Well-Quasi-Orders for Algorithms
MPRI Course 2.9.1 – 2018/2019

S. Demri, A. Finkel, J. Goubault-Larrecq, S. Schmitz, and Ph. Schnoebelen
LSV, ENS Cachan, CNRS, Université Paris-Saclay, 94235 Cachan, France

Lecture Notes

Work supported in part by ANR ReacHard.

CONTENTS

1

BASICS OF WQOS AND APPLICATIONS

1.1 Well Quasi Orderings

A *quasi-ordering* (some say *preorder*) on a set X is a reflexive transitive relation $\leq$. It need not be antisymmetric, allowing us to avoid quotients. An *ordering* is an antisymmetric quasi-ordering. A *quasi-order* is a set with a quasi-ordering, an *order* or a *poset* is a set with an ordering. A *qo* will denote a quasi-ordering or a quasi-order, depending on context.

quasi-ordering
preorder
ordering
quasi-order
order
poset
qo

Two elements x, y are *comparable* iff $x \leq y$ or $x \geq y$. A qo is *total*, or *linear*, iff all its elements are pairwise comparable. The usual ordering on $\mathbb{N}$ is total, the componentwise ordering on $\mathbb{N}^2$ is not.

comparable
total
linear

We write $x \equiv y$, and say that x and y are *equivalent*, iff $x \leq y$ and $y \leq x$. We write $x < y$ iff $x \leq y$ and $x \not\equiv y$.

equivalent

A binary relation $\to$ is *terminating* iff there is no infinite rewrite $x_0 \to x_1 \to \cdots \to x_n \to \cdots$. A qo $\leq$ is *well-founded* (on X) iff there is no infinite strictly descending chain $x_0 > x_1 > \cdots > x_n > \cdots$, namely iff $>$ is terminating. An *antichain* is a set of pairwise *in*comparable elements.

terminating
well-founded
antichain

Definition 1.1 (Well). A qo $\leq$ is *well* iff, for every infinite sequence $x_0, x_1, \cdots, x_n, \cdots$, there is a pair $i < j$ such that $x_i \leq x_j$. A well qo is called a *wqo*.

well
wqo

Please mind the condition $i < j$, too! A (finite or infinite) sequence satisfying that property is called *good*. All others are called *bad*. good bad

The usual ordering on $\mathbb{N}$ is well. More generally:

Lemma 1.2. *The following implications hold for a qo: total well-founded $\Rightarrow$ well $\Rightarrow$ well-founded. They are all strict implications.*

Proof. Leftmost: If $\leq$ is not well, there is a sequence x_n, $n \in \mathbb{N}$ such that $i < j$ implies $x_i \not\leq x_j$. By totality the latter is equivalent to $x_i > x_j$, and we get an infinite strictly descending chain, contradiction.

Rightmost: take an infinite strictly descending chain $x_0 > x_1 > \cdots > x_n \cdots$, by wellness $x_i \leq x_j$ for some $i < j$, which is absurd.

Counterexamples for the reverse implications: $\{0, 1\}$ with $=$ as qo, $\mathbb{N}$ with $=$ as qo. □

minimal An element $x \in A$ is *minimal* iff every $y < x$ is outside A.

Lemma 1.3. *Let $\leq$ be a well-founded qo on a subset A of X.*

minimal
1. *For every $a \in A$, there is a* minimal *$x \in A$ such that $x \leq a$.*
2. *There is an antichain $A_0 \subseteq A$ such that every element of A is $\geq$ some element of A_0.*

Proof. (1) Let $x_0 = a$. If x_0 is minimal in A, stop. Otherwise, pick an element $x_1 \in A$ such that $x_0 > x_1$. If x_1 is minimal in A, stop. Otherwise, pick $x_2 \in A$ such that $x_1 > x_2$, and proceed. The process must stop at some rank n since $\leq$ is well-founded on A. Then $x = x_n$ is minimal in A, and $x \leq a$.

(2) Let $\operatorname{Min} A$ be the set of minimal elements in A. If $\leq$ is an ordering, then we just take $A_0 = \operatorname{Min} A$, and minimality implies that this is an antichain. Otherwise, use the axiom of choice and associate each $a \in A$ with a canonical representative $f(a)$ of its equivalence class, then let $A_0 = \{f(a) \mid a \in A\}$. □

Remark 1.4. (1) fails without well-foundedness. For example, $\mathbb{Z}$ does not even have any minimal element. (1) is in fact *equivalent* to the well-foundedness of $\leq$ on A.

subsequence A *subsequence* of an infinite sequence $x_0, x_1, \cdots, x_n, \cdots$ is an infinite subsequence $x_{i_0}, x_{i_1}, \cdots, x_{i_m}, \cdots$ where (important) $i_0 < i_1 < \cdots < i_m < \cdots$. A sequence x_n, $n \in \mathbb{N}$, is *monotone* iff $x_i \leq x_j$ for *all* $i < j$. monotone

Theorem 1.5 (Characterizations of wqos, I). *The following are equivalent for a qo $\leq$:*

1. *$\leq$ is well;*
2. *every infinite sequence has a monotone subsequence;*

3. *$\leq$ is well-founded and has no infinite antichain.*

Proof. (2)⇒(1) is clear. (1)⇒(3): if $\leq$ is well, then any infinite antichain $x_n, n \in \mathbb{N}$, must contain two elements $x_i \leq x_j$, with $i < j$, hence cannot be an antichain. Moreover, $\leq$ is well-founded by Lemma 1.2.

(3)⇒(2). Fix an infinite sequence $x_n, n \in \mathbb{N}$. Define $I = \{i \mid \forall j > i, x_i \not\leq x_j\}$, $A = \{x_i \mid i \in I\}$ and let A_0 be an antichain of minimal elements of A as given in Lemma 1.3 (2). (This is where we require well-foundedness.) There is no infinite antichain, so A_0 is finite, say $A_0 = \{x_{i_1}, \cdots, x_{i_n}\}$. Let $m = \max(i_1, \cdots, i_n)$. Lemma 1.3 tells us that for every $i \in I$, there is an x_{i_k} in A_0 such that $x_i \geq x_{i_k}$. Since $i_k \in I$, $x_{i_k} \not\leq x_j$ for every $j > i_k$, so $i \not> i_k$. Therefore $i \leq i_k$. In particular, $i \leq m$, and this holds for every $i \in I$. It follows that I is finite.

Let n_0 be strictly larger than every element of I. Note that no number $\geq n_0$ can be in I. Since $n_0 \notin I$, there is an $n_1 > n_0$ such that $x_{n_0} \leq x_{n_1}$. Then $n_1 \notin I$, so there is an $n_2 > n_1$ such that $x_{n_1} \leq x_{n_2}$. Proceed ad infinitum, building the desired monotone subsequence. □

Remark 1.6. The standard proof of the hard implication (3)⇒(2) rests on Ramsey's Theorem (see (Wikipedia, 2017)) with three colors. Color the edges (i, j), with $i < j$ in $\mathbb{N}$ as follows: red if $x_i \leq x_j$, blue if $x_i > x_j$, green if x_i and x_j are incomparable. Ramsey tells us that there is an infinite set $I \subseteq \mathbb{N}$ such that all the edges (i, j), with $i < j$ in I, have the same color. Green would give us an infinite antichain, blue an infinite strictly descending chain, remains red.

A subset A is *upwards-closed* iff $x \in A$ and $x \leq y$ imply $y \in A$. We write $\uparrow A_0$ for the smallest upwards-closed subset containing A_0, namely $\{x \mid \exists a \in A_0, a \leq x\}$. We also write $\uparrow x$ for $\uparrow\{x\}$. upwards-closed

Proposition 1.7 (Characterization of wqos, II). *A qo $\leq$ is well if and only if, for every upwards-closed subset A, there is a finite antichain A_0 such that $A = \uparrow A_0$.*

If $\leq$ is an ordering, one can take $A_0 = \text{Min}\, A$. In any case, a finite set A_0 such that $A = \uparrow A_0$ is often called a *basis* of A in the literature, and one says that every upwards-closed subset has a *finite basis.* basis finite basis

Proof. If $\leq$ is well, Lemma 1.3 (2) produces an antichain $A_0 \subseteq A$. Since A is upwards-closed, $\uparrow A_0 \subseteq A$. The lemma asserts the reverse inclusion. By Theorem1.5 (3), A_0 is finite.

Conversely, consider an infinite sequence $x_n, n \in \mathbb{N}$. Let $A = \uparrow\{x_n \mid n \in \mathbb{N}\}$. There is a finite antichain $A_0 = \{x_{i_1}, \cdots, x_{i_m}\}$ (with, say, $i_1 < \cdots < i_m$) such that $A = \uparrow A_0$ by assumption. For $j = i_m + 1$, there is a k, $1 \leq k \leq m$, such that $x_{i_k} \leq x_j$. Define $i = i_k$. Then $x_i \leq x_j$, and $i < j$. □

Remark 1.8. An equivalent characterization is: $\leq$ is well iff for every set A, there is a finite subset A_0 such that every element of A is $\geq$ some element of A_0.

Corollary 1.9 (Characterization of wqos, III). *A quasi-ordering $\leq$ is well iff every infinite monotone sequence $U_0 \subseteq U_1 \subseteq \cdots \subseteq U_n \subseteq \cdots$ of upwards-closed subsets is* stationary*: there is a rank n from where U_n is constant, namely $U_n = U_{n+1} = \cdots$.*

stationary

Proof. If $\leq$ is well, then the upwards-closed subset $U = \bigcup_{n\in\mathbb{N}} U_n$ is equal to $\uparrow A_0$ for some finite antichain $A_0 = \{x_1, \cdots, x_m\}$. For each i, $1 \leq i \leq m$, there is an $n_i \in \mathbb{N}$ such that $x_i \in U_{n_i}$. Let $n = \max(n_1, \cdots, n_m)$. Then $A_0 \subseteq U_n$. Since U_n is upwards-closed, $U = \uparrow A_0 \subseteq U_n$. Therefore $U \subseteq U_n \subseteq U_{n+1} \subseteq \cdots U$, hence all the terms in that chain are equal.

Conversely, assume that every infinite monotone sequence of upwards-closed subsets is stationary, and consider a sequence of points x_n, $n \in \mathbb{N}$. The sequence $\uparrow x_0 \subseteq \uparrow\{x_0, x_1\} \subseteq \cdots \subseteq \cdots\uparrow\{x_0, x_1, \cdots, x_n\} \subseteq \cdots$ consists of upwards-closed subsets. It must therefore be stationary, say at rank n. Then $x_{n+1} \in \uparrow\{x_0, x_1, \cdots, x_n\}$. Letting $j = n + 1$, we have found $i < j$ such that $x_i \leq x_j$. ☐

downwards-closed

The *downwards-closed* subsets are defined similarly, and we write $\downarrow A_0 = \{x \in X \mid \exists a \in A_0, x \leq a\}$.

Lemma 1.10. *The downwards-closed subsets are exactly the complements of upwards-closed subsets.* ☐

Corollary 1.11 (Characterization of wqos, IV). *A qo $\leq$ is well iff the strict inclusion relation $\subsetneq$ on $\mathcal{H}(X)$, the set of downwards-closed subsets of X, is well-founded, that is, iff there is no infinite strictly descending chain $F_0 \supsetneq F_1 \supsetneq F_2 \supsetneq \cdots \supsetneq F_n \supsetneq \cdots$ of downwards-closed subsets.*

Hence wqos are a kind of second-order lifting of the notion of well-founded qos.

Remark 1.12. The apparent symmetry between upwards-closed and downwards-closed fails. Notably, the statement "every downwards-closed subset can be written $\downarrow B_0$ for some finite set B_0" is wrong. Take $\mathbb{N}$ for a counter-example. However, the Erdős-Tarski Theorem 1.32 is very close to that statement.

1.2 An algebra of well quasi-ordered domains

We will show that the following algebra of data types:

D	$::=$	A	finite qos, lemma 1.13 (1)
	$\mid$	$\mathbb{N}$	lemma 1.2
	$\mid$	$\sum_{i=1}^{n} D_i$	finite disjoint sums, lemma 1.14
	$\mid$	$\prod_{i=1}^{n} D_i$	finite products, corollary 1.16
	$\mid$	D^*	finite words, embedding, theorem 1.19
	$\mid$	$D^{\circledast}$	finite multisets, embedding, corollary 1.21
	$\mid$	$\mathcal{P}_f(D)$	finite sets, Hoare qo, Corollary 1.26
	$\mid$	$T(D)$	finite trees, homeomorphic embedding, theorem 1.29
	$\mid$	$\mathcal{G}$	finite unordered graphs, section 1.7

yields an infinite family of wqos. We start with some easy facts.

