it is a limit ordinal term, usually denoted $\lambda$. Note that a $\text{ord}(0) = 0$, $\text{ord}(\alpha + 1)$ is a successor ordinal, and $\text{ord}(\lambda)$ is a limit ordinal if $\lambda$ 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 $\lambda$ in $\Omega$, 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 $\lambda$ is a limit.

A standard way of obtaining fundamental sequences with good properties for every limit ordinal term $\lambda$ is to fix a particular sequence $(\omega_x)_{x < \omega}$ for $\omega$ and to define
\[
(\gamma + \omega^{\beta+1})_x \overset{\text{df}}{=} \gamma + \omega^\beta \cdot \omega_x, \quad (\gamma + \omega^\lambda)_x \overset{\text{df}}{=} \gamma + \omega^\lambda x .
\] (A.1)

We assume $\omega_x$ 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 $s(x) = x + 1$.richer ordinal notations can be designed, notably the structured ordinals of [Dennis-Jones and Wainer (1984); Fairtlough and Wainer (1992)] below $\varepsilon_0$, and of course richer notations are required in order to go beyond $\varepsilon_0$.1
We will see in Section A.6 how different choices for \( \omega_x \) 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 \((x\text{-indexed})\) predecessor \( P_x(\alpha) < \alpha \) of an ordinal \( \alpha \neq 0 \) in \( \Omega \) as

\[
P_x(\alpha + 1) \overset{\text{def}}{=} \alpha, \quad P_x(\lambda) \overset{\text{def}}{=} P_x(\lambda_x). \tag{A.2}
\]

**Lemma A.1.** Assume \( \alpha > 0 \) in \( \Omega \). 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 \( \alpha \). For the successor case \( \alpha = \beta + 1 \), this goes

\[
P_x(\gamma + \beta + 1) \overset{(A.2)}{=} \gamma + \beta \overset{(A.2)}{=} \gamma + P_x(\beta + 1).
\]

For the limit case \( \alpha = \lambda \), this goes

\[
P_x(\gamma + \lambda) \overset{(A.2)}{=} P_x((\gamma + \lambda), \overset{(A.1)}{=} P_x(\gamma + \lambda_x) \overset{ih}{=} \gamma + P_x(\lambda_x) \overset{(A.2)}{=} \gamma + P_x(\lambda). \tag*{□}
\]

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

\[
P_x(\omega^{\beta+1}) \overset{(A.2)}{=} P_x((\omega^{\beta+1}), \overset{(A.1)}{=} P_x(\omega^\beta \cdot \omega_x) \overset{(A.3)}{=} \omega^\beta \cdot (\omega_x - 1) + P_x(\omega^\beta)
\]

\[
\overset{(A.2)}{=} \omega^{P_x(\beta+1)} \cdot (\omega_x - 1) + P_x(\omega^{P_x(\beta+1)}).
\]

For the limit case \( \alpha = \lambda \), this goes

\[
P_x(\omega^\lambda) \overset{(A.2)}{=} P_x((\omega^\lambda), \overset{(A.1)}{=} P_x(\omega^{\lambda_x}) \overset{ih}{=} \omega^{P_x(\lambda_x)} \cdot (\omega_x - 1) + P_x(\omega^{P_x(\lambda_x)})
\]

\[
\overset{(A.2)}{=} \omega^{P_x(\lambda)} \cdot (\omega_x - 1) + P_x(\omega^{P_x(\lambda)}). \tag*{□}
\]

### 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)):

\[
\alpha \prec_x \alpha + 1, \quad \lambda_x \prec_x \lambda. \tag{A.5}
\]

In particular, using induction on \( \alpha \), one immediately sees that

\[
0 \prec_x \alpha, \tag{A.6}
\]

\[
P_x(\alpha) \prec_x \alpha. \tag{A.7}
\]
The inductive definition of \( \prec_x \) implies
\[
\alpha’ \prec_x \alpha \iff \begin{cases} 
\alpha = \beta + 1 \text{ is a successor and } \alpha’ \prec_x \beta, \\
\alpha = \lambda \text{ is a limit and } \alpha’ \preceq_x \lambda.
\end{cases}
\] (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 \( \omega_x + 1 \npreceq_x \omega \), although \( \text{ord}(\omega_x + 1) \) is by definition a finite ordinal, smaller than \( \text{ord}(\omega) \).

The \( \prec_x \) relations are linearly ordered themselves
\[
\prec_0 \subseteq \cdots \subseteq \prec_x \subseteq \prec_{x+1} \subseteq \cdots
\] (A.9)
and, over terms in CNF, \(<\) can be recovered by
\[
\left( \bigcup_{x\in\mathbb{N}} \prec_x \right) = <.
\] (A.10)

We will soon prove these results in \textbf{Corollary A.4} and \textbf{Lemma A.5}, but we need first some basic properties of \( \prec_x \).

\textbf{Lemma A.2.} For all \( \alpha, \alpha’, \gamma \) in \( \Omega \) and all \( x \) in \( \mathbb{N} \)
\[
\alpha’ \prec_x \alpha \text{ implies } \gamma + \alpha’ \preceq_x \gamma + \alpha, \quad \text{(A.11)}
\]
\[
\omega_x > 0 \text{ and } \alpha’ \prec_x \alpha \text{ imply } \omega^{\alpha’} \prec_x \omega^\alpha. \quad \text{(A.12)}
\]

\textbf{Proof.} All proofs are by induction over \( \alpha \) (NB: the case \( \alpha = 0 \) is impossible).
\textbf{(A.11):} For the successor case \( \alpha = \beta + 1 \), this goes through
\[
\alpha’ \prec_x \beta + 1 \text{ implies } \alpha’ \preceq_x \beta \quad \text{(by (A.8))}
\]
\[
\text{implies } \gamma + \alpha’ \preceq_x \gamma + \beta \quad \text{(A.5)} \quad \text{implies } \gamma + \beta \preceq_x \gamma + 1. \quad \text{(by ind. hyp.)}
\]
For the limit case \( \alpha = \lambda \), this goes through
\[
\alpha’ \prec_x \lambda \text{ implies } \alpha’ \preceq_x \lambda_x \quad \text{(by (A.8))}
\]
\[
\text{implies } \gamma + \alpha’ \preceq_x \gamma + \lambda_x \text{ (A.1)} \quad \text{implies } \gamma + \lambda \preceq_x \gamma + \lambda. \quad \text{(by ind. hyp.)}
\]
\textbf{(A.12):} For the successor case \( \alpha = \beta + 1 \), we go through
\[
\alpha’ \prec_x \beta + 1 \text{ implies } \alpha’ \preceq_x \beta \quad \text{(by (A.8))}
\]
\[
\text{implies } \omega^{\alpha’} \preceq_x \omega^\beta = \omega^\beta + 0 \quad \text{(by ind. hyp.)}
\]
\[
\text{implies } \omega^{\alpha’} \preceq_x \omega^\beta + \omega^\beta \cdot (\omega_x - 1) \quad \text{(by equations (A.6) and (A.11))}
\]
\[
\text{implies } \omega^{\alpha’} \preceq_x \omega^\beta \cdot \omega_x = (\omega^\beta+1)_x \preceq_x \omega^\beta+1. \quad \text{(A.5)}
\]
For the limit case \( \alpha = \lambda \), this goes through

\[
\alpha' \prec_x \lambda \quad \text{implies} \quad \alpha' \prec_x \lambda x
\]

(by (A.8))

implies \( \omega^{\alpha'} \prec_x \omega^{\lambda x} \) \( \xrightarrow{\text{(A.1)}} \) \( \omega^{\lambda} \) \( \prec_x \omega^{\lambda} \). \( \text{(by ind. hyp.)} \)

Lemma A.2 shows that \( \prec_x \) is left congruent for + and congruent for \( \omega \)-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 \( \prec_x \subseteq \prec \) for terms in CNF(\( \varepsilon_0 \)).