Lemma 1.13. *1. Every qo on a finite set is well.*

2. Every subset of a wqo is wqo.

3. Every extension of a wqo is wqo. (An extension $\leq'$ of $\leq$ is a qo such that $x \leq y$ implies $x \leq' y$.)

4. The image of a wqo by a surjective monotonic map is wqo. (Namely, if $\leq_X$ is wqo on X and $\leq_Y$ is a qo on Y and $f : X \to Y$ is surjective and monotonic, then $\leq_Y$ is wqo on Y.)

Proof. 1. In a finite set, every strictly descending chain and every antichain is finite.

2. Let $A \subseteq X$, and let $\leq$ be a wqo on X. For every infinite sequence x_n, $n \in \mathbb{N}$ inside A hence (trivially) in X, there are $i < j$ such that $x_i \leq x_j$.

3. If $x_i \leq x_j$ then $x_i \leq' x_j$.

4. Let $f : X \to Y$ be monotonic and surjective, X be wqo under $\leq_X$ and Y be wqo under $\leq_Y$. Consider an infinite sequence y_n, $n \in \mathbb{N}$, in Y. Since f is surjective (and the axiom of countable choice), there are elements x_n such that $y_n = f(x_n)$ for every n. By wellness, there are $i < j$ such that $x_i \leq_X x_j$. By monotonicity, $y_i \leq_Y y_j$.

□

The (finite) *sum* $\sum_{i=1}^{n} D_i$ of qos D_i is the set of pairs (i, d) with $d \in D_i$, quasi-ordered by $(i, d) \leq (j, e)$ iff $i = j$ and $d \leq e$. sum

Lemma 1.14. *If $D_1, \ldots, D_n$ are wqo then $\sum_{i=1}^{n} D_i$ is wqo.*

Proof. Fix an infinite sequence (i_m, x_m), $m \in \mathbb{N}$. By the pigeonhole principle, there is an index i, $1 \leq i \leq n$, such that $\{i_m, m \in \mathbb{N} \mid i_m = i\}$ is finite. Replacing the sequence by the infinite subsequence of elements (i_m, x_m) with $i_m = i$ if necessary, we may assume that $i_m = i$ for every $m \in \mathbb{N}$. Since D_i is wqo, there are $m < m'$ such that $x_m \leq x_{m'}$, hence $(i_m, x_m) \leq (i_{m'}, x_{m'})$. □

1.3 Finite products, Dickson's Lemma

The product of D_i, $1 \leq i \leq n$, is the set of n-tuples $(d_1, \cdots, d_n)$ where each d_i is in D_i. This is quasi-ordered componentwise: $(d_1, \cdots, d_n) \leq (e_1, \cdots, e_n)$ iff $d_i \leq_i e_i$ for every i, $1 \leq i \leq n$ (writing $\leq_i$ for the qo on D_i).

Lemma 1.15 (Dickson's Lemma). *The product of two wqos is a wqo.*

Proof. Fix an infinite sequence (x_n, y_n), $n \in \mathbb{N}$, in $X \times Y$. Extract a monotone sequence x_{n_m} ($m \in \mathbb{N}$) …from the sequence of first components x_n, $n \in \mathbb{N}$. Further extract a monotone sequence $y_{n_{m_k}}$ ($k \in \mathbb{N}$) from the sequence y_{n_m}, $m \in \mathbb{N}$. Then $(x_{n_{m_k}}, y_{n_{m_k}})$, $k \in \mathbb{N}$, is a monotone subsequence of the sequence we started with. □

By induction on n, where for $n = 0$ the empty product is a one-element set:

Corollary 1.16. *The product $\prod_{i=1}^n D_i$ of finitely many wqos is wqo.*

Corollary 1.17 (Dickson's Lemma, 1913). *For every $d \in \mathbb{N}$, the componentwise ordering on $\mathbb{N}^d$ is well.*

Dickson's original application is a result on odd perfect numbers, see Section 1.10.7. Another mathematical application is Hilbert's Basis Theorem in ring theory, see Section 1.10.8. This is computational, and we describe Buchberger's algorithm in Section 1.10.9. Its termination rests again on Dickson's Lemma.

Corollary 1.18. *Let X be a set, $\leq_1$, …, $\leq_n$ be finitely many wqos on X. The intersection relation $\leq$ defined by $x \leq y$ iff $x \leq_i y$ for every i, $1 \leq i \leq n$, is a wqo.*

Proof. Write X_i for X with the $\leq_i$ wqo. $\prod_{i=1}^n X_i$ is wqo, hence the diagonal $\Delta = \{(x_1, \cdots, x_n) \in \prod_{i=1}^n X_i \mid x_1 = \cdots = x_n\}$ is also wqo by Lemma 1.13 (2). The map $(x, \cdots, x) \in \Delta \mapsto x \in X$ is surjective and monotonic (in fact an isomorphism of qos), where X is equipped with $\leq$. We conclude by Lemma 1.13 (4). □

1.4 Finite words, and Higman's Lemma

First a useful special case. Let Σ be a finite alphabet. For finite words $w, w' \in \Sigma^*$, say that w is a (scattered) *subword* of w' iff one can obtain w' by inserting

subword

arbitrary many letters inside w, anywhere. This is a wqo, as a particular case of the following result (where $\leq$ is simply equality).

bedding

Let D be qo. The *embedding* qo $\leq_*$ on D^* is defined by $w = d_1 d_2 \cdots d_n \leq_* w'$ iff one can write w' as $w'_0 d'_1 w'_1 d'_2 w'_2 \cdots w'_{n-1} d'_n w'_n$ where each d'_i is in D and $d_i \leq d'_i$, and each w' is in Σ^*. I.e., "insert as many letters as you wish anywhere and increase any of the original letters".

A practical equivalent characterization is given by the following inductive definition, where $d, d' \in D$, $w, w' \in D^*$.

$$\frac{}{\varepsilon \leq_* w'}\,(\varepsilon) \qquad \frac{w \leq_* w'}{w \leq_* dw'}\,(add) \qquad \frac{d \leq d' \quad w \leq_* w'}{dw \leq_* d'w'}\,(inc)$$

Theorem 1.19 (Higman's Lemma (Higman, 1952a)). *If $\leq$ is wqo on D, then $\leq_*$ is wqo on D^*.*

This is actually an equivalence. The converse is left as an exercise.

Proof. We use an argument due to Nash-Williams (Nash-Williams, 1968), called the *minimal bad sequence* argument. Let $\trianglelefteq$ denote the suffix qo: $w \trianglelefteq w'$ iff w' can be written $w''w$ for some $w'' \in D^*$. Quasi-order the infinite sequences by the lexicographic extension $\trianglelefteq_{lex}$ of $\trianglelefteq$. This is *not* a well-founded qo (despite the fact that $\trianglelefteq$ is well-founded), however the following argument shows that the set Bad of infinite bad sequences, if non-empty, must have a minimal element. Consider the set of first elements of infinite sequences in Bad. If Bad is not empty, that must have a minimal element w_0 since $\trianglelefteq$ is well-founded. The set of second elements of those infinite bad sequences that start with w_0 must also have a minimal element w_1. The set of third elements of those infinite bad sequences that start with w_0, w_1 has a minimal element w_2. Continuing this way, we obtain an infinite sequence $w_0, w_1, \cdots, w_n, \cdots$. If we had $w_i \leq w_j$ for some $i < j$, then our choice of w_j would necessarily have been from a good sequence, which is impossible. Hence $w_0, w_1, \cdots, w_n, \cdots$ is bad.

minimal bad sequence

It is *minimally bad* in the sense that any strictly smaller infinite sequence must be good, in the $\trianglelefteq_{lex}$ qo. Explicitly, any infinite sequence of the form $w_0, w_1, \cdots, w_{n-1}, , w', \cdots$ with $w' \triangleleft w_n$ must be good. (We write $\triangleleft$ for the strict part of $\trianglelefteq$.)

minimally bad

Now fix a minimal bad sequence of words w_n. No w_n can be empty, since any sequence containing ε is good, using rule (ε). Hence we can write w_n as $d_n w'_n$ for some $d_n \in D$, $w'_n \in D^*$.

Since $\leq$ is wqo on D, the infinite sequence of first letters $d_0, d_1, \cdots, d_n, \cdots$ has a monotone subsequence $d_{n_0} \leq d_{n_1} \leq \cdots$, with $n_0 < n_1 < \cdots$. Consider the new infinite sequence of words, whose first n_0 terms are w_0, w_1, …, w_{n_0-1}, followed by w'_{n_0}, w'_{n_1}, …. Notice that that new sequence is strictly smaller that the one we started with in $\trianglelefteq_{lex}$, so it must be good by minimality. Hence some word embeds into some later word in that sequence. This splits into three cases, depending on the positions of those two words relative to n_0.

1. Both are before n_0: we find $w_i \leq_* w_j$ for some $i < j < n_0$. That is impossible because the sequence we started from was bad.

2. One is before n_0, the other one is not: $w_i \leq_* w'_{n_j}$ for some $i < n_0$ and some $j \geq 0$. By (add), $w_i \leq_* d_{n_j} w'_{n_j} = w_{n_j}$, and $i < n_0$ implies $i < n_j$. Again that contradicts the fact that the sequence we started from was bad.

3. Both are at or after position n_0: $w'_{n_i} \leq_* w'_{n_j}$ for some $i < j$, then, using $d_{n_i} \leq d_{n_j}$ and rule (inc), we obtain $w_{n_i} = d_{n_i} w'_{n_i} \leq_* d_{n_j} w'_{n_j} = w_{n_j}$: contradiction again.

□

Example 1.20 (Subword ordering)**.** We use ε to denote the empty sequence. Over $(\Sigma, =)$, where $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ is a 3-letter alphabet and where different letters are incomparable, the word $\mathtt{abb}$ is a subword of $\mathtt{c}\underline{\mathtt{ab}}\mathtt{ca}\underline{\mathtt{b}}$, as witnessed by the underlined letters, and written $\mathtt{abb} \leq_* \mathtt{cabcab}$. On the other hand $\mathtt{bba} \not\leq_* \mathtt{cabcab}$. Over $(\mathbb{N}, \leq)$, examples are $\varepsilon \leq_* 4{\cdot}1{\cdot}3 \leq_* 1{\cdot}5{\cdot}0{\cdot}3{\cdot}3{\cdot}0{\cdot}0$ and $4{\cdot}1{\cdot}3 \not\leq_* 1{\cdot}5{\cdot}0{\cdot}3{\cdot}0{\cdot}0$. Over $(\mathbb{N}^2, \leq_\times)$, one checks that $\binom{0}{1}\cdot\binom{2}{0}\cdot\binom{0}{2} \not\leq_* \binom{2}{0}\cdot\binom{0}{2}\cdot\binom{0}{2}\cdot\binom{2}{2}\cdot\binom{2}{0}\cdot\binom{0}{1}\cdot\binom{1}{0}$.

1.5 Finite multisets, finite sets

multiset

A (finite) *multiset* over D is a finite word up to permutation of the letters. Equivalently, this is a map $m\colon D \to \mathbb{N}$ such that $m(d) = 0$ except for finitely many $d \in D$: intuitively, $m(d)$ counts how many times the letter d occurs in the given word. Let $D^{\circledast}$ be the set of finite multisets over D. The intuition is formalized by the *Parikh mapping* $\varphi\colon D^* \to D^{\circledast}$, defined by: $\varphi(w)(d)$ is the number of occurrences of d in the word w.

Parikh mapping

The empty multiset $\emptyset$ is the constant 0 function, and multiset union $\uplus$ is pointwise addition. We write $\{\!|d_1, d_2, \cdots, d_n|\!\}$ for $\varphi(d_1 d_2 \cdots d_n)$. The *embedding* qo $\leq_{\circledast}$ on $D^{\circledast}$ is defined inductively by:

embedding

$$\frac{}{\emptyset \leq_{\circledast} m'}\,(\emptyset) \qquad \frac{m \leq_{\circledast} m'}{m \leq_{\circledast} m' \uplus \{\!|d|\!\}}\,(add) \qquad \frac{d \leq d' \quad m \leq_{\circledast} m'}{m \uplus \{\!|d|\!\} \leq_{\circledast} m' \uplus \{\!|d'|\!\}}\,(inc)$$

That is in fact the image qo of $\leq_*$ by φ. Clearly φ is surjective and monotonic. By Lemma 1.13 (4), we obtain:

Corollary 1.21. *If $\leq$ is wqo on D, then $\leq_{\circledast}$ is wqo on $D^{\circledast}$.*

multiset extension

The qo $\leq_{\circledast}$ is in general different from the so-called *multiset extension* $<_{mul}$ of a strict ordering $<$. That has many definitions. One is $m <_{mul} m'$ iff one can write m as $m_0 \uplus m_1$, m' as $m_0 \uplus m'_1$, and for each $d \in m_1$ (i.e., such that $m_1(d) \neq 0$), there is a $d' \in m'_1$ such that $d < d'$. Equivalently, $>_{mul}$ (note the reversal, by which we mean the opposite of $<_{mul}$, not the multiset extension of $>$) is the transitive closure $\rightarrow^+$ where $m \uplus \{\!|d|\!\} \rightarrow m \uplus \{\!|d_1, \cdots, d_n|\!\}$ for all $d > d_1, \cdots, d_n$, $n \in \mathbb{N}$ ("replace any element by arbitrarily many smaller elements").

With either definition, one checks that, if $\leq$ is an ordering, then $m \leq_{\circledast} m'$ implies $m = m'$ or $m <_{mul} m'$. By Lemma 1.13 (3):

Proposition 1.22. *If $\leq$ is an ordering and a wqo on D, then $= \cup <_{mul}$ is wqo on $D^{\circledast}$.*

Finally, let $\sigma\colon D^{\circledast} \rightarrow \mathcal{P}_f(D)$ map each multiset $\{\!|d_1, \cdots, d_n|\!\}$ to its set of elements $\{d_1, \cdots, d_n\}$. Quasi-order the set $\mathcal{P}_f(D)$ of finite subsets of D by the *Hoare qo* $\leq^\flat$: $E \leq^\flat E'$ iff for every $d \in E$, there is a $d' \in E'$ such that $d \leq d'$. Clearly, σ is monotonic and surjective. By Lemma 1.13 (4), we obtain:

Hoare qo

Proposition 1.23. *If $\leq$ is wqo on D, then $\leq^\flat$ is wqo on $\mathcal{P}_f(D)$.*

The Hoare qo of course extends to all, even infinite, subsets, and is very natural:

Lemma 1.24. *Let D be a qo. For all $A, B \subseteq D$, $A \leq^\flat B$ iff $\downarrow A \subseteq \downarrow B$.*

Remark 1.25. One may consider any finite set as a multiset, where each element occurs exactly once. Let $\leq$ be an ordering with strict part $<$. Although $\leq_{\circledast}$ is in general different from $= \cup <_{mul}$, on finite sets, they coincide. This follows from our first definition of the multiset extension.

Let $\mathcal{H}_{\text{fin}}(D)$ denote the set of *finitary* downwards-closed subsets, namely those of the form $\downarrow A$ with A finite.

finitary

Corollary 1.26. *If $\leq$ is wqo on D, then $\subseteq$ is wqo on $\mathcal{H}_{fin}(D)$.*

Those last results do not extend to infinite/infinitary subsets, namely to $\mathbb{P}(D)$ or to $\mathcal{H}(D)$. The canonical counterexample is the following (Rado, 1954):

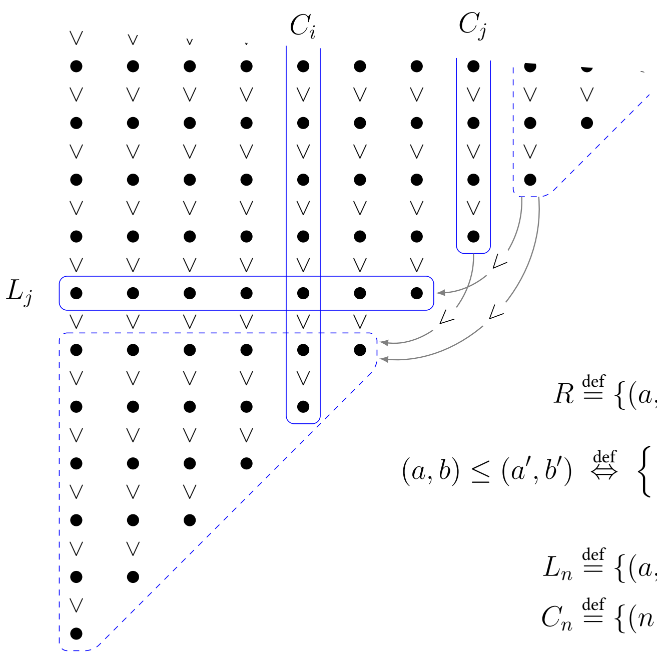
Definition 1.27 (Rado's Structure). *Rado's structure R* is $\{(m, n) \in \mathbb{N}^2 \mid m < n\}$, with the ordering $\leq_R$ defined by $(m, n) \leq_R (m', n')$ iff $m = m'$ and $n \leq n'$, or $n < m'$.

Rado's structure

Theorem 1.28. *1. R is wqo.*

2. $\mathbb{P}(R)$ under $\leq^\flat$, $\mathcal{H}(R)$ under inclusion, are not.

Proof. (1) Checking that $\leq_R$ is transitive is a tedious case analysis. Reflexivity is trivial. Assume an infinite bad sequence (m_i, n_i), $i \in \mathbb{N}$, in R. By Dickson's Lemma our sequence has a subsequence that is monotone with respect to $\leq \times \leq$ (not $\leq_R$!). Since any subsequence of a bad sequence is bad, we may as well replace



$$R \overset{\text{def}}{=} \{(a,b) \in \mathbb{N}^2 \mid a < b\}$$

$$(a,b) \leq (a',b') \overset{\text{def}}{\Leftrightarrow} \begin{cases} & a = a' \wedge b \leq b' \\ \vee & b < a' \end{cases}$$

$$L_n \overset{\text{def}}{=} \{(a,n) \in R \mid a < n\}$$
$$C_n \overset{\text{def}}{=} \{(n,b) \in R \mid n < b\}$$

Rado's structure $(R, \leq)$.

---

the original sequence by a bad one, and assume that $m_0 \leq m_1 \leq \cdots \leq m_i \leq \cdots$ and $n_0 \leq n_1 \leq \cdots \leq n_i \leq \cdots$. If we had $m_i = m_{i+1}$ for some i, then $(m_i, n_i) \leq_R (m_{i+1}, n_{i+1})$ would ensue, contradiction. Hence $m_i < m_{i+1}$ for every i. In particular, there are terms m_j as large as we wish. Take one such that $m_j > n_0$. Then $(m_0, n_0) \leq_R (m_j, n_j)$, contradiction.

(2) For every $i \in \mathbb{N}$, let $\omega_i = \{(i,n) \mid i < n\} \cup \{(m,n) \in R \mid n \leq i-1\}$. One checks that ω_i is downwards-closed. (We will explain in Section 1.8 where that comes from.) If $\mathcal{H}(R)$ were wqo, there would be indices $i < j$ such that $\omega_i \subseteq \omega_j$. However (i,j) belongs to ω_i but not to ω_j: contradiction. The same argument serves for $\mathbb{P}(X)$ since for downwards-closed subsets A and B, $A \subseteq B$ iff $A \leq^\flat B$. $\square$

1.6 Kruskal's Tree Theorem

If D is a signature, thought of consisting of function symbols of variable arity, one can form the set $T(D)$ of finite trees (a.k.a., *terms*) over D inductively by: for any finite list of terms $t_1, \ldots, t_n$, for every $f \in D$, $f(t_1, \cdots, t_n)$ is a term. The base case is given by the case $n = 0$. In that case, we write f for $f()$.

terms

Since lists $(t_1, \cdots, t_n)$ of terms are just words $\mathbf{t} = t_1 \cdots t_n$ in $T(D)^*$, we shall also agree that $f(\mathbf{t})$ denotes $f(t_1, \cdots, t_n)$. This will allow us to use Higman's qo on lists of arguments.

The *homeomorphic embedding* ordering $\leq_T$ is defined inductively by:

homeomorphic embedding

$$\frac{s \leq_T t_i}{s \leq_T f(t_1 \cdots t_n)} \;(T\text{-}add) \qquad \frac{f \leq g \quad \mathbf{s}\ (\leq_T)_* \ \mathbf{t}}{f(\mathbf{s}) \leq_T g(\mathbf{t})} \;(T\text{-}inc)$$

where i is such that $1 \leq i \leq n$ in $(T\text{-}add)$.

Theorem 1.29 (Kruskal (Kruskal, 1960)). *If $\leq$ is wqo on D, then $\leq_T$ is wqo on $T(D)$.*

This is actually an equivalence. The converse is left as an exercise.

Proof. We use Nash-Williams' minimal bad sequence argument, plus an additional trick.

Let $\trianglelefteq$ denote the subterm ordering, defined inductively by: $t \trianglelefteq t$, and if $s \trianglelefteq t_i$ for some i, $1 \leq i \leq n$, then $s \trianglelefteq f(t_1 \cdots t_n)$. As in the proof of Theorem 1.19, we assume that the set Bad of infinite bad sequences in $T(D)$ is non-empty, and then, it has a minimal element with respect to $\trianglelefteq_{lex}$, say $t_0, t_1, \cdots, t_{n-1}, \cdots$.

Write each t_n as $f_n(\mathbf{s}_n)$. The additional trick consists in considering the set U of all elements s_{nk} of some $\mathbf{s}_n$, $n \in \mathbb{N}$ (explicitly, writing $\mathbf{s}_n$ as $s_{n1}s_{n2}\cdots s_{nm_n}$, $U = \{s_{nk} \mid n \in \mathbb{N}, 1 \leq k \leq m_n\}$), and to note that $\leq_T$ must be wqo on U. Indeed, imagine U has an infinite bad sequence $s_{n_0k_0}, s_{n_1k_1}, \cdots, s_{n_ik_i}, \cdots$. Note that those are pairwise distinct terms, otherwise the sequence would be good. In particular, each term in that sequence can be written $s_{n_ik_i}$ for a unique index i, and therefore n_i is uniquely determined from $s_{n_ik_i}$. From that bad sequence, remove all those terms $s_{n_ik_i}$ such that $i \geq 1$ and $n_i < n_0$, i.e., that are elements of the finitely many finite lists $\mathbf{s}_0$, $\mathbf{s}_1$, …, $\mathbf{s}_{n_0-1}$. Since we remove only finitely many terms this way, and since subsequences of bad sequences are bad, we still have an infinite bad sequence. Without loss of generality, we can therefore assume that we have an infinite bad sequence $s_{n_0k_0}, s_{n_1k_1}, \cdots, s_{n_ik_i}, \cdots$ such that $n_i \geq n_0$ for every $i \geq 1$. We now form the infinite sequence $t_0, t_1, \cdots, t_{n_0-1}, s_{n_0k_0}, s_{n_1k_1}, \cdots, s_{n_ik_i}, \cdots$. That is strictly smaller than our original sequence in $\trianglelefteq_{lex}$, hence it must be good: some term must be $\leq_T$ some later term. These two terms cannot be both at indices $< n_0$ since the sequence of terms t_n is bad. They cannot be both at indices $\geq n_0$, since the sequence of terms $s_{n_ik_i}$ is bad. Therefore, there is an $i < n_0$ and and a $j \in \mathbb{N}$ such that $t_i \leq_T s_{n_jk_j}$. By rule $(T\text{-}add)$, $t_i \leq_T t_{n_j}$. Since we have made sure that $n_j \geq n_0$, and $i < n_0$, we have $i < n_j$, contradicting the fact that the sequence of terms t_n is bad.

Using Higman's Lemma (Theorem 1.19) and Lemma 1.15, $D \times U^*$ is wqo under $\leq \times \leq_*$. We can therefore find $i < j$ with $f_i \leq f_j$ and $\mathbf{s}_i\ (\leq_T)_*\ \mathbf{s}_j$. By rule $(T\text{-}inc)$, $t_i = f_i(\mathbf{s}_i) \leq_T f_j(\mathbf{s}_j) = t_j$, contradiction. □

Using Lemma 1.13, one obtains the following corollaries. If $\leq$ is wqo on D, then:

1. $\leq_T$ is wqo on the set of first-order terms over a given signature (specifying arities for each function symbol);

2. $\leq_T$ is wqo on the set of trees with unordered successors (i.e., the arguments of function symbols are multisets, instead of lists, of terms).

1.7 Graphs

The Robertson-Seymour Theorem (Robertson and Seymour, 2004) states that (finite, undirected, without self-loops) graphs are well-quasi-ordered under *minors*. minors This is a landmark result, and a very difficult one: the cited paper is the culmination of 20 papers leading to that theorem, published over roughly 20 years and totally about 500 pages.

The reader is directed towards supplementary notes to an MPRI course by Pierre Charbit for additional material (Charbit, 2016).

Contracting an edge $\{x, y\}$ in a graph G means removing it from G and equating x with y. G is a *minor* of H, in notation $G \sqsubseteq H$, iff one can obtain G from H by contracting edges, by removing edges, and by removing isolated vertices. minor Alternatively, $G \sqsubseteq H$ iff one can find a connected subgraph H_v of H for each vertex v of G in such a way that for all $v \neq v'$ in G, H_v and $H_{v'}$ have disjoint sets of vertices but are connected by at least one edge.