**Lemma A.3.** Let \( \lambda \) be a limit ordinal in \( \Omega \) and \( x < y \) in \( \mathbb{N} \). Then \( \lambda_x \prec y \lambda_y \), and if furthermore \( \omega_x > 0 \), then \( \lambda_x \prec x \lambda_y \).

**Proof.** By induction over \( \lambda \). Write \( \omega_y = \omega_x + z \) for some \( z \geq 0 \) by monotonicity of \( s \) (recall that \( \omega_x \) and \( \omega_y \) are in \( \mathbb{N} \)) and \( \lambda = \gamma + \omega^\alpha \) with \( 0 < \alpha \).

If \( \alpha = \beta + 1 \) is a successor, then \( \lambda_x = \gamma + \omega^\beta \cdot \omega_x \prec y \gamma + \omega^\beta \cdot \omega_x + \omega^\beta \cdot z \) by (A.11) since \( 0 \prec x \omega^\beta \cdot z \). We conclude by noting that \( \lambda_y = \gamma + \omega^\beta \cdot (\omega_x + z) \); the same arguments also show \( \lambda_x \prec x \lambda_y \).

If \( \alpha \) is a limit ordinal, then \( \alpha_x \prec y \alpha_y \) by ind. hyp., hence \( \lambda_x = \gamma + \omega^{\alpha_x} \prec y \gamma + \omega^{\alpha_y} = \lambda_y \) by (A.12) (applicable since \( \omega_y \geq y > x \geq 0 \)) and (A.11). If \( \omega_x > 0 \), then the same arguments show \( \lambda_x \prec x \lambda_y \).

\( \square \)

Now, using (A.8) together with Lemma A.3 we see

**Corollary A.4.** Let \( \alpha, \beta \) in \( \Omega \) 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, \( \omega_{x+1} \prec x \omega \) but \( \omega_{x+1} \prec x \omega \) would imply \( \omega_{x+1} \prec x \omega_x \), contradicting \( \prec x \subseteq \prec \).

**LEAN ORDINALS.** Let \( k \) be in \( \mathbb{N} \). We say that an ordinal \( \alpha \) in CNF(\( \varepsilon_0 \)) 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 \( \beta_i \), this for all \( i = 1, \ldots, m \). Observe that only 0 is 0-lean, and that any term in CNF is \( k \)-lean for some \( k \).

A value \( k \) of particular importance for lean ordinal terms is \( k = \omega_x - 1 \): observe that this is the coefficient value introduced when we compute a predecessor ordinal at \( x \). Stated differently, \( (\omega_x - 1) \)-leaness is an invariant of predecessor computations: if \( \alpha \) 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 $\alpha$ in CNF($\varepsilon_0$) be $(\omega_x - 1)$-lean. Then:

$$\alpha < \gamma \iff \alpha \prec_x \gamma \iff \alpha \preceq_x P_x(\gamma) \iff \alpha \leq P_x(\gamma).$$ (A.13)

One sees $(\bigcup_{x \in \mathbb{N}} \prec_x) = \prec$ over terms in CNF($\varepsilon_0$) as a result of Lemma A.5.

The proof relies on the syntactic characterisation of the ordinal ordering over terms in CNF($\varepsilon_0$) by

$$\alpha < \alpha' \iff \begin{cases} 
\alpha = 0 \text{ and } \alpha' \neq 0, \\
\alpha = \omega^\beta + \gamma, \alpha' = \omega^{\beta'} + \gamma' \text{ and } \begin{cases} 
\beta < \beta', \text{ or } \\
\beta = \beta' \text{ and } \gamma < \gamma'.
\end{cases}
\end{cases}$$ (A.14)

Since $\alpha \preceq_x P_x(\gamma)$ directly entails all the other statements of Lemma A.5, it is enough to prove:

Claim A.5.1. Let $\alpha, \gamma$ in CNF($\varepsilon_0$) and $x$ in $\mathbb{N}$. If $\alpha$ is $(\omega_x - 1)$-lean, then

$$\alpha < \gamma \implies \alpha \preceq_x P_x(\gamma).$$

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 $\gamma$, considering two cases:

1. if $\gamma = \gamma' + 1$ is a successor then $\alpha < \gamma$ implies $\alpha \leq \gamma'$, hence $\alpha \leq_x \gamma'$ (A.2)

2. if $\gamma$ is a limit, we claim that $\alpha < \gamma_x$, from which we deduce $\alpha \leq_x P_x(\gamma_x)$ (A.2)

We consider three subcases for the claim:

(a) if $\gamma = \omega^\lambda$ with $\lambda$ a limit, then $\alpha = \sum_{i=1}^m \omega^{\beta_i} \cdot c_i < \gamma$ implies $\beta_1 < \lambda$, hence $\beta_1 \leq \gamma_x \implies P_x(\lambda) = P_x(\lambda_x) < \lambda_x$, since $\beta_1$ is $(\omega_x - 1)$-lean. Thus $\alpha < \omega^\lambda = (\omega^\lambda)_x \leq \gamma_x$.

(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 $\alpha$ is $(\omega_x - 1)$-lean, hence $\alpha < \omega^{\beta_1} \cdot (c_1 + 1) \leq \omega^{\beta_1} \cdot \omega_x = \omega^{\beta_1 + \omega_x} = \gamma_x$.

(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 $\alpha$ can be written as $\alpha = \gamma' + \alpha'$ with $\alpha' < \omega^\beta$. In that case $\alpha \leq_x P_x(\omega^\beta)$ (A.2) $\implies P_x((\omega^\beta)_x) < (\omega^\beta)_x$, hence $\alpha = \gamma' + \alpha' < \gamma' + (\omega^\beta)_x = (\gamma' + \omega^\beta)_x = \gamma_x$. \qed
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 $h^\alpha : \mathbb{N} \to \mathbb{N}$, by inner iteration:

$$
\begin{align*}
  h^0(x) &\overset{\text{def}}{=} x; \\
  h^{\alpha+1}(x) &\overset{\text{def}}{=} h^\alpha(h(x)); \\
  h^\lambda(x) &\overset{\text{def}}{=} h^{\lambda x}(x).
\end{align*}
$$

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

$$
\begin{align*}
  H^0(x) &\overset{\text{def}}{=} x; \\
  H^{\alpha+1}(x) &\overset{\text{def}}{=} H^\alpha(x + 1); \\
  H^\lambda(x) &\overset{\text{def}}{=} H^{\lambda x}(x).
\end{align*}
$$

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:

$$
\begin{align*}
  h_0(x) &\overset{\text{def}}{=} 0; \\
  h_{\alpha+1}(x) &\overset{\text{def}}{=} 1 + h_\alpha(h(x)); \\
  h_\lambda(x) &\overset{\text{def}}{=} h_{\lambda x}(x).
\end{align*}
$$

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

$$
\begin{align*}
  H_0(x) &\overset{\text{def}}{=} 0; \\
  H_{\alpha+1}(x) &\overset{\text{def}}{=} 1 + H_\alpha(x + 1); \\
  H_\lambda(x) &\overset{\text{def}}{=} H_{\lambda x}(x).
\end{align*}
$$