Theorem 1.30 (Robertson-Seymour). *The minor relation $\sqsubseteq$ is wqo on the set $\mathcal{G}$ of undirected graphs.* □

The theorem extends to the case of vertex-labeled graphs. In other words, consider finite undirected graphs with vertices labeled by elements of some set X with a wqo $\leq$.

Theorem 1.30 does not extend to various other qos on $\mathcal{G}$, such as the subdivision-of ordering or the topological-minor relation. It fails on *directed* graphs as well.

Theorem 1.30 was inspired by Wagner's Theorem: a finite graph is planar iff it has neither K_5 nor $K_{3,3}$ as a minor. K_5 is the complete graph on 5 vertices, and $K_{3,3}$ is a 6-vertex bipartite graph where each element of the first 3-element part has a common edge with every element of the second part.

Note that every minor of a planar graph is planar. Hence the family of non-planar graphs is upwards-closed in $\mathcal{G}$ with the minor relation. By Proposition 1.7, Theorem 1.30 implies that there is a finite family A_0 of graphs such that $G \in \mathcal{G}$ is non-planar iff it has some element of A_0 as minor. Taking complements, G is planar iff it has no element of A_0 as minor. Wagner had shown that we could take $A_0 = \{K_5, K_{3,3}\}$.

By Remark 1.8, Theorem 1.30 implies that *every* minor-closed family A of graphs has a similar property (minor-closed means downwards-closed in $\sqsubseteq$): there is a finite family A_0 of graphs—the so-called *forbidden minors*—such that for every forbidden minors

$G \in \mathcal{G}$, G is in A iff it has no graph of A_0 as minor.

Robertson and Seymour also showed that testing whether G is a minor of H can be done in polynomial time (Robertson and Seymour, 1995) *when G is fixed*— the input is H, only. On the contrary, beware that testing whether $G \sqsubseteq H$ when both G and H are given as input is **NP**-complete. Indeed, in the special case where G is a cycle of length N, where N is the number of vertices in H, $G \sqsubseteq H$ iff one can obtain G from H by edge deletions only, iff H has a Hamiltonian cycle.

The polynomial time algorithm mentioned above is cubic. That was improved to quadratic (Kawarabayashi et al., 2012). This implies that planarity can be tested in quadratic time. (Planarity can be tested in linear time by a different technique.) More generally, this implies that for *every* minor-closed family A of graphs, testing membership in A can be done in quadratic time. Just test, given the graph G given as input, whether some element of A_0 is a minor of G.

That is paradoxical. The following example is given by Fellows and Langston (Fellows and Langston, 1988). Consider the 3D linkless embedding problem: given a graph G as input, can G be embedded in a 3-dimensional space such that no two disjoint cycles are topologically linked? (Imagine links of a chain.) The family of graphs for which the answer is yes is minor-closed, hence this problem can be solved in quadratic time. But, although the algorithm exists, we do not *know* it. Indeed, we do not know the set A_0 of forbidden minors for that problem. The best *known* algorithm takes exponential time, even for checking that two embeddings of cycles in 3-dimensional space are topologically linked.

1.8 Ideals, and completions

Ideals and completions will be the subject of Chapter 4. We lay out the basic theory here nonetheless.

A subset A of a qo X is *directed* iff every finite subset of A has an upper bound in A. Equivalently, A is directed iff: directed

1. A is non-empty;
2. and for every pair of elements $x, y \in A$, there is a third element z in A such that $x, y \leq z$.

The notion is used in domain theory, notably, that is, the foundation of denotational semantics. A *dcpo* is a poset where every directed family has a least upper bound, and a map $f\colon X \to Y$ between dcpos is *Scott-continuous* iff: dcpo Scott-continuous

1. f is monotonic;
2. and f preserves directed suprema, in the sense that for every directed family $(x_i)_{i\in I}$ in X, $\sup_{i\in I} f(x_i) = f(\sup_{i\in I} x_i)$.

An *ideal* of a qo X is a subset that is directed and downwards-closed. ideal

A subset A of a qo X is *irreducible* iff it is non-empty, and for all downwards-closed subsets D_1, D_2 such that $A \subseteq D_1 \cup D_2$, A is already contained either in D_1 or in D_2. Using Lemma 1.10, A is irreducible iff it is non-empty and for any two upwards-closed subsets U_1, U_2, if A intersects each of U_1, U_2 then it intersects $U_1 \cap U_2$. Every one-point set is irreducible. Every set of the form $\downarrow x$ is irreducible. In general: irreducib

Lemma 1.31. *In a qo X, the ideals are exactly the irreducible downwards-closed subsets of X.*

Proof. Let D be downwards-closed. If D is directed, then it is non-empty, and it is irreducible: imagine D intersects U_1 at x_1 and U_2 at x_2; by directedness, there is a point $z \in D$ above x_1 and x_2, and since U_1 and U_2 are upwards-closed, z is in $U_1 \cap U_2$.

Conversely, let D be irreducible. Consider $x_1, x_2 \in D$. Hence D intersects $\uparrow x_1$ and $\uparrow x_2$, and by irreducibility, also $\uparrow x_1 \cap \uparrow x_2$: any point z in the intersection is in D and above x_1 and x_2. □ □

principal ideal Every subset of the form $\downarrow x$ is an ideal, called the *principal ideal* (at x). In general, there are others. E.g., as soon as there is an infinite strictly increasing chain x_n, $n \in \mathbb{N}$, then $\bigcup_{n \in \mathbb{N}} \downarrow x_n$ is a non-principal ideal.

Let $\mathbb{N}_\omega$ be $\mathbb{N}$ plus a fresh element ω larger than all others. The ideals of $\mathbb{N}^k$ are exactly the sets of the form $\mathbb{N}^k \cap \downarrow z$ where $z \in \mathbb{N}^k_\omega$. For example, the non-principal ideal $\{(x_0, x_1, x_2) \in \mathbb{N}^3 \mid x_0 \leq 4, x_2 \leq 3\} = \mathbb{N}^3 \cap \downarrow (4, \omega, 3)$.

The ideals of the Rado structure R are the principal ideals $\downarrow_R (m, n)$, the subsets ω_i introduced in the proof of Theorem 1.28 (2), $i \in \mathbb{N}$, and R itself.

The following finds its source in (Erdős and Tarski, 1943), where it is implicit. This was rediscovered many times since then.

Theorem 1.32. *If X is wqo, then every downwards-closed subset of X is a* finite *union of ideals.*

Proof. Imagine the conclusion fails. Let A be the set of those $D \in \mathcal{H}(X)$ such that D is not a finite union of ideals. Then A is not empty. Since $\subsetneq$ is well-founded on $\mathcal{H}(X)$ (Corollary 1.11), we can apply Lemma 1.3 (1) and obtain a minimal element D of A. Since D is not a finite union of ideals, it is in particular not empty.

We claim that D is irreducible. Let D_1, D_2 be two downwards-closed subsets of X such that D is contained in neither. We wish to show that D is not contained in $D_1 \cup D_2$. By minimality of D in A, each set $D \cap D_i$ *can* be expressed as a finite union of ideals. Therefore $(D \cap D_1) \cup (D \cap D_2) = D \cap (D_1 \cup D_2)$ is also a finite union of ideals. Since D is not, D is different from $D \cap (D_1 \cup D_2)$, meaning $D \not\subseteq D_1 \cup D_2$.

Since D is irreducible, it is an ideal by Lemma 1.31, contradicting $D \in A$. □

More is known: the qos such that every downwards-closed subset is a finite union of ideals are exactly those that have no infinite antichain (Kabil and Pouzet, 1992b, Lemma 5.3).

Let *Idl*(X) denote the *ideal completion* of X. This is the set of ideals of X, ordered by inclusion $\subseteq$. One checks that *Idl*(X) is a dcpo, where directed sups are computed as unions. The important thing to check here is that the union of a *directed* family of ideals is itself directed.

There is a monotonic map $\eta\colon X \to$ *Idl*(X) which maps x to $\downarrow x$. When X is a poset ($\leq$ is antisymmetric) this is an *order-embedding*, namely $x \leq y$ *if and only if* $\eta(x) \leq \eta(y)$. In that case, we may see X as a subposet of *Idl*(X), and *Idl*(X) as a completion of X, justifying the name.

Proposition 1.33. *Idl*(X) *is the free dcpo over the qo* X. *This means that every monotonic map* $f\colon X \to Y$ *to a dcpo* Y *extends to a unique Scott-continuous map* $\hat{f}\colon$ *Idl*$(X) \to Y$, *where 'extends' means* $\hat{f} \circ \eta = f$.

Proof. If $\hat{f}$ exists, then $\hat{f}(\downarrow x) = f(x)$ for every x. Consider an arbitrary ideal I. Then I is the directed union (=supremum) of the ideals $\downarrow x$, $x \in I$. Scott-continuity implies that $\hat{f}(I) = \sup_{x\in I} f(x)$. This shows uniqueness.

Define $\hat{f}$ by that formula. This defines a monotonic map, and we need to show preservation of directed suprema. Let $(I_k)_{k\in K}$ be a directed family of ideals, $I = \bigcup_{k\in K} I_k$. Note that $x \in I$ iff there exists $k \in K$ such that $x \in I_k$. Hence $\hat{f}(I) = \sup_{\exists k\in K, x\in I_k} f(x) = \sup_{k\in K} \sup_{x\in I_k} f(x) = \sup_{k\in K} \hat{f}(I_k)$. □

Assume X wqo. *Idl*(X) may fail to be wqo: Rado's structure R is a counterexample. One may either realize that the downwards-closed subsets ω_i constructed in the proof of Theorem 1.28 are ideals of R, or show that, for a wqo X, *Idl*(X) is a wqo iff $\mathcal{H}(X)$ is a wqo, using Theorem 1.32. However, *Idl*(X) is Noetherian (see Section 1.11.1) in the so-called Scott topology. Also, if X is bqo then *Idl*(X) is bqo, see Section 1.11.2, and bqo implies wqo. We shall see that all the data types of Section 1.2, except possibly $\mathcal{G}$, are bqo.

1.9 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* $\langle S, \to\rangle$, where S is a set of states—not constrained to be finite—and $\to\, \subseteq S \times S$ is a transition relation, further endowed with a wqo $\leq\, \subseteq S \times S$ that satisfies a *compatibility* condition:

well-structured transition system

transition system

compatibility

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

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

Several variants of the basic WSTS notion exist (backward compatibility, strict compatibility, …) and we shall mention some of them in exercises 1.17 to 1.20.

vector addition system with states

Example 1.34. A d-dimensional *vector addition system with states* (VASS) is a finite-state system that manipulates d counters with only increment and decrement operations. Formally, it is a tuple $\mathcal{V} = \langle Q, \delta, q_0, \mathbf{x}_0 \rangle$ where Q is a finite set of states, $\delta \subseteq Q \times \mathbb{Z}^d \times Q$ is a finite set of translations, q_0 in Q is an initial state, and $\mathbf{x}_0$ in $\mathbb{N}^d$ describes the initial counter contents.

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

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

1.9.1 Termination

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

Proof. The direct implication follows from the definition of wqos 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*

successor set

$$Post(s) \stackrel{\text{def}}{=} \{s' \in S \mid s \rightarrow s'\} \tag{1.1}$$

ge-finite
effective

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

Proof. The algorithm consists of two semi-algorithms. The first one attempts to prove termination and builds a *reachability tree* starting from s_0; if $\mathcal{S}$ terminates from s_0, then every branch of this tree will be finite, and since $\mathcal{S}$ is image-finite this tree is also finitely branching, hence finite overall by Kőnig's Lemma. The second one attempts to prove non-termination, and looks nondeterministically for a finite witness matching Lemma 1.35.

reachability tree

This proof displays a useful decidability argument: find two semi-algorithms, each deciding one side of the question. In this particular case, we can combine the two semi-algorithms in one. We start a reachability tree as above. Every leaf of the current tree is the end of a branch $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_i$. We add all elements s of *Post*(s_i) as successors to s_i, unless $s_j \leq s$ for some $j \leq i$. In that case we stop and answer "non-termination". If $\mathcal{S}$ does not terminate from s_0, then we will eventually find ourselves in such a situation, provided we use a fair leaf selection process. Otherwise we will build a finite reachability tree, as above, and answer "termination". □

1.9.2 Coverability

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

[Cover] Coverability

coverability

instance: A transition system $\langle S, \rightarrow \rangle$, 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 $x \leq_A x'$.

control-state reachability

The decidability of Cover for WSTS uses a *set-saturation method*, whose termination relies on (Characterization of wqos, III). 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*

backward coverability

predecessor set

$$Pre_\exists(U) \overset{\text{def}}{=} \{s \in S \mid \exists s' \in U,\, s \rightarrow s'\}\ . \tag{1.2}$$

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

$$U = U_0 \subseteq U_1 \subseteq \cdots \text{ where } U_{n+1} \stackrel{\text{def}}{=} U_n \cup \mathit{Pre}_\exists(U_n) \ . \tag{1.3}$$

There is no reason for (1.3) to converge in general, but for WSTSs, this can be solved when U is upwards-closed:

Lemma 1.37. *If $U \subseteq S$ is an upwards-closed set of states, then $\mathit{Pre}_\exists(U)$ is upwards-closed.*

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

A corollary is that the sequence (1.3) stabilises to $\mathit{Pre}_\exists^*(U)$ after a finite amount of time thanks to (Characterization of wqos, III). 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 $\mathit{pb}(s)$, a finite basis of $\uparrow\mathit{Pre}_\exists(\uparrow\{s\})$.[1]

effective pred-basis

Proposition 1.38 (Decidability of Coverability for WSTSs). *Let $\mathcal{S} = \langle S, \to, \leq\rangle$ be a WSTS and s, t be two states in S. If $\mathcal{S}$ has effective pred-basis and decidable $\leq$, then coverability of t from s in $\mathcal{S}$ is also decidable.*

Proof. Compute a finite basis B for $\mathit{Pre}_\exists^*(\uparrow\{t\})$ using sequence (1.3) 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.

1.10 Examples of Applications

Let us present applications of wqos in several different areas. We do not present just applications in computer science, but some in mathematics as well.

1.10.1 Program Termination

Bad Sequences and Termination. Recall 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 also recall that (finite or infinite) sequences with an increasing pair $a_{i_1} \leq a_{i_2}$ are *good* sequences, and call *bad* a sequence where no such increasing pair can be found. Therefore every infinite sequence over the wqo A is good, i.e., bad sequences over A are finite.

good sequence

bad sequence

In order to see how bad sequences are related to termination, consider the simple program presented in Figure 2.1. We can check that every run of simple

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

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

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

terminates, this for any choice of initial values $\langle a_0, b_0 \rangle$ of a and b. Indeed, we can consider any sequence

$$\langle a_0, b_0, c_0 \rangle, \ldots, \langle a_j, b_j, c_j \rangle, \ldots \tag{1.4}$$

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

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

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 (1.5) is a bad sequence over $(\mathbb{N}^2, \leq)$, and is therefore a finite sequence. Consequently, (1.4) is also finite, which means that SIMPLE always terminates.

well-founded relation
Noetherian relation

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

$$\begin{aligned}\langle a, b, c\rangle \; R \; \langle a', b', c'\rangle \text{ iff } a > 0 \land b > 0 \land ((a' = a - 1 \land b' = b \land c' = 2c) \quad (1.6)\\ \lor (a' = 2c \land b' = b - 1 \land c' = 1)) \; .\end{aligned}$$

ranking function

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

$$R \subseteq \{(x_i, x_j) \mid \rho(x_i) > \rho(x_j)\} \; . \tag{1.7}$$

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

This is roughly what we did in (1.5), by projecting away the third component and using $\mathbb{N}^2$ as codomain; this does not satisfy (1.7) for the product ordering $(\mathbb{N}^2, \leq)$, but it does satisfy it for the lexicographic ordering $(\mathbb{N}^2, \leq_{\text{lex}})$. Equivalently, one could define $\rho \colon \mathbb{Z}^3 \to \omega^2$ by $\rho(a, b, c) = \omega \cdot b + a$ if $a, b \geq 0$ and $\rho(a, b, c) = 0$ otherwise.

However our argument with (1.5) was rather to use bad sequences: we rather require ρ to have a wqo $(A, \leq)$ as co-domain, and check that the *transitive closure* R^+ of R verifies

$$R^+ \subseteq \{(x_i, x_j) \mid \rho(x_i) \not\leq \rho(x_j)\} \tag{1.8}$$

instead of (1.7). Again, (1.8) 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 (1.7) for well-founded orders or (1.8) for wqos. As it turns out that finding an adequate ranking function is often the hardest part, this second option might be preferable.

disjunctive termination argument

Transition Invariants. A 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

$$R^+ \subseteq T_1 \cup \cdots \cup T_k \; . \tag{1.9}$$

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

$$T_1 = \{(\langle a, b, c\rangle, \langle a', b', c'\rangle) \mid a > 0 \land a' < a\} \tag{1.10}$$

$$T_2 = \{(\langle a, b, c\rangle, \langle a', b', c'\rangle) \mid b > 0 \land b' < b\} \tag{1.11}$$

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

Let us prove the well-foundedness of R when each of the T_j is proven well-founded thanks to a ranking function ρ_j into some wqo $(A_j, \leq_j)$ (see Exercise 1.23 for a generic proof that only requires each T_j to be well-founded). Then with a sequence

$$x_0, x_1, \ldots \tag{1.12}$$

of program configurations one can associate the sequence of tuples

$$\langle \rho_1(x_0), \ldots, \rho_k(x_0) \rangle, \langle \rho_1(x_1), \ldots, \rho_k(x_1) \rangle, \ldots \tag{1.13}$$

in $A_1 \times \cdots \times A_k$, the latter being a wqo for the product ordering by Dickson's Lemma. Since for any indices $i_1 < i_2$, $(x_{i_1}, x_{i_2}) \in R^+$ is in some T_j for some $1 \leq j \leq k$, we have $\rho_j(x_{i_1}) \not\leq_j \rho_j(x_{i_2})$ by definition of a ranking function. Therefore the sequence of tuples is bad for the product ordering and thus finite, and the program terminates. (See Exercise 1.22.)

Different strategies can be used in practice to find a disjunctive termination invariant of the form (1.9). One that works well in the example of SIMPLE is to use the structure of the program relation R: if R can be decomposed as a union $R_1 \cup \cdots \cup R_k$, then applying rank function synthesis to each R_j, thereby obtaining a well-founded over-approximation $\mathrm{wf}(R_j) \supseteq R_j$, provides an initial candidate termination argument

$$\mathrm{wf}(R_1) \cup \cdots \cup \mathrm{wf}(R_k)\ . \tag{1.14}$$

Applying this idea to SIMPLE, we see that R in (1.6) is the union of

$$R_1 = \{(\langle a, b, c \rangle, \langle a', b', c' \rangle) \mid a > 0 \wedge b > 0 \wedge a' = a - 1 \wedge b' = b \wedge c' = 2c\} \tag{1.15}$$

$$R_2 = \{(\langle a, b, c \rangle, \langle a', b', c' \rangle) \mid a > 0 \wedge b > 0 \wedge a' = 2c \wedge b' = b - 1 \wedge c' = 1\}\ , \tag{1.16}$$

which can be over-approximated by T_1 and T_2 in (1.10) and (1.11).

It remains to check that (1.9) holds. If it does not, we can iterate the previous approximation technique, computing an over-approximation $\mathrm{wf}(\mathrm{wf}(R_{j_1}) \fatsemi R_{j_2})$ of the composition of R_{j_1} with R_{j_2}, then $\mathrm{wf}(\mathrm{wf}(\mathrm{wf}(R_{j_1}) \fatsemi R_{j_2}) \fatsemi R_{j_3})$ etc. until their union reaches a fixpoint or proves termination.

1.10.2 Relevance Logic

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

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

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

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

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

exchange
contraction
weakening

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

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

cut elimination

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

subformula property

Lemma 1.39 (Subformula Property). *If $\alpha \vdash A$ is a derivable sequent in $\mathbf{R}_{\supset}$, then there is a cut-free derivation of $\alpha \vdash A$ where every formula appearing in any sequent is a subformula of some formula of αA.*

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

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

A natural idea to pursue for deciding RI is to build a proof search tree with nodes labelled by sequents, and reversing rule applications from the root $\vdash A$ until only axioms are found as leaves. An issue with this idea is that the tree grows to an unbounded size, due in particular to contractions. See Exercise 1.26 for an algorithm that builds on this idea.

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

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

$$S \stackrel{\text{def}}{=} \mathcal{P}_f(\text{Seq}(A)) \tag{1.17}$$

be the set of finite subsets of Seq(A).

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

$$s' \to s' \cup \{\alpha \vdash B\} \tag{1.18}$$

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

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

multiset support

contraction ordering

Lemma 1.40 (Kripke's Lemma)**.** *The qo* $(\text{Seq}(A), \ll)$ *is a wqo.*

Then, by Proposition 1.23, the qo $(S, \leq^\flat)$, where $\leq^\flat$ is the *Hoare quasi-ordering* applied to $\ll$, is a wqo, and we easily see that $\mathcal{S} = \langle S, \to, \leq^\flat \rangle$ is a WSTS with effective pred-basis and a decidable ordering (see Exercise 1.25), thus the coverability problem for

$$s \stackrel{\text{def}}{=} \{B \vdash B \mid B \in \text{Sub}(A)\} \qquad\qquad t \stackrel{\text{def}}{=} \{\vdash A\} \tag{1.19}$$

is decidable by Proposition 1.38.

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

1.10.3 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. vector ac

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

$$\mathbf{x}_0 \xrightarrow{\mathbf{a}_1} \mathbf{x}_1 \xrightarrow{\mathbf{a}_2} \mathbf{x}_2 \xrightarrow{\mathbf{a}_3} \cdots \xrightarrow{\mathbf{a}_n} \mathbf{x}_n = \mathbf{x}\,. \tag{1.20}$$

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 $\mathbb{N}_\omega \overset{\text{def}}{=} \mathbb{N} \cup \{\omega\}$ of $\mathbb{N}$ and equip it with the obvious ordering. Tuples $\mathbf{y} \in \mathbb{N}_\omega^d$, called *$\omega$-markings*, are ordered with the product ordering. Note that $\mathbb{N}_\omega$ is a wqo, and thus $\mathbb{N}_\omega^d$ as well by Dickson's Lemma.

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

Definition 1.41 (Covering). Let $\mathcal{V}$ be a d-dimensional VAS. A set $C \subseteq \mathbb{N}_\omega^d$ of ω-markings is a *covering* for $\mathcal{V}$ if covering

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

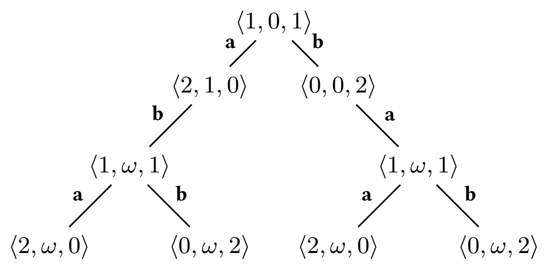


Figure 1.2: A Karp & Miller tree constructed for the VAS $\langle\{\mathbf{a},\mathbf{b},\mathbf{c}\},\langle 1,0,1\rangle\rangle$ with translations $\mathbf{a}=\langle 1,1,-1\rangle$, $\mathbf{b}=\langle -1,0,1\rangle$, and $\mathbf{c}=\langle 0,-1,0\rangle$.

---

iller tree

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

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

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

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

$$\mathbf{y}''(j)\overset{\text{def}}{=}\begin{cases}\omega & \text{if } \mathbf{y}'(j)>\mathbf{y}_i(j)\\ \mathbf{y}'(j) & \text{otherwise.}\end{cases} \tag{1.21}$$

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

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

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

Theorem 1.42. *The above algorithm terminates and the set of labels in the Karp & Miller tree is a covering for $\mathcal{V}$.*

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

We leave the second part of the proof as Exercise 1.28.

Karp-Miller trees are not only useful to compute coverings, and can be used to model-check VAS against LTL (linear time logic) formulae. This was extended to a larger class of WSTS in (Blondin et al., 2017).

1.10.4 Termination of rewrite systems

Recursive path orderings

Recursive path orderings (rpo) are a simple and powerful class of orderings that allow one to show termination of rewrite systems. Let Σ be a set of function symbols, and let $\alpha \colon \Sigma \to \mathbb{N}$ be an *arity* function. We restrict ourselves to the set $T_\alpha(\Sigma)$ terms obeying the arity. This is defined inductively by: for every $f \in \Sigma$, if $n = \alpha(f)$, then for all $t_1, \cdots, t_n \in T_\alpha(\Sigma)$, $f(t_1, \cdots, t_n)$ is in $T_\alpha(\Sigma)$.

arity

Given a precedence $\succ$ (a strict ordering) on Σ, one defines $\succ^{rpo}$ on terms (with or without arities) inductively by:

1. if $s_i \succeq^{rpo} t$ for some i, $1 \leq i \leq m$, then $s = f(\mathbf{s}) \succ^{rpo} t$;
2. if ($f \succ g$, or $f = g$ and $\mathbf{s}$ $(\succ^{rpo})_{mul}$ $\mathbf{t}$) and $s = f(\mathbf{s}) \succ^{rpo} t_j$ for every j, then $s \succ^{rpo} t = g(\mathbf{t})$.