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

$$
\begin{align*}
  f_0(x) &\overset{\text{def}}{=} h(x); \\
  f_{\alpha+1}(x) &\overset{\text{def}}{=} f^{\omega_x}(x); \\
  f_\lambda &\overset{\text{def}}{=} f_{\lambda x}(x),
\end{align*}
$$

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

$$
\begin{align*}
  F_0(x) &\overset{\text{def}}{=} x + 1; \\
  F_{\alpha+1}(x) &\overset{\text{def}}{=} F^{\omega_x}(x); \\
  F_\lambda &\overset{\text{def}}{=} F_{\lambda x}(x).
\end{align*}
$$

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

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

$$
\begin{align*}
  h_\alpha(x) &\overset{\text{def}}{=} 1 + h_{P_\alpha}(h(x)); \\
  h^\alpha(x) &\overset{\text{def}}{=} h_{P_\alpha}(h(x)) = h^{P_\alpha+1}(x); \\
  f_\alpha(x) &\overset{\text{def}}{=} f_{P_\alpha}(x) = f_{P_\alpha+1}(x).
\end{align*}
$$
Proof. We only prove (A.21); (A.22) and (A.23) can be proven similarly. By transfinite induction over \(\alpha\). For a successor ordinal \(\alpha + 1\), \(h_{\alpha+1}(x) = 1 + h_{\alpha}(h(x))\). For a limit ordinal \(\lambda\), \(h_\lambda(x) = h_{\lambda_\ast}(x) = 1 + h_{P_x(\lambda)}(h(x))\), where the ind. hyp. can applied since \(0 < \lambda_\ast < \lambda\).

\[\text{Lemma A.7.} \quad \text{Let } h(x) > x \text{ for all } x. \text{ Then for all } \alpha \in \Omega \text{ and } x \in \mathbb{N} \text{ with } \omega_x > 0, \]
\[h_\alpha(x) \leq h^\alpha(x) - x.\]

\[\text{Proof.} \quad \text{By induction over } \alpha. \text{ For } \alpha = 0, h_0(x) = 0 = x - x = h^0(x) - x. \text{ For } \alpha > 0,\]
\[h_\alpha(x) = 1 + h_{P_x(\alpha)}(h(x)) \quad \text{(by Lemma A.6)}\]
\[\leq 1 + h_{P_x(\alpha)}(h(x)) - h(x) \quad \text{(by ind. hyp. since } P_x(\alpha) < \alpha)\]
\[\leq h_{P_x(\alpha)}(h(x)) - x \quad \text{(since } h(x) > x)\]
\[= h^\alpha(x) - x. \quad \text{(by (A.22))}\]

Using the same argument, one can check that in particular for \(h(x) = x + 1,\)
\[H_\alpha(x) = H^\alpha(x) - x. \quad \text{(A.24)}\]

\[\text{Lemma A.8.} \quad \text{For all } \alpha, \gamma \in \Omega \text{, and } x,\]
\[h^{\gamma + \alpha}(x) = h^\gamma(h^\alpha(x)).\]

\[\text{Proof.} \quad \text{By transfinite induction on } \alpha. \text{ For } \alpha = 0, h^{\gamma+0}(x) = h^\gamma(x) = h^\gamma(h^0(x)).\]

For a successor ordinal \(\alpha + 1,\)
\[h^{\gamma + \alpha}(x) = h_\gamma(h(x)) \quad \text{(by A.22)}\]
\[= h^\gamma(h^{\alpha+1}(x)). \text{ For a limit ordinal } \lambda, h^{\gamma + \lambda}(x) = h^{(\gamma + \lambda)}(x) = h^{\gamma + \lambda}(x) \quad \text{(A.15)}\]
\[h^{\gamma + \lambda}(x) = h^\gamma(h^{\lambda_\ast}(x)) = h^\gamma(h^\lambda(x)). \quad \text{(A.24)}\]

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.

\[\text{Lemma A.10.} \quad \text{For all } \beta \in \Omega \text{, and } r, x \in \mathbb{N},\]
\[h^{\omega^\beta}(x) = f^r_\beta(x).\]

\[\text{Proof.} \quad \text{In view of Lemma A.8 and } f^0 = f^0 = Id_\mathbb{N}, \text{ it is enough to prove } h^{\omega^\beta} = f_\beta, \]
\[\text{i.e., the } r = 1 \text{ case. We proceed by induction over } \beta. \]

For the base case. \(h^{\omega^0}(x) = h^1(x) \quad \text{(A.19)}\]
\(= f_0(x).\)

For a successor \(\beta + 1.\)
\[h^{\omega^{\beta+1}}(x) \quad \text{(A.15)}\]
\[= h^{(\omega^{\beta+1})}(x) = h^{\omega^{\beta, \omega^x}}(x) \quad \text{(A.19)}\]
\[= f_{\beta+1}(x). \quad \text{(A.19)}\]

For a limit \(\lambda.\)
\[h^{\omega^\lambda}(x) \quad \text{(A.15)}\]
\[= h^{\omega^{\lambda_\ast}}(x) \quad \text{(A.19)}\]
\[= f_{\lambda}(x). \quad \text{(A.19)}\]
We set to prove in this section the main monotonicity and expansiveness properties of our various hierarchies.

**Lemma A.11** ([Cichon and Tahhan Bittar, 1998]). Let \( h \) be an expansive monotone function. Then, for all \( \alpha, \alpha' \in \Omega \) and \( x, y \in \mathbb{N} \),

\[
\begin{align*}
x < y & \implies h_\alpha(x) \leq h_\alpha(y), & (A.25) \\
\alpha' \prec_x \alpha & \implies h_{\alpha'}(x) \leq h_\alpha(x). & (A.26)
\end{align*}
\]

**Proof.** Let us first deal with \( \alpha' = 0 \) for (A.26). Then \( h_0(x) = 0 = h_\alpha(x) \) for all \( \alpha \) and \( x \).

Assuming \( \alpha' > 0 \), the proof now proceeds by simultaneous transfinite induction over \( \alpha \).

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), \( h_{\alpha+1}(x) = 1 + h_\alpha(h(x)) \leq 1 + h_\alpha(h(y)) = h_{\alpha+1}(y) \) where the ind. hyp. on (A.25) can be applied since \( h \) is monotone.

For (A.26), we have \( \alpha' \preceq_x \alpha \prec_x \alpha + 1 \), hence \( h_{\alpha'}(x) \leq h_{\alpha}(x) \leq h_\alpha(h(x)) \) where the ind. hyp. on (A.25) can be applied since \( h(x) \geq x \).

For a limit \( \lambda \). For (A.25), \( h_\lambda(x) = h_{\lambda_x}(x) \leq h_{\lambda_y}(y) = h_\lambda(y) \) where the ind. hyp. on (A.26) can be applied since \( \lambda_x \preceq_y \lambda_y \) by Lemma A.3.

For (A.26), we have \( \alpha' \preceq_x \lambda_x \prec_x \lambda \) with \( h_{\alpha'}(x) \leq h_{\lambda_x}(x) = h_\lambda(x) \).

Essentially the same proof can be carried out to prove the same monotonicity properties for \( h^\alpha \) and \( f_\alpha \). As the monotonicity properties of \( f_\alpha \) 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 \( \alpha, \alpha' \in \Omega \), \( x, y \in \mathbb{N} \) with \( \omega_x > 0 \),

\[
\begin{align*}
f_\alpha(x) & \geq h(x) \geq x. & (A.27) \\
\alpha' \prec_x \alpha & \implies f_{\alpha'}(x) \leq f_\alpha(x). & (A.28) \\
x < y & \text{ and } h \text{ monotone imply } f_{\alpha'}(x) \leq f_\alpha(y). & (A.29)
\end{align*}
\]
Proof of (A.27). By transfinite induction on $\alpha$. 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. \qed

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 $\alpha$ by (A.27).