Here $\succeq^{rpo} = \succ^{rpo} \cup =$. In $\mathbf{s}$ $(\succ^{rpo})_{mul}$ $\mathbf{t}$), $\mathbf{s}$ and $\mathbf{t}$ are seen as multisets of terms (i.e., we implicitly apply Parikh's mapping).

Dershowitz showed that $\succ^{rpo}$ is a well-founded qo for every well-founded precedence $\succ$, and that it is also monotonic (if $s_i \succ^{rpo} s'_i$, then $f(s_1, \cdots, s_i, \cdots, s_n) \succ^{rpo} f(s_1, \cdots, s'_i, \cdots, s_n)$) and stable (if $s \succ^{rpo} t$ then $s\sigma \succ^{rpo} t\sigma$ for every substitution σ) (Dershowitz, 1982). That was historically the first application of wqo theory to termination. It turns out that wqo theory is in fact not needed here, as was discovered later (Goubault-Larrecq, 2001), but one should note the following theorem (Dershowitz, 1979), which *does* require wqo theory.

simplification ordering

A *simplification ordering* is a monotone ordering $\sqsubseteq$ on terms in $T_\alpha(\Sigma)$ such ordering

that $f(t_1, \cdots, t_n) \sqsupset t_i$ for every i. By *monotone ordering* we mean that $t_i \sqsubset t'_i$ implies $f(t_1, \cdots, t_{i-1}, t_i, t_{i+1}, \cdots, t_n) \sqsubset f(t_1, \cdots, t_{i-1}, t'_i, t_{i+1}, \cdots, t_n)$. Note that the rpo is a simplification ordering.

Lemma 1.43. *Let $\sqsubseteq$ be a simplification ordering, and let $\leq_T$ be the homeomorphic embedding ordering on $T(\Sigma)$ with respect to the equality ordering on Σ. For all terms s, t in $T_\alpha(\Sigma)$, if $s \leq_T t$ then $s \sqsubseteq t$.*

Proof. By structural induction on a given proof of $s \leq_T t$ using rules $(T\text{-}add)$ and $(T\text{-}inc)$. In the former case, we use $t_i \sqsubseteq f(t_1, \cdots, t_n)$. In the latter case, we know that $f \leq g$ hence $f = g$ since the ordering on Σ is equality; and that $\mathbf{s}\ (\leq_T)_*\ \mathbf{t}$, allowing us to derive $f(\mathbf{s}) \leq_T f(\mathbf{t})$. Since the concerned terms are in $T_\alpha(\Sigma)$, they obey the arity function α, so that $\mathbf{s}$ and $\mathbf{t}$ have the same length $n = \alpha(f)$: $\mathbf{s} = (s_1, \cdots, s_n)$ and $\mathbf{t} = (t_1, \cdots, t_n)$. Looking at the definition of (word) embedding $(\leq_T)_*$, we must have $s_1 \leq_T t_1$, …, $s_n \leq_T t_n$, hence $s_1 \sqsubseteq t_1$, …, $s_n \sqsubseteq t_n$ by induction hypothesis. By monotonicity, $f(\mathbf{s}) \sqsubseteq f(\mathbf{t})$. $\square$

Theorem 1.44. *Every simplification ordering on $T_\alpha(\Sigma)$, where Σ is finite, is well-founded.*

Proof. Let $t_0 \sqsupset t_1 \sqsupset \cdots \sqsupset t_n \sqsupset \cdots$ be an infinite descending chain. The equality qo $\leq$ is well on Σ since Σ is finite. (Of course, $\leq$ is $=$, but writing $=_T$ instead of $\leq_T$ would be distressing.) By Kruskal's theorem $\leq_T$ is well, so there are indices $i < j$ such that $t_i \leq_T t_j$. By Lemma 1.43, $t_i \sqsubseteq t_j$, contradiction. $\square$

1.10.5 Van Der Meyden's algorithm, and Ogawa's improvement

Van der Meyden showed that, given a fixed disjunctive monadic query φ, there is a linear-time algorithm which takes an indefinite database D as input and answers whether φ is true of D (van der Meyden, 1997). However, just like for minor-closed properties of graphs, the algorithm exists but, until Ogawa (Ogawa, 2003), noone knew such an algorithm. The answer to that final riddle goes through the Curry-Howard correspondence, which you should learn in another MPRI course on type theory.

A (monadic) *atom* obeys the syntax:

atom

$$A ::= P(t) \mid s < t \mid s \leq t$$

where s, t are constants or variables, and P is a predicate symbol, taken from a finite set $Pred$. An atom is *ground* iff it contains no variable. An *indefinite database* D is just a finite collection of ground atoms. A *model* M maps each constant c to an element of $\mathbb{Z}$, and each predicate P to a relation on $\mathbb{Z}$. (The theory is unchanged if we take $\mathbb{Q}$ instead, or an unspecified linearly ordered, or dense linear order, or finite linear order.) We say that M is a model *of* D iff it makes every atom of D true, in notation $M \models D$.

ground

indefinite database

model

disjuncti

A *disjunctive monadic query* φ is any formula built from atoms using conjunction ($\wedge$), disjunction ($\vee$), and existential quantification ($\exists x$). One can assume that φ does not contain any constant, by replacing a constant c with an existential variable x constrained to satisfy $P_c(x)$, where P_c is a fresh unary predicate, and restricting ourselves to indefinite databases containing the atom $P_c(c)$ and no other atom with predicate P_c.

We say that φ satisfies D iff every model M of D makes φ true. Van der Meyden's algorithm solves the following question: for a fixed query φ, given D as input, does φ satisfy D?

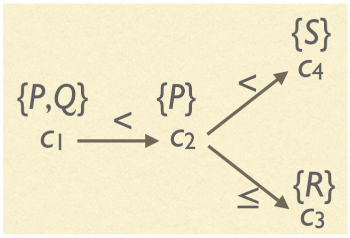
The following problem:
INPUT: a disjunctive monadic query φ, an indefinite database D;
QUESTION: does φ satisfy D?
is **coNP**-complete, even when φ is conjunctive, that is, written without $\vee$ (van der Meyden, 1997, Proposition 5.2, Theorem 4.6). What we shall do is examine the problem where φ is fixed, and only D is given as input.

normalized database

We may as well assume that D is a *normalized database*, by which we mean that there is no cycle of temporal constraints $c_1 \sqsubset c_2 \sqsubset \cdots \sqsubset c_n = c_1$, where each $\sqsubset$ is $\leq$ or $<$. Indeed, we can preprocess D by building a graph whose vertices are the constants in D, and having an edge $c \to d$ for every ground atom of the form $c < d$ or $c \leq d$. The edge is labeled '$<$' in the first case, '$\leq$' in the second case. We then run an algorithm that detects the strongly connected components of this graph in linear time. If some strongly connected component S contains an edge labeled '$<$', then the database is inconsistent, hence satisfies every query. Otherwise, we replace all the constants that are vertices of S by a single constant throughout D.

DAG

A normalized database D can then be represented as a *DAG*, that is, as a directed acyclic graph, where vertices are constants, edges are labeled '$<$' or '$\leq$', and each constant c is labeled by the set of predicates P such that $P(c)$ occurs in D. For example, the database consisting of the ground atoms $P(c_1)$, $Q(c_1)$, $P(c_2)$, $R(c_3)$, $S(c_4)$, $c_1 < c_2$, $c_2 < c_4$, $c_2 \leq c_3$ can be represented as the DAG:



Each path of such as DAG can be represented as a word over the finite alphabet $\Sigma = \mathbb{P}(Pred) \cup \{<, \leq\}$, where $<$ and $\leq$ are assumed to be distinct symbols, different from all predicate symbols in $Pred$. Call such a word a *flexiword*. For example, the above database is encoded as the set consisting of the two flexiwords $\{P, Q\} < \{P\} < \{R\}$ and $\{P, Q\} < \{P\} \leq \{R\}$.

flexiword

Order the letters of Σ by the ordering $\sqsubseteq$ defined by:

1. for all $A, B \subseteq Pred$, $A \sqsubseteq B$ iff $A \subseteq B$;

2. $\leq \sqsubseteq <$ ($\leq$ is below $<$), $< \sqsubseteq <$ and $\leq \sqsubseteq \leq$.

If you think as sets of predicates as conjunctions, you should recognize reverse logical implication. This allows us to order flexiwords by $\sqsubseteq_*$, and then databases by $(\sqsubseteq_*)^\flat$.

Van der Meyden shows that if $D \models \varphi$ and $D(\sqsubseteq_*)^\flat D'$, then $D' \models \varphi$. The proof is longish but not difficult, and is an induction on φ. It follows that for a fixed query φ, the set A_φ of all indefinite databases D satisfying φ is upwards-closed in $(\sqsubseteq_*)^\flat$. By Higman's Lemma (Theorem 1.19) and its consequence Proposition 1.23, there is a finite set A_0 of indefinite databases such that, for every indefinite database D, φ satisfies D iff $D_0(\sqsubseteq_*)^\flat D$ for some $D_0 \in A_0$.

Checking $\sqsubseteq_*$ can be done, by dynamic programming (or using memoization) in polynomial time. For D_0 fixed, this turns out to be linear time in the size of D. The algorithm that, on input D, checks whether some element of A_0 is $\sqsubseteq_*$ D, then runs in linear time as well.

However, we do not *know* the finite set A_0, and that prevents us from implementing the algorithm. (The situation is similar to the question of deciding minor-closed properties of graphs, see Section 1.7.)

Ogawa solved that last problem (Ogawa, 2003). He observed that Murthy and Russell had produced an intuitionistic proof of Higman's Lemma (Murthy and Russell, 1990). Such a proof, by the Curry-Howard correspondence, is a *program*. Using that program, one can compute the finitely many elements of A_0.

There are other intuitionistic proofs of Higman' Lemma in the literature (Richman and Stolzenberg, 1990; Coquand and Fridlender; Berghofer, 2004; Geser, 1996), and some will produce faster algorithms than others.

1.10.6 A short word on parameterized complexity

Ogawa's version of van der Meyden's algorithm gives an algorithm that takes both φ and D in argument, and decides whether φ satisfies D in time $O(f(k)n)$, where k is the size of φ, n is the size of D, for some computable function f. The value $f(k)$ may be extremely high, but for k fixed, this is linear in n.

This shows that satisfiability of disjunctive monadic queries by indefinite databases is in the parameterized complexity class **FPT** (Downey and Fellows, 1999), the class of languages of pairs of inputs of sizes k, n that can be decided in time $O(f(k)p(n))$ for some total function f and some polynomial n. The fundamental aspect of this class is the multiplicative dependence on k and n: f does not depend on n, and p does not depend on k. Another aspect is that f is not required to be computable.

The point is to recognize that the parameter of size k (e.g., the query φ) will usually be small, and that we would like an algorithm of low complexity on the size n of the other parameter (the database), which can be big.

Finding **FPT** algorithms is by no means obvious. There are a few standard techniques to achieve that, e.g., kernelization. Wqo-based techniques, when applicable, yield parameterized algorithms of the form: "test whether the input is larger than or equal to some element of A_0", for some finite set A_0. (If one can compute the finite set A_0, then f is computable.) This applies, notably, to various algorithms on graphs, thanks to the Robertson-Seymour Theorem. Historically the first application was Fellows and Langston's **FPT** algorithm for the **NP**-complete problem VERTEX COVER (Fellows and Langston, 1987). The book by Downey and Fellows is a comprehensive introduction to parameterized complexity (Downey and Fellows, 1999).

1.10.7 Dickson's 1913 theorem on odd perfect numbers

Dickson proved and used Dickson's lemma to show that for every $k \in \mathbb{N}$, there are only finitely many odd perfect numbers with at most k prime factors (Dickson, 1913b). (We still do not know whether there is any odd perfect number *at all.*) Here is a sketch of his proof.

For every $N \in \mathbb{N}$, let $\sigma(N)$ denote the sum of the divisors of N, $\sum_{n|N} n$. Here $|$ is the *divisibility qo*, and it is not wqo, because the infinitely many primes form an antichain. However, for every finite set P of primes, the set of P-*smooth* numbers, which are numbers whose only prime factors lie in P, *is* wqo under divisibility. The reason is that every number can be written in a unique form as a product of powers of primes $\prod_i p_i^{n_i}$, and that describes an isomorphism of ordered monoids between $(\mathbb{N}, \times, |)$ et $(\mathbb{N}[\mathbb{N}], +, \leq)$, where $\mathbb{N}[\mathbb{N}]$ is the set of infinite sequences of natural numbers with only finitely many non-zero entries. This isomorphism cuts down to an isomorphism between P-smooth numbers and $\mathbb{N}^d$ where d is the cardinality of P. Dickson's Lemma then implies that divisibility is wqo on P-smooth numbers for finite P.

divisibility qo

P-smooth

A number N is *perfect* iff $\sigma(N) = 2N$, *deficient* iff $\sigma(N) < 2N$, *abundant* iff $\sigma(N) > 2N$. Dickson shows (Lemma D, op. cit.) that there is a number M larger than all the prime factors of non deficient odd numbers with at most k prime factors. Let $P = \{p_1, \cdots, p_d\}$ enumerate the primes smaller than M. The odd perfect numbers we are looking for are P-smooth. Then Dickson notices that every proper divisor of a non-abundant number is deficient, that every proper multiple of a non-deficient number is abundant. Hence the P-smooth numbers form an upwards-closed subset for $|$, and the perfect numbers are minimal in that set. Proposition 1.7 tells us there can be only finitely many of them.

perfect

deficient

abundant

Nowadays, we know explicit upper bounds, and those improved results do not use Dickson's Lemma any longer. That is life.

1.10.8 Polynomial ideals, and Hilbert's Basis Theorem

The main reference here is Jean-Charles Faugère's course. One can recommend (Buchberger and Loos, 1982-1983) for a high-level view, or (Adams and Loustaunau, 1994).

Let $\mathbb{K}$ be a commutative field (in general, a ring with various amenable properties is enough). Let $\overline{X} = X_1, \cdots, X_n$ be finitely many variables, and consider the ring $\mathbb{K}[\overline{X}]$ of polynomials over those variables. A *unitary monomial* $X_1^{k_1} \cdots X_n^{k_n}$ is nothing else than an element of $\mathbb{N}^n$, and a polynomial is merely a formal linear combination of unitary monomials, with coefficients in $\mathbb{K}$, that is, a map from $\mathbb{N}^n$ to $\mathbb{K}$ whose values at elements of $\mathbb{N}^n$ are 0 except for finitely many unitary monomials.

unitary monomial

A *monomial* is the product aM of a non-zero element a of $\mathbb{K}$ with a unitary monomial M, so that a polynomial is also a sum of monomials.

monomial

It is practical to consider a *monomial ordering*. This is a total ordering $\preceq$ on unitary monomials that is compatible with products of unitary monomials (i.e., with addition in $\mathbb{N}^n$—equivalently, products of unitary monomials is monotonic with respect to $\preceq$), and which extends the componentwise ordering $\leq$. For example, the lexicographic ordering on $\mathbb{N}^n$ is fine. We shall fix one in the sequel.

monomial ordering

Given that, every non-zero polynomial P has a unique *leading monomial* $\text{lm}(P)$, which is the unique $\preceq$-largest unitary monomial that appears in P with a non-zero coefficient. Then one can write any non-zero polynomial P as $aL + R$, where $a \in \mathbb{K}$, $L = \text{lm}(P)$, and every unitary monomial M in R is $\prec L$.

leading monomial

Example 1.45. With the lexicographic ordering as monomial ordering, $P = 3X_1^3X_2X_3^2 - 2X_1^3X_2X_3 + 9X_1^2X_2^4X_3^5 - 5X_1^2X_2^3X_3^9 + 7X_1^2X_2X_3 - 6X_1X_2^2X_3^2 + 4X_1X_2X_3 + 2X_2^2X_3 - 11X_3^7$ is a polynomial in $\mathbb{Q}[X_1, X_2, X_3]$ such that $\text{lm}(P) = X_1^3X_2X_3^2$. With our definition of polynomials as formal linear combinations of elements of $\mathbb{N}^n$, P is $3 \times (3,1,2) - 2 \times (3,1,1) + 9 \times (2,4,5) - 5 \times (2,3,9) + 7 \times (2,1,1) - 6 \times (1,2,2) + 4 \times (1,2,3) + 2 \times (0,2,1) - 11 \times (0,0,7)$. The leading monomial is then $(3,1,2)$. By the way, we have actually written P sorted by decreasing order of unitary monomials, with respect to the monomial ordering.

It is useful to see P as a *rewrite rule* $L \to_P -1/aR$. Applying the rule to a polynomial Q means finding a unitary monomial in Q through which the leading monomial L of P factors, namely a monomial of the form bLM, and replacing it by $-b/aRM$. In other words, Q is written $bLM + Rem$ with $b \in \mathbb{K}$ non-zero and M a unitary monomial, and then Q rewrites to $-b/aRM + Rem$. In general, for a set G of polynomials, we write $Q \to_G Q'$ iff $Q \to_P Q'$ for some $P \in G$.

rewrite rule

Example 1.46. The polynomial P of Example 1.45 can be interpreted as the rewrite rule $X_1^3X_2X_3^2 \to_P \frac{2}{3}X_1^3X_2X_3 - 3X_1^2X_2^4X_3^5 + \frac{5}{3}X_1^2X_2^3X_3^9 - \frac{7}{3}X_1^2X_2X_3 - 2X_1X_2^2X_3^2 - \frac{4}{3}X_1X_2X_3 - \frac{2}{3}X_2^2X_3 + \frac{11}{3}X_3^7$. The idea is that given any point with coordinates (X_1, X_2, X_3) such that $P(X_1, X_2, X_3) = 0$, we can simplify expressions $Q(X_1, X_2, X_3)$ involving X_1, X_2 and X_3 by this rewrite rule. For example, with $Q = X_1^5X_2^2X_3^4 - 7X_1^3X_2^2X_3^2$, we can first rewrite $\mathrm{lm}(Q) = X_1^5X_2^2X_3^4 = X_1^2X_2X_3^2 \times \mathrm{lm}(P)$ to:

$$\begin{aligned} X_1^2X_2X_3^2 \times (&\frac{2}{3}X_1^3X_2X_3 - 3X_1^2X_2^4X_3^5 + \frac{5}{3}X_1^2X_2^3X_3^9 - \frac{7}{3}X_1^2X_2X_3 \\ &- 2X_1X_2^2X_3^2 - \frac{4}{3}X_1X_2X_3 - \frac{2}{3}X_2^2X_3 + \frac{11}{3}X_3^7) \\ = \frac{2}{3}&X_1^5X_2^2X_3^3 - 3X_1^4X_2^5X_3^7 + \frac{5}{3}X_1^4X_2^4X_3^{11} - \frac{7}{3}X_1^4X_2^2X_3^3 \\ &- 2X_1^3X_2^3X_3^4 - \frac{4}{3}X_1^3X_2^2X_3^3 - \frac{2}{3}X_1^2X_2^3X_3^3 + \frac{11}{3}X_1^2X_2X_3^9 \end{aligned}$$

by $\to_P$. We have the following sequence of rewrites:

$$\begin{aligned} Q \to_P\ & \frac{2}{3}X_1^5X_2^2X_3^3 - 3X_1^4X_2^5X_3^7 + \frac{5}{3}X_1^4X_2^4X_3^{11} - \frac{7}{3}X_1^4X_2^2X_3^3 \\ & - 2X_1^3X_2^3X_3^4 - \frac{4}{3}X_1^3X_2^2X_3^3 - \frac{2}{3}X_1^2X_2^3X_3^3 + \frac{11}{3}X_1^2X_2X_3^9 \\ & - 7X_1^3X_2^2X_3^2 \\ = & \frac{2}{3}X_1^5X_2^2X_3^3 - 3X_1^4X_2^5X_3^7 + \frac{5}{3}X_1^4X_2^4X_3^{11} - \frac{7}{3}X_1^4X_2^2X_3^3 \\ & - 2X_1^3X_2^3X_3^4 - \frac{4}{3}X_1^3X_2^2X_3^3 - 7X_1^3X_2^2X_3^2 - \frac{2}{3}X_1^2X_2^3X_3^3 + \frac{11}{3}X_1^2X_2X_3^9 \\ & \qquad\qquad \text{(sorting the monomials)} \\ \to_P\ & \frac{4}{9}X_1^5X_2^2X_3^2 - 3X_1^4X_2^5X_3^7 - 2X_1^4X_2^5X_3^6 + \frac{5}{3}X_1^4X_2^4X_3^{11} \\ & + \frac{10}{9}X_1^4X_2^4X_3^{10} - \frac{7}{3}X_1^4X_2^2X_3^3 - \frac{14}{9}X_1^4X_2^2X_3^2 - 2X_1^3X_2^3X_3^4 \\ & - \frac{4}{3}X_1^3X_2^3X_3^3 - \frac{4}{3}X_1^3X_2^2X_3^3 - \frac{71}{9}X_1^3X_2^2X_3^2 - \frac{2}{3}X_1^2X_2^3X_3^3 \\ & - \frac{4}{9}X_1^2X_2^3X_3^2 + \frac{11}{3}X_1^2X_2^1X_3^9 + \frac{22}{9}X_1^2X_2^1X_3^8 \\ \to_P\ & \cdots \end{aligned}$$

If we always rewrite the largest monomial, we obtain a normal form of Q in 77 more steps. This is a polynomial $\frac{8}{27}X_1^5X_2^2X_3^1 - 0.790123\cdots X_1^4X_2^5X_3^1 + 0.0867076\cdots X_1^4X_2^4X_3^1 - \frac{28}{9}X_1^4X_2^2X_3^1 + \cdots$ (331 more monomials). Note that non of the unitary monomials is divisible by (larger than or equal to, in the pointwise ordering on $\mathbb{N}^3$) $\mathrm{lm}(P) = X_1^3X_2^1X_3^2$.

Measure a polynomial Q by letting $\mu(Q)$ be the set of all the unitary monomials that arise in Q with a non-zero coefficient. Considering every finite set as a multiset, $Q \to_G Q'$ implies that $\mu(Q)$ is larger than $\mu(Q')$ in the multiset extension of $\prec$. Now $\prec$ has no infinite descending chain by Lemma 1.13 (3) and Dickson's Lemma. A theorem due to Dershowitz and Manna (Dershowitz and

Manna, 1979) states that the multiset extension $\prec_{mul}$ of a strict terminating relation $\prec$ is again terminating[2] So $\to_G$ terminates.

Gröbner basis

G is called a *Gröbner basis* iff $\to_G$ is confluent. In that case, every polynomial P has a unique $\to_G$-normal form $P \downarrow G$. It is clear that $P - P \downarrow G$ lies in the ideal $I = (G)$ generated by G. In particular $P \downarrow G = 0$ iff $P \in (I)$ (this is the point!). The if direction is obvious, and the converse direction is a consequence of our argument.

If G is a Gröbner basis for I $(= (G))$, then every $Q \in I$ can be written as a linear combination $\sum_{i=1}^m Q_i P_i$ where each P_i is in G, each Q_i is a polynomial, and the leading monomials of those $Q_i P_i$ that are non-zero are pairwise distinct[3] Note also that $\text{lm}(P_i)\text{lm}(Q_i) = \text{lm}(P_i Q_i)$, because the monomial ordering $\preceq$ is compatible with products of unitary monomials.

Write $\text{lm}(G)$ for $\{\text{lm}(P) \mid P \in G, P \neq 0\}$. It will be useful to note that $Q \to_P Q'$ for some Q' if and only if some unitary monomial of Q is $\geq \text{lm}(P)$, equating monomials with elements of $\mathbb{N}^n$ with its pointwise ordering. Hence $Q \to_P Q'$ for some Q' if and only if some unitary monomial of Q is in $\uparrow \text{lm}(G)$; and Q is $\to_G$-normal if and only if it is a linear combination of monomials in the downwards-closed complement of $\uparrow \text{lm}(G)$.

Lemma 1.47. *G is a Gröbner basis for $I = (G)$ iff $\text{lm}(I) = \uparrow \text{lm}(G)$ (where unitary monomials are equated with elements of $\mathbb{N}^n$).*

Proof. Assume that G is a Gröbner basis. For every $M \in \text{lm}(I)$, by definition of $\text{lm}(I)$ there is a $P \in I$ with $\text{lm}(P) = M$. Write P as $\sum_{i=1} Q_i P_i$ as above. Since the leading monomials of those $Q_i P_i$ that are non-zero are pairwise distinct, M is one of them, say $M = \text{lm}(P_i)\text{lm}(Q_i)$. That implies $M \geq \text{lm}(P_i)$ in the usual ordering on $\mathbb{N}^n$.

Conversely, assume $\text{lm}(I) = \uparrow \text{lm}(G)$. Let $Q \in I$. One can write Q as $\sum_{j=1}^m Q_i P_i$ where each P_i is in G, since $I = (G)$. We show by induction on

[2]Sketch. Let Acc be the set of multisets from which no infinite $\succ$-chain can start, and show that every multiset M is in Acc by induction on the number of elements of M. This requires you to prove that $M \in Acc$ implies M union x is in Acc, for every element x. Imagine that fails, and take (x, M) $(\prec, \prec_{mul})_{lex}$-minimal such that M union x starts an infinite $\succ$-chain. Look at the first step and reach a contradiction.