Assuming $\alpha' > 0$, the proof proceeds by transfinite induction over $\alpha$. The case $\alpha = 0$ is impossible. For the successor case, $\alpha' \leq_x \alpha \leq_x \alpha + 1$ with $f_{\alpha+1}(x) = f_\alpha^{\omega_x - 1}(f_\alpha(x)) \geq f_\alpha(x) \geq f_{\alpha'}(x)$. For the limit case, we have $\alpha' \leq_x \lambda_x \leq_x \lambda$ with $f_{\alpha'}(x) \leq f_{\lambda_x}(x) = f_\lambda(x)$.

Proof of (A.29). By transfinite induction over $\alpha$. For the base case, $f_0(x) = h(x) \leq h(y) = f_0(y)$ since $h$ is monotone. For the successor case, $f_{\alpha+1}(x) = f_\alpha^{\omega_x}(x) \leq f_\alpha^{\omega_y}(x) \leq f_{\alpha+1}(y)$ using $\omega_x \leq \omega_y$. For the limit case, $f_\lambda(x) = f_\lambda(x) \leq f_\lambda(x) \leq f_{\lambda_y}(y) = f_\lambda(y)$, where (A.28) can be applied thanks to Lemma A.3. \qed

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 $\alpha$ in $\Omega$. 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 $\alpha$. For $0$, $h_{s,0}(x) = 0 = h_{id,0}(s(x))$. For a successor ordinal $\alpha + 1$, $h_{s,\alpha+1}(x) = 1 + h_{s,\alpha}(h(x)) \leq 1 + h_{id,\alpha}(s(h(x))) \leq 1 + h_{id,\alpha}(h(s(x))) = h_{id,\alpha+1}(s(x))$ since $s(h(x)) \leq h(s(x))$. For a limit ordinal
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 Cichon and Wainer (1983):

**Lemma A.14.** For all \( \gamma \) and \( \alpha \) in \( \Omega \), if \( h \) is monotone eventually dominated by \( F_\gamma \), then \( f_\alpha \) 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 \( \alpha \) 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 \( x \geq x_0 \),

\[
\begin{align*}
\alpha^n(x) & \leq F_{\gamma+\alpha}^n(x) . \\
\text{Indeed, by induction on } n, \text{ for all } x \geq x_0, \\
f_0(x) &= x = F_0^0(x) \\
f_\alpha^n(x) &= f_\alpha(f_\alpha^n(x)) \\
&\leq f_\alpha(F_{\gamma+\alpha}^n(x)) \quad \text{(by (A.29) on } f_\alpha \text{ and the ind. hyp. on } n) \\
&\leq F_{\gamma+\alpha}(F_{\gamma+\alpha}^n(x)) \quad \text{(since by (A.27) } F_{\gamma+\alpha}(x) \geq x \geq x_0 \text{ and by ind. hyp. on } \alpha) \\
&= F_{\gamma+\alpha}^{n+1}(x) .
\end{align*}
\]

Therefore

\[
\begin{align*}
f_{\alpha+1}(x) &= f_\alpha^x(x) \\
&\leq F_{\gamma+\alpha}^x(x) \quad \text{(by (A.30) for } n = x) \\
&= F_{\gamma+\alpha+1}(x) .
\end{align*}
\]
For a limit ordinal \( \lambda \): for all \( x \geq x_0 \), \( f_\lambda(x) = f_{\lambda x}(x) \leq F_{\gamma+\lambda x}(x) = F_{(\gamma+\lambda),x}(x) = F_{\gamma+\lambda}(x) \).

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 \( + \) 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 \( \Omega \), 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 \( "+" \), by interpreting \( + \) as the direct sum in ordinal arithmetic. This would be a mistake: in a situation where two different terms \( \alpha \) and \( \alpha' \) denote the same ordinal \( \text{ord}(\alpha) = \text{ord}(\alpha') \), we do not necessarily have \( F_\alpha(x) = F_{\alpha'}(x) \): for instance, \( \alpha = \omega^\omega \) and \( \alpha' = \omega^0 + \omega^\omega \) denote the same ordinal \( \omega \), but \( F_\alpha(2) = F_\alpha(2) = 2^2 \cdot 2 = 3 \) and \( F_{\alpha'}(2) = F_3(2) = 2^2 \cdot 2 = 2^{11} \). 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. \( F_{\gamma+\omega^\omega} \) with \( \gamma < \omega^\omega \) is multiply-recursive if our results are to be of any use.

The most interesting case is the one where \( \gamma \) is finite but \( \alpha \) 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 \( \text{CNF}(\varepsilon_0) \), 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 \( x \), \( F_{\text{id},\gamma+\alpha}(x) \leq F_{s,\alpha}(x) \).

**base case for \( \omega \):** \( F_{\text{id},\gamma+\omega}(x) = F_{\text{id},\gamma+x}(x) = F_{s,\omega}(x) \),

**successor case \( \alpha + 1 \):** with \( \alpha \geq \omega \), an induction on \( n \) shows that \( F_{\text{id},\gamma+\alpha}(x) \leq F_{s,\alpha}(x) \) for all \( n \) and \( x \) using the ind. hyp. on \( \alpha \), thus \( F_{\text{id},\gamma+\alpha+1}(x) = F_{\text{id},\gamma+\alpha}(x) \overset{(A.27)}{\leq} F_{s,\alpha}(x) \leq F_{s,\alpha+1}(x) \).

**limit case \( \lambda > \omega \):** \( F_{\text{id},\gamma+\lambda}(x) = F_{\text{id},\gamma+\lambda x}(x) \overset{i_h}{\leq} F_{s,\lambda x}(x) \overset{(A.28)}{\leq} F_{s,\lambda x+\gamma}(x) = F_{s,\lambda}(x) \) where \( (A.28) \) can be applied since \( \lambda_x \leq x \lambda_{x+\gamma} \) 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 $x$,

$$
F_{id,\gamma+\alpha}(x) \leq F_{s,\alpha}(x)
$$

(by the previous claim)

$$
= H_s^\omega(x)
$$

(by Lemma A.10)

$$
\leq H_{id}^\omega(s(x))
$$

(by Lemma A.13 and (A.24))

$$
= F_{id,\alpha}(s(x)).
$$

(by Lemma A.10)

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 ($\mathcal{F}_\alpha$)$_{\alpha \in \text{CNF}(\varepsilon_0)}$ defined by L"ob and Wainer (1970) on top of the fast-growing hierarchy ($F_\alpha$)$_{\alpha \in \text{CNF}(\varepsilon_0)}$ for $\omega_x \overset{\text{def}}{=} x$.

Let us first provide some background on the definition and properties of $\mathcal{F}_\alpha$.

The class of functions $\mathcal{F}_\alpha$ is the closure of the constant, addition, projection (including identity), and $F_\alpha$ 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

$$
 f(0, x_1, \ldots, x_n) = h_1(x_1, \ldots, x_n),
$$

$$
 f(y + 1, x_1, \ldots, x_n) = h_2(y, x_1, \ldots, x_n, f(y, x_1, \ldots, x_n)),
$$

$$
 f(y, x_1, \ldots, x_n) \leq h_3(y, x_1, \ldots, x_n).
$$

The hierarchy is strict for $\alpha > 0$, i.e. $\mathcal{F}_\alpha' \subsetneq \mathcal{F}_\alpha$ if $\alpha' < \alpha$, because in particular $F_{\alpha'} \notin \mathcal{F}_\alpha$. For small finite values of $\alpha$, the hierarchy characterises some well-known classes of functions:

- $\mathcal{F}_0 = \mathcal{F}_1$ contains all the linear functions, like $\lambda x. x + 3$ or $\lambda x. 2x$, along with many simple ones like cut-off subtraction: $\lambda x y. x \div y$, which yields $x - y$ if $x \geq y$ and 0 otherwiseootnote{By limited primitive recursion: first define $\lambda x. x \div 1$ by $0 \div 1 = 0$ and $(y + 1) \div 1 = y$; then $x \div 0 = x$ and $x \div (y + 1) = (x \div y) \div 1$.} or simple predicates like odd: $\lambda x. x \mod 2$ootnote{By limited primitive recursion: $0 \mod 2 = 0$ and $(y + 1) \mod 2 = 1 \div (y \mod 2)$.}

- $\mathcal{F}_2$ is exactly the set of elementary functions, like $\lambda x. 2^{2x}$.
Appendix A. Subrecursive Functions

- \( F_3 \) contains all the tetration functions, like \( \lambda x. 2^{2^{\ldots^2}} \), etc.

The union \( \bigcup_{\alpha<\omega} F_\alpha \) is the set of primitive-recursive functions, while \( F_\omega \) is an Ackermann-like non primitive-recursive function. Similarly, \( \bigcup_{\alpha<\omega} F_\alpha \) is the set of multiply-recursive functions with \( F_{\omega \omega} \) a non multiply-recursive function.

The following properties (resp. Theorem 2.10 and Theorem 2.11 in \( \text{(Löb and Wainer, 1970)} \)) are useful: for all \( \alpha \), unary \( f \) in \( F_\alpha \), and \( x \),

\[
\alpha > 0 \implies \exists p, \ f(x) \leq F_\alpha^p(x+1) , \tag{A.31}
\]

\[
\exists p, \forall x \geq p, \ f(x) \leq F_{\alpha+1}(x) . \tag{A.32}
\]

Also note that by \( (A.31) \), if a unary function \( g \) is dominated by some function \( g' \) in \( F_\alpha \) with \( \alpha > 0 \), then there exists \( p \) s.t. for all \( x \), \( g(x) \leq g'(x) \leq F_\alpha^p(x+1) \).

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 \( (F_\alpha)_{\alpha} \):

**Lemma A.17.** For all \( \gamma > 0 \) and \( \alpha \), if \( h \) is monotone and eventually dominated by a function in \( F_\gamma \), then

1. if \( \alpha < \omega \), \( f_\alpha \) is dominated by a function in \( F_{\gamma+\alpha} \), and

2. if \( \gamma < \omega \) and \( \alpha \geq \omega \), \( f_\alpha \) is dominated by a function in \( F_\alpha \).

**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 \( F_\gamma \) by hypothesis.

For the successor case \( \alpha = k + 1 \): by ind. hyp. \( f_k \) is dominated by a function in \( F_{\gamma+k} \), thus by \( (A.31) \) there exists \( p \) s.t. \( f_k(x) \leq F_{\gamma+k}^p(x+1) = F_{\gamma+k}^p \circ F_0(x) \).

By induction on \( n \), we deduce

\[
f_k^n(x) \leq (F_{\gamma+k}^p \circ F_0)^n(x) ; \tag{A.33}
\]

Therefore,

\[
f_{k+1}(x) = f_k^n(x) \tag{A.34}
\]

\[
\leq (F_{\gamma+k}^p \circ F_0)^x(x) \tag{A.35}
\]

\[
\leq F_{\gamma+k}^{(p+1)x+1}((p+1)x+1) \tag{A.36}
\]

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 \( F_{\gamma+k+1} \). \( \square \)
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, $f_\alpha(x) \leq f_\alpha(x + x_0) \leq F_{\alpha}(x + x_0 + \gamma + 1)$ for all $x$, where the latter function $x \mapsto F_{\alpha}(x + x_0 + \gamma + 1)$ is in $\mathcal{F}_{\alpha}$. □
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 \textsc{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 \\textsc{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

\[
\textsc{ExpTime} \overset{\text{def}}{=} \bigcup_{c} \textsc{DTIME}(2^{n^c})
\]

and the corresponding nondeterministic and space-bounded classes as

\[
\textsc{NExpTime} \overset{\text{def}}{=} \bigcup_{c} \textsc{NTIME}(2^{n^c}) \quad \text{and} \quad \textsc{ExpSpace} \overset{\text{def}}{=} \bigcup_{c} \textsc{SPACE}(2^{n^c}).
\]

Problems complete for \textsc{ExpTime}, like corridor tiling games (Chlebus, 1986) or equivalence of regular tree languages (Seidl, 1990), are known not to be in \textsc{PTime},
hence the denomination “truly intractable” or “provably intractable” in the literature.

We can generalise these classes of problems to the exponential hierarchy

\[ k\text{-ExpTime} \overset{\text{def}}{=} \bigcup_{c} DTime \left( 2^{\cdot \cdot \cdot 2^n} \right), \]

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

\[ \text{Elementary} \overset{\text{def}}{=} \bigcup_{k} k\text{-ExpTime} = \bigcup_{c} DTime \left( 2^{\cdot \cdot \cdot 2^n} \right). \]

Note that we could as easily define \text{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 \((\mathcal{F}_\alpha)_{\alpha<\varepsilon_0}\) is an infinite hierarchy of classes of functions \(f\) with argument(s) and images in \(\mathbb{N}\) \cite{LobWainer}. At the heart of each \(\mathcal{F}_\alpha\) lies the \(\alpha\)th fast-growing function \(F_\alpha: \mathbb{N} \to \mathbb{N}\), which is defined by

\[
F_0(x) \overset{\text{def}}{=} x + 1, \quad F_{\alpha+1}(x) \overset{\text{def}}{=} F_{\alpha}^{x+1}(x) = F_{\alpha} (F_{\alpha} (\cdots F_{\alpha}(x))) ,
\]

where \(\lambda_x < \lambda\) is the \(x\)th element of the fundamental sequence for the limit ordinal \(\lambda\), defined by

\[
(\gamma + \omega^{\beta+1})_x \overset{\text{def}}{=} \gamma + \omega^\beta \cdot x, \quad (\gamma + \omega^\lambda)_x \overset{\text{def}}{=} \gamma + \omega^\lambda x.
\]

For instance,

\[
F_1(x) = 2x + 1, \quad F_2(x) = 2^{x+1}(x + 1) - 1,
\]

\[
F_3(x) > 2^{\cdot \cdot \cdot 2} \text{ \(x\) times},
\]

\(F_\omega\) is an Ackermannian function,

\(F_{\omega^\omega}\) 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

\[ \mathcal{F}_\alpha = \bigcup_{c} \text{FDTIME} \left( F_{\alpha}^c(n) \right), \quad \text{(B.1)} \]
the choice between deterministic and nondeterministic or between time-bounded 
and space-bounded computations being once more irrelevant because $F_2$ is al-
ready a function of exponential growth. In particular, $F_\alpha^c$ belongs to $\mathcal{F}_\alpha$ for every 
$\alpha$ and fixed $c$.

Every function $f$ in $\mathcal{F}_\alpha$ is honest, i.e. can be computed in time elementary in 
itself \cite{Vainer1970}—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 ex-
ists 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 $\alpha' > \alpha > 0$, $F_{\alpha'} \notin \mathcal{F}_\alpha$, and the hierarchy 
$(\mathcal{F}_\alpha)_{\alpha < \varepsilon_0}$ is strict for $\alpha > 0$.

**Important Stops.** Although some deep results have been obtained on the lower 
classes\footnote{See \cite{Ritchie1963} for a characterisation of $\text{FLinSpace}$, and for variants see e.g. \cite{Cobham1965, BellantoniCook1992} for $\text{FPTime}$, or the chapter by \cite{Clote1999} for a survey of these techniques.}, we focus here on the non-elementary classes, i.e. on $\alpha \geq 2$, where we find for instance

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

We are dealing here with classes of functions, but writing $\mathcal{F}_\alpha^*$ for the restriction 
of $\mathcal{F}_\alpha$ 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 prob-
lems, which are non elementary (resp. non primitive-recursive, or non multiply-
recursive, \ldots), but only barely so. The issue is that complexity classes like e.g. $\mathcal{F}_3^*$, 
which is the first class that contains non-elementary problems, are very large: $\mathcal{F}_3^*$ 
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 $\mathcal{F}_3$ cannot be 
obtained for the classical examples of non-elementary problems.

We therefore introduce smaller classes:

$$F_\alpha \overset{\text{def}}{=} \bigcup_{p \in \bigcup_{\beta < \alpha} \mathcal{F}_\beta} \text{DTIME}(F_\alpha(p(n))) . \quad \text{(B.2)}$$
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 $F_3$ of non-elementary decision problems closed under elementary reductions, a class $F_\omega$ of Ackermannian problems closed under primitive-recursive reductions, a class $F_{\omega,\omega}$ of hyper-Ackermannian problems closed under multiply-recursive reductions, etc. We can name a few of these complexity classes:

$$F_\omega = \text{ACKERMANN},$$

$$F_{\omega,\omega} = \text{HYPER-ACKERMANN}.$$ 

Of course, we could replace in (B.2) the class of reductions $\cup_{\beta<\alpha} F_\beta$ by a more traditional one, like FLogSpace or FPTIME, or for $\alpha \geq \omega$ by primitive-recursive reductions in $\cup_k F_k$ as done by Chambart (2011). However this definition better captures the intuition one can have of a problem being “complete for $F_\alpha$.”

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 $F_3$-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 $\Sigma$ as a term $e$ with abstract syntax

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

where $a$ ranges over $\Sigma$ and $\varepsilon$ denotes the empty string. Such expressions are inductively interpreted as languages included in $\Sigma^*$ by:

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

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) proved that $F_3$ is an example of such a problem.

An alternative class for $\alpha \geq 3$ is

$$F_\alpha \overset{\text{def}}{=} \bigcup_c \text{DTime} \left(F_\alpha(n + c)\right),$$

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 $(H_\beta)_{\beta<\varepsilon_0}$ of functions (Wainer, 1972), which provides an infinite refinement of each $F_\beta$ as $\bigcup_{\beta<\omega^{\omega+1}} H_\beta$, but its classes lack both forms of robustness: any $f$ in $H_\beta$ is bounded by $H_\beta^\beta$ the $\beta$th function of the Hardy hierarchy. What we define here as $F_\alpha$ seems closer to $\bigcup_{\beta<\omega^{\omega+2}} H_\beta^\ast$. 

---

2 An alternative class for $\alpha \geq 3$ is
show that this problem is hard for $2^{\log n \times n}$ space under FLOGSPACE reductions. Then, $F_3$-hardness follows by an F ELEMENTARY reduction from any Turing machine working in space $F_3(p(n))$ into a machine working in space $2^{\log n \times n}$.

That the problem is in $F_3$ 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 $2^{\log n \times n}$, thus in $F_3$.

B.2 Ackermann-Complete Problems

We gather here some decision problems that can be proven decidable in $F_\omega$. The common trait of all these problems is their reliance on Dickson’s Lemma over $\mathbb{N}^d$ 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 $F_{\alpha}$-complete problems for higher $\alpha$ 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 $\langle v_0, A \rangle$ where $v_0$ is an initial configuration in $\mathbb{N}^d$ and $A$ is a finite set of transitions in $\mathbb{Z}^d$. A transition $u$ in $A$ can be applied to a configuration $v$ in $\mathbb{N}^d$ if $v' = v + u$ is in $\mathbb{N}^d$; the resulting configuration is then $v'$. The complexity of decision problems for VAS usually varies from EXPSPACE-complete (Lipton, 1976; Rackoff, 1978; Blockelet and Schmitz, 2011) to $F_\omega$-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 Reach($\mathcal{V}_1$) and Reach($\mathcal{V}_2$) of reachable configurations.

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


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 \(\text{ExpSpace}\)-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 \(\text{Ackermann}\)-complete by Hofman and Totzke (2014). A labelled vector addition system with states (VASS) \(\mathcal{V} = (Q, \Sigma, d, T, q_0, v_0)\) 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 \(\Sigma\): transitions are then defined as quadruples \(q \xrightarrow{a, u} q'\) for \(a\) in \(\Sigma\) and \(u\) in \(\mathbb{Z}^d\). Such a system defines an infinite labelled transition system \(\langle Q \times \mathbb{N}^d, \to, (q_0, v_0) \rangle\) where \((q, v) \xrightarrow{a} (q', v + u)\) if \(q \xrightarrow{a, u} q'\) is in \(T\) and \(v + u \geq 0\). The set of traces of \(\mathcal{V}\) is the set of finite sequences \(L(\mathcal{V}) = \{a_1 \cdots a_n \in \Sigma^* \mid \exists (q, v) \in Q \times \mathbb{N}^d. (q_0, v_0) \xrightarrow{a_1 \cdots a_n} (q, v)\}\).

[1VASSU] One-Dimensional VASS Universality
instance: A one-dimensional labelled VASS \(\mathcal{V} = (Q, \Sigma, 1, T, q_0, x_0)\).
question: Does \(L(\mathcal{V}) = \Sigma^*\), i.e. is every finite sequence over \(\Sigma\) 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 \(\text{Ackermann}\)-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 \(u\) 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, \(u\) plus the vector labels of all the edges used so far is component-wise 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 \(2\text{ExpTime}\)-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 $F_\omega$-complete by Pérez (2016):

**[POE]** Energy Games with Partial Observation

*instance*: A multi-weighted finite game graph $G = (V, E)$ with labelling $\lambda: E \to \mathbb{Z}^d$, $v \in V$, $u \in \mathbb{N}^d$, 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 $u$?

*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 $\sigma$.

*question*: Is $\sigma$ reachable in $M$ with lossy semantics?

*lower bound*: Schnoebelen (2010a), by a direct reduction from $F_\omega$-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 “$A \rightarrow B$”, only if $A$ is actually relevant in the deduction of $B$. This excludes for instance $A \rightarrow (B \rightarrow A)$, $(A \land \neg A) \rightarrow B$, etc.—see Dunn and Restall (2002) for more details. Although the full logic $R$ is undecidable (Urquhart, 1984), its conjunctive-implicative fragment $R_{\rightarrow, \land}$ is decidable, and Ackermann-complete:

**[CRI]** Conjunctive Relevant Implication

*instance*: A formula $A$ of $R_{\rightarrow, \land}$.

*question*: Is $A$ a theorem of $R_{\rightarrow, \land}$?
lower bound: Urquhart (1999), from a variant of LCM: the emptiness problem of alternating expansive counter machines, for which he proved $F_{\omega}$-hardness directly from $F_{\omega}$-MM the halting problem in $F_{\omega}$-bounded Minsky machines.


comment: Hardness also holds for any intermediate logic between $R_{\rightarrow, \wedge}$ and $T_{\rightarrow, \wedge}$, 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 $A$.

question: Is the data language $L(A)$ 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 $F_{\omega}$ in the case of linearly ordered data (Ouaknine and Worrell, 2007), and even to $F_{\omega}$ 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 $\Sigma$ 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 $F_{[3,\infty)} \varphi$ at position $i$, written $w, i \models F_{[3,\infty)} \varphi$, only if $\varphi$ holds at some
position \( j > i \) of \( w \) with timestamp \( \tau_j - \tau_i \geq 3 \). The safety fragment of MTL restricts the intervals decorating “until” modalities to be right-bounded.

[SMTL] Satisfiability of Safety Metric Temporal Logic

*instance:* A safety MTL formula \( \varphi \).

*question:* Does there exist an infinite non-Zeno timed word \( w \) s.t. \( 0 \models \varphi \)?

*lower bound:* \[ \text{Lazić et al. (2016)}, \] by a direct reduction from \( F_\omega \)-bounded Turing machines.

*upper bound:* \[ \text{Lazić et al. (2016)}, \] 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 \[ \text{Lazić et al. (2016)} \] show to be \text{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 \[ \text{(Dauchet and Tison, 1990)}, \] 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 \( \Sigma \) 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 \( (q, t) \rightarrow (q', t') \) acting over pairs of states and trees, which rewrite a configuration \( (q, C[t]) \) into \( (q', C[t']) \) in any context \( C \).

\[ \text{Hague (2014)}, \] 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 \( (q, C[t]) \rightarrow (q', C[t']) \), 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:* \[ \text{Hague (2014)}, \] from coverability in reset Petri nets, see \[ \text{LCM} \].

*upper bound:* \[ \text{Hague (2014)}, \] by reducing to coverability in reset Petri nets, see \[ \text{LCM} \].

### B.2.8 Interval Temporal Logics

Interval Temporal Logics provide a formal framework for reasoning about temporal intervals. \[ \text{Halpern and Shoham (1991)} \] 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 \( \langle B \rangle \) and \( \langle E \rangle \). 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 \( A \) expresses that the two intervals “meet”, i.e. share an endpoint—, has an \( F_\omega \)-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 \( \varphi \).

question: Does there exist an interval structure \( S \) over some finite linear order and an interval \( I \) of \( S \) s.t. \( S, I \models \varphi \)?

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


References


<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ackermann complexity</td>
<td>107</td>
</tr>
<tr>
<td>function</td>
<td>31, 104, 107</td>
</tr>
<tr>
<td>antichain</td>
<td>2</td>
</tr>
<tr>
<td>ascending chain condition</td>
<td>3</td>
</tr>
<tr>
<td>atom of $X$</td>
<td>80</td>
</tr>
<tr>
<td>backward coverability</td>
<td>32</td>
</tr>
<tr>
<td>basis, of an upward-closed set</td>
<td>13</td>
</tr>
<tr>
<td>better quasi orders</td>
<td>23</td>
</tr>
<tr>
<td>bounding function</td>
<td>40</td>
</tr>
<tr>
<td>canonical decomposition</td>
<td>71</td>
</tr>
<tr>
<td>Cantor Normal Form</td>
<td>11, 89</td>
</tr>
<tr>
<td>cartesian product</td>
<td>29, 37</td>
</tr>
<tr>
<td>ideals</td>
<td>78</td>
</tr>
<tr>
<td>with lexicographic ordering</td>
<td>15, 79</td>
</tr>
<tr>
<td>Cichoń hierarchy</td>
<td>13, 49, 51, 93</td>
</tr>
<tr>
<td>compatibility</td>
<td>4</td>
</tr>
<tr>
<td>downward</td>
<td>20</td>
</tr>
<tr>
<td>reflexive transitive</td>
<td>19</td>
</tr>
<tr>
<td>strict</td>
<td>20</td>
</tr>
<tr>
<td>transitive</td>
<td>19</td>
</tr>
<tr>
<td>contraction</td>
<td>11</td>
</tr>
<tr>
<td>ordering</td>
<td>10</td>
</tr>
<tr>
<td>rule</td>
<td>10</td>
</tr>
<tr>
<td>control</td>
<td>27</td>
</tr>
<tr>
<td>control function</td>
<td>28, 85</td>
</tr>
<tr>
<td>controlled sequence</td>
<td>28</td>
</tr>
<tr>
<td>control-state reachability</td>
<td>8</td>
</tr>
<tr>
<td>counter machine</td>
<td>54</td>
</tr>
<tr>
<td>expansive alternating</td>
<td>110</td>
</tr>
<tr>
<td>extended</td>
<td>54</td>
</tr>
<tr>
<td>incrementing</td>
<td>67, 110</td>
</tr>
<tr>
<td>lossy</td>
<td>55, 110</td>
</tr>
<tr>
<td>Minsky</td>
<td>54</td>
</tr>
<tr>
<td>reset</td>
<td>53, 110</td>
</tr>
<tr>
<td>transfer</td>
<td>67, 110</td>
</tr>
<tr>
<td>coverability</td>
<td>32, 56, 62, 67</td>
</tr>
<tr>
<td>backward</td>
<td>5</td>
</tr>
<tr>
<td>dual</td>
<td>53</td>
</tr>
<tr>
<td>forward</td>
<td>82</td>
</tr>
<tr>
<td>covering</td>
<td>12</td>
</tr>
<tr>
<td>cut elimination</td>
<td>10</td>
</tr>
<tr>
<td>cut-off subtraction</td>
<td>48</td>
</tr>
<tr>
<td>data logic</td>
<td>110</td>
</tr>
<tr>
<td>Descent Equation</td>
<td>35</td>
</tr>
<tr>
<td>Dickson’s Lemma</td>
<td>8, 14</td>
</tr>
<tr>
<td>directed</td>
<td>71</td>
</tr>
<tr>
<td>disjoint sum</td>
<td>29, 37</td>
</tr>
<tr>
<td>ideals</td>
<td>78</td>
</tr>
<tr>
<td>with lexicographic ordering</td>
<td>15, 79</td>
</tr>
<tr>
<td>disjunctive termination argument</td>
<td>8</td>
</tr>
<tr>
<td>downward</td>
<td>2</td>
</tr>
<tr>
<td>closed</td>
<td>2</td>
</tr>
<tr>
<td>closure</td>
<td>2</td>
</tr>
<tr>
<td>compatibility</td>
<td>20</td>
</tr>
<tr>
<td>WSTS</td>
<td>20</td>
</tr>
<tr>
<td>effective pred-basis</td>
<td>82, 84</td>
</tr>
<tr>
<td>effective presentation of a wqo ideal presentation</td>
<td>73</td>
</tr>
<tr>
<td>effective wqo</td>
<td>73</td>
</tr>
<tr>
<td>Egli-Milner ordering</td>
<td>18</td>
</tr>
<tr>
<td>energy game</td>
<td>109</td>
</tr>
<tr>
<td>exchange rule</td>
<td>10</td>
</tr>
<tr>
<td>excluded minor</td>
<td>69</td>
</tr>
<tr>
<td>excluded minors</td>
<td>5</td>
</tr>
<tr>
<td>fast-growing hierarchy</td>
<td>30, 50, 73, 104</td>
</tr>
<tr>
<td>filter</td>
<td>71</td>
</tr>
<tr>
<td>principal</td>
<td>69</td>
</tr>
<tr>
<td>finite antichain condition</td>
<td>2</td>
</tr>
<tr>
<td>finite basis property</td>
<td>15</td>
</tr>
<tr>
<td>fundamental sequence</td>
<td>42, 90</td>
</tr>
<tr>
<td>Graph Minor Theorem</td>
<td>4</td>
</tr>
<tr>
<td>ground term rewrite system</td>
<td>111</td>
</tr>
<tr>
<td>Grzegorczyk hierarchy</td>
<td>26, 30, 48, 51, 100</td>
</tr>
<tr>
<td>Hardness Theorem</td>
<td>53, 54, 64, 67</td>
</tr>
<tr>
<td>Hardy</td>
<td>53</td>
</tr>
<tr>
<td>Term</td>
<td>Page(s)</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------</td>
</tr>
<tr>
<td>computation</td>
<td>57</td>
</tr>
<tr>
<td>hierarchy</td>
<td>49, 56, 94</td>
</tr>
<tr>
<td>Higman’s Lemma</td>
<td>3, 17</td>
</tr>
<tr>
<td>for infinite words</td>
<td>17</td>
</tr>
<tr>
<td>Hoare ordering</td>
<td>18</td>
</tr>
<tr>
<td>honest function</td>
<td>51</td>
</tr>
<tr>
<td>ideal</td>
<td>71</td>
</tr>
<tr>
<td>of $\Sigma^*$</td>
<td>79</td>
</tr>
<tr>
<td>principal</td>
<td>69</td>
</tr>
<tr>
<td>proper</td>
<td>84</td>
</tr>
<tr>
<td>ideally effective wqo</td>
<td>74</td>
</tr>
<tr>
<td>image-finite</td>
<td>5</td>
</tr>
<tr>
<td>increasing pair</td>
<td>1, 7, 23</td>
</tr>
<tr>
<td>incrementing counter machine</td>
<td>10, 110</td>
</tr>
<tr>
<td>infinite words</td>
<td>17</td>
</tr>
<tr>
<td>ordering</td>
<td>17</td>
</tr>
<tr>
<td>interval temporal logic</td>
<td>112</td>
</tr>
<tr>
<td>Karp &amp; Miller</td>
<td>52, 13</td>
</tr>
<tr>
<td>graph</td>
<td>13</td>
</tr>
<tr>
<td>Kruskal’s Tree Theorem</td>
<td>4, 19</td>
</tr>
<tr>
<td>length function</td>
<td>28</td>
</tr>
<tr>
<td>Theorem</td>
<td>31, 47</td>
</tr>
<tr>
<td>lexicographic ordering</td>
<td>14, 15, 50, 51, 78</td>
</tr>
<tr>
<td>lexicographic product</td>
<td>15, 78</td>
</tr>
<tr>
<td>lexicographic sum</td>
<td>13, 78</td>
</tr>
<tr>
<td>linear ordering</td>
<td>1</td>
</tr>
<tr>
<td>linearisation</td>
<td>16, 40</td>
</tr>
<tr>
<td>lossy counter machine</td>
<td>55, 110</td>
</tr>
<tr>
<td>metric temporal logic</td>
<td>111</td>
</tr>
<tr>
<td>Minsky machine</td>
<td>54</td>
</tr>
<tr>
<td>multiset support</td>
<td>11</td>
</tr>
<tr>
<td>natural product</td>
<td>11</td>
</tr>
<tr>
<td>sum</td>
<td>11</td>
</tr>
<tr>
<td>Noetherian relation</td>
<td>8</td>
</tr>
<tr>
<td>norm</td>
<td></td>
</tr>
<tr>
<td>infinity norm</td>
<td>27</td>
</tr>
<tr>
<td>wqo, see nwqo</td>
<td></td>
</tr>
<tr>
<td>nwqo</td>
<td>27</td>
</tr>
<tr>
<td>derivation</td>
<td>38</td>
</tr>
<tr>
<td>empty</td>
<td>29</td>
</tr>
<tr>
<td>isomorphism</td>
<td>28</td>
</tr>
<tr>
<td>naturals</td>
<td>29</td>
</tr>
<tr>
<td>polynomial</td>
<td>29</td>
</tr>
<tr>
<td>polynomial normal form</td>
<td>30, 38</td>
</tr>
<tr>
<td>reflection</td>
<td>36</td>
</tr>
<tr>
<td>residual</td>
<td>34</td>
</tr>
<tr>
<td>singleton</td>
<td>29</td>
</tr>
<tr>
<td>$\omega$-monotone</td>
<td>85</td>
</tr>
<tr>
<td>$\omega$-node</td>
<td>22</td>
</tr>
<tr>
<td>$\omega$-set</td>
<td>85</td>
</tr>
<tr>
<td>order type</td>
<td>40</td>
</tr>
<tr>
<td>maximal</td>
<td>10, 11, 51</td>
</tr>
<tr>
<td>order-extension principle</td>
<td>16</td>
</tr>
<tr>
<td>ordinal</td>
<td></td>
</tr>
<tr>
<td>ideally effective</td>
<td>88</td>
</tr>
<tr>
<td>ideals</td>
<td>87</td>
</tr>
<tr>
<td>lean</td>
<td>93</td>
</tr>
<tr>
<td>limit</td>
<td>11, 90</td>
</tr>
<tr>
<td>ordering</td>
<td>49</td>
</tr>
<tr>
<td>pointwise ordering</td>
<td>91, 96</td>
</tr>
<tr>
<td>predecessor set</td>
<td>6</td>
</tr>
<tr>
<td>prefix ordering</td>
<td>14</td>
</tr>
<tr>
<td>prime</td>
<td>69</td>
</tr>
<tr>
<td>prime decomposition</td>
<td>70</td>
</tr>
<tr>
<td>canonical</td>
<td>71</td>
</tr>
<tr>
<td>Rado’s structure</td>
<td>16, 18, 13</td>
</tr>
<tr>
<td>ideally effective</td>
<td>88</td>
</tr>
<tr>
<td>ideals</td>
<td>87</td>
</tr>
<tr>
<td>Ramsey Theorem</td>
<td>3, 21</td>
</tr>
<tr>
<td>ranking function</td>
<td>8</td>
</tr>
<tr>
<td>reachability tree</td>
<td>5</td>
</tr>
<tr>
<td>reflection, see nwqo reflection</td>
<td>19</td>
</tr>
<tr>
<td>reflexive transitive compatibility</td>
<td>19</td>
</tr>
<tr>
<td>register automaton</td>
<td>110</td>
</tr>
<tr>
<td>relevance logic</td>
<td>8, 110</td>
</tr>
<tr>
<td>sequence</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>7, 23</td>
</tr>
<tr>
<td>controlled, see control extension</td>
<td>3</td>
</tr>
<tr>
<td>fundamental, see fundamental sequence</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>7, 25</td>
</tr>
<tr>
<td>$r$-bad</td>
<td>19</td>
</tr>
<tr>
<td>$r$-good</td>
<td>19</td>
</tr>
</tbody>
</table>
Smyth's ordering, 14, 18
sparser-than ordering, 16
star-free expression, 107
strict compatibility, 20
strict ordering, 1
subformula property, 10
subsets, see powerset
substitution, 18, 101
subword embedding, 3
successor set, 5
sum function, 48
super-homogeneous function, 46
termination, 5, 33, 56, 67
threshold, 22
timed words, 111
total ordering, 1
transfer machine, 67, 110

transition system, 1
transitive compatibility, 19
upward
  closed, 2
closure, 2

vector addition system, 12, 83, 84, 108
  with states, 4, 108
weakening, 10
well founded
  ordering, 1
  relation, 8
well partial ordering, 1
well quasi ordering, 1
well-structured transition system, 4
zero function, 48