[3]There is a $\to_G$ rewrite sequence $Q = Q^0 \to_{P_{i_0}} Q^1 \to_{P_{i_1}} \to \cdots \to Q^{n-1} \to_{P_{i_{n-1}}} \to Q^n = 0$ from Q to 0, where each P_{i_k} is in G. For each k, $0 \leq k \leq n-1$, $Q^k - Q^{k+1}$ is of the form $c_k M_k P_{i_k}$ for some monomial $c_k M_k$, and its leading monomial is $\text{lm}(Q^k - Q^{k+1}) = \text{lm}(Q^k) = L_k M_k$ where $L_k = \text{lm}(P_{i_k})$. Then $Q = \sum_{k=0}^{n-1}(Q^k - Q^{k+1}) = \sum_{i=1}^m Q_i P_i$ where, for each i, $Q_i = \sum_{k/i_k=i} c_k M_k$. We also have $\text{lm}(Q^0) \succ \text{lm}(Q^1) \succ \cdots \succ \text{lm}(Q^{n-1})$ because rewriting decreases the leading monomial strictly (in the monomial ordering). That is, $L_0 M_0 \succ L_1 M_1 \succ \cdots \succ L_{n-1} M_{n-1}$. In particular, the subsequence of those M_k such that i_k is equal to some fixed i is strictly decreasing in $\prec$. (For every such k, $L_k = \text{lm}(P_i)$, so L_k is the same for every k in the subsequence. If M_k were not $\succ$ some later $M_{k'}$, then $M_k \preceq M_{k'}$ since $\preceq$ is total, so $L_k M_k \preceq L_{k'} M_{k'}$, contradiction.) It follows that either there is no k such that $i_k = i$, in which case $Q_i = 0$, or $\text{lm}(Q_i) = M_k$ where k is the smallest index such that $i_k = i$. In the latter case, $\text{lm}(P_i)\text{lm}(Q_i) = L_k M_k$. Since $L_0 M_0 \succ L_1 M_1 \succ \cdots \succ L_{n-1} M_{n-1}$, the leading monomials of $P_i Q_i$, $\text{lm}(P_i)\text{lm}(Q_i)$, are pairwise distinct.

$\mu(Q)$ that $Q \to_G^* 0$. If $Q = 0$, that is clear. Otherwise, by assumption there is a $P \in G$ such that $\mathrm{lm}(Q) \geq \mathrm{lm}(P)$. This implies that $Q \to_P Q'$ for some polynomial Q'. Since $\mu(Q) >_{mul} \mu(Q')$, the induction hypothesis gives us $Q' \to_G^* 0$, whence the conclusion.

Now, if $Q \to_G^* Q_1, Q_2$, we need to show that Q_1 and Q_2 have a common $\to_G$-normal form. Since $\to_G$ is terminating, Q_1 has a normal form Q_1' and Q_2 has a normal form Q_2'. By construction $Q_1' - Q_2'$ is in I, so $Q_1' - Q_2' \to_G^* 0$. However, every unitary monomial that occurs in $Q_1' - Q_2'$ occurs in Q_1' or in Q_2', and since each Q_i' is normal, there is no such monomial in $\uparrow \mathrm{lm}(G)$. It follows that $Q_1' - Q_2'$ is in normal form already. Hence $Q_1' - Q_2' = 0$, so $Q_1' = Q_2'$ is the desired common normal form. □

Theorem 1.48 (Hilbert's Basis Theorem). *Every polynomial ideal has a Gröbner basis, and is in particular of the form (G) for G finite.*

Proof. Let I be an ideal. $\mathrm{lm}(I)$ is upwards-closed in $\mathbb{N}^n$, hence is of the form $\uparrow\{M_1, \cdots, M_k\}$ for a finite antichain of monomials M_j by Dickson's Lemma. Each M_i is in $\mathrm{lm}(I)$, hence of the form $\mathrm{lm}(P_i)$ for some non-zero $P_i \in I$. Let $G = \{P_1, \cdots, P_k\}$. Clearly $(G) \subseteq I$.

In the converse direction, let $Q \in I$, and let Q' be a $\to_G$-normal form of Q. We recall that $Q - Q'$ can be written as $\sum_{i=1}^k Q_i P_i$ for some polynomials P_i, so $Q - Q'$ is in (G), hence in I. Since Q is also in I, Q' is in I. Since Q' is normal, it rewrites by no $\to_{P_i}$, so no unitary monomial of Q' is in $\uparrow \mathrm{lm}(G)$. By construction, $\uparrow \mathrm{lm}(G) = \mathrm{lm}(I)$, so no unitary monomial of Q' is in $\mathrm{lm}(I)$. That is impossible if $Q' \neq 0$, in which case, since $Q' \in I$, $\mathrm{lm}(Q') \in \mathrm{lm}(I)$. Hence $Q' = 0$. Since $Q - Q' \in (G)$, Q is in (G).

We have obtained obtain that $I = (G)$, and by construction $\mathrm{lm}(I) = \uparrow \mathrm{lm}(G)$. Lemma 1.47 then tells us that G is a Gröbner basis for I. □

Noetherian

A ring is *Noetherian* if and only if every infinite monotone sequence $I_0 \subseteq I_1 \subseteq \cdots \subseteq I_n \subseteq \cdots$ of ideals is stationary. Theorem 1.48 implies $\mathbb{K}[\overline{X}]$ is a Noetherian ring. In fact, for a ring, Noetherianness is equivalent to every ideal being finitely generated—the proof is similar to Proposition 1.7. That also works for every Noetherian ring $\mathbb{K}$, not just fields.

1.10.9 Buchberger's algorithm

Given a finite set of polynomials A, one can compute a Gröbner basis G for the ideal (A), by a process similar to Knuth-Bendix completion, called *Buchberger's algorithm* (Buchberger, 1965).

Buchberger's algorithm

We use the following notations. For $L_1, L_2 \in \mathbb{N}^n$, let $L_1 \vee L_2$ be the upper bound of L_1 and L_2. This is obtained by taking a componentwise max. Let $L_1 \wedge L_2$ be its lower bound, computed as a componentwise min. In monomial parlance, $L_1 \vee L_2$ is the *lcm* (least common multiple) of the unitary monomials L_1 and

lcm

L_2, and $L_1 \wedge L_2$ is the *gcd* (greatest common divisor) of L_1 and L_2. We shall interpret those as operations on unitary monomials, whose product is sum in $\mathbb{N}^n$. For monomials, note that $L_1 L_2 = (L_1 \vee L_2)(L_1 \wedge L_2)$. gcd

Consider two non-zero polynomials $P_1 = a_1 L_1 + R_1$ and $P_2 = a_2 L_2 + R_2$ where $L_1 = \mathrm{lm}(P_1)$, $L_2 = \mathrm{lm}(P_2)$. Let $L'_1 = L_2/(L_1 \wedge L_2)$, $L'_2 = L_1/(L_1 \wedge L_2)$, so that $L_1 L'_1 = L_2 L'_2 = L_1 \vee L_2$. The *S-polynomial* of P_1 and P_2 is defined as $S(P_1, P_2) = 1/a_1 L'_1 P_1 - 1/a_2 L'_2 P_2 = L'_1 R_1/a_1 - L'_2 R_2/a_2$. This is the simplest linear combination of P_1 and P_2 that cancels their leading monomials. S-polynomial

By definition, any ideal containing P_1 and P_2 must contain $S(P_1, P_2)$. The basic form of Buchberger's algorithm is:

```
BUCHBERGER ($A$ : finite set of polynomials) =
    **while** there is an S-polynomial $P := S(P_1, P_2)$ with $P_1, P_2 \in A$
            such that $P \downarrow_A \neq 0$ **do**
            $A := A \cup \{P \downarrow_A\}$;
    **return** $A$;
```

where $P \downarrow_A$ denotes an arbitrary normal form of P with respect to $\to_A$. That algorithm can be improved (a lot!), see J.-C. Faugère's course.

This is only an algorithm provided: (1) the arithmetic operations on elements of $\mathbb{K}$ are computable, and (2) it terminates. We deal with (2) right away. (1) is dealt with either by using oracle Turing machines, or by reasoning on $\mathbb{K} = \mathbb{Q}$ for example.

Proposition 1.49. *BUCHBERGER terminates.*

Proof. Let $A_0, A_1, \dots, A_n, \dots$ be the sequence of values of A obtained through the loop, and imagine it is infinite. Write $A_{n+1} = A_n \cup \{Q_n\}$, where $Q_n \neq 0$ is in $\to_{A_n}$-normal form. In particular, $\mathrm{lm}(Q_n) \notin \uparrow \mathrm{lm}(A_n)$. However, by Dickson's Lemma we can find $i < j$ such that $\mathrm{lm}(Q_i) \leq \mathrm{lm}(Q_j)$. This is impossible since $\mathrm{lm}(Q_j) \notin \uparrow \mathrm{lm}(A_j)$ and $Q_i \in A_j$. □

Soundness is a consequence of the following.

Proposition 1.50 (Buchberger's Criterion). *G is a Gröbner basis if and only if, for any two polynomials $P_1, P_2 \in G$, $S(P_1, P_2)_{\downarrow G} = 0$.*

We will admit the if direction. The only if direcction is easier to prove. Assume G is a Gröbner basis. By Lemma 1.47 $\mathrm{lm}(G) = \mathrm{lm}((G))$. Clearly $S(P_1, P_2)_{\downarrow G}$ in in (G). If it is non-zero, then its leading monomial is in $\mathrm{lm}((G)) = \mathrm{lm}(G)$, hence it is not $\to_G$-normal: contradiction.

Corollary 1.51. *For each finite set of polynomials A, G =BUCHBERGER(A) is a Gröbner basis of the ideal (A).*

Proof. (Sketch.) On termination, Buchberger's algorithm produces a set G that satisfies Buchberger's criterion. □

Buchberger's algorithm looks a lot like Knuth-Bendix completion. However an additional difficulty is that we do not add $S(P_1, P_2)$, rather a normal form of it (with respect to the current set of polynomials). The fact that this is correct was generalized by Marché (Marché, 1996) under the heading of *normalized* completion.

normalized

1.11 Beyond Wqos

1.11.1 Noetherian Spaces

Noetherian

Generalizing Corollary 1.9, call a topological space X *Noetherian* iff every infinite monotone sequence $U_0 \subseteq U_1 \subseteq \cdots \subseteq U_n \subseteq \cdots$ of *open* subsets is stationary. One can generalize most of what we have done from wqos to Noetherian spaces, and the following is a convenient dictionary between the two worlds:

Wqo	Noetherian
Upwards-closed	Open
Downwards-closed	Closed
Monotonic map	Continuous map
Compatibility for $\rightarrow$	$\rightarrow$ is lower semicontinuous

See (Goubault-Larrecq, 2013, Section 9.7) for the mathematics of Noetherian spaces, and (Goubault-Larrecq, 2010) for a few applications in verification.

A strange property is that, in contrast to Theorem 1.28, if X is Noetherian, then $\mathbb{P}(X)$ is Noetherian, too, for some topology. Noetherianness is preserved by all the data type constructions mentioned in Section 1.2, and a few more.

Zariski topology

Historically, the first Noetherian topology was the *Zariski topology* on $\mathbb{C}^k$. This is the topology whose closed subsets are the sets of the form $\mathcal{Z}(I)$ for some ideal I of $\mathbb{C}[X_1, \cdots, X_k]$, where $\mathcal{Z}(I)$ is the set of common zeroes of all polynomials in I.

$\mathbb{C}$ does not lend itself to computations on a Turing machine: one cannot even represent all complex numbers, which are uncountable. In practice, one often considers the coarser topology whose closed subsets are $\mathcal{Z}(I)$, where I is an ideal on $\mathbb{Q}[X_1, \cdots, X_k]$. Call that the $\mathbb{Q}$-Zariski topology.

The following properties show how one may compute on closed sets (and by complementation, on open subsets), coded as $\mathcal{Z}(I)$. I itself is represented as a Gröbner basis G, as $I = (G)$. Then:

1. If $I \subseteq I'$ then $\mathcal{Z}(I) \supseteq \mathcal{Z}(I')$, and if $G \subseteq G'$ then $(G) \subseteq (G')$;
2. (arbitrary intersections) $\bigcap_k \mathcal{Z}(I_k) = \mathcal{Z}(\sum_k I_k)$, and $\sum_k (G_k) = (\bigcup_k G_k)$;
3. (finite unions) $\bigcup_{k=1}^n \mathcal{Z}(I_k) = \mathcal{Z}(\bigcap_{k=1}^n I_k)$.
4. $\mathcal{Z}(\{0\}) = \mathbb{C}^k$, $\mathcal{Z}(\mathbb{Q}[X_1, \cdots, X_k]) = \emptyset$;

5. $\mathcal{Z}(I) = \mathbb{C}^k$ iff $I = \mathbb{Q}[X_1, \cdots, X_k]$, and $(G) = \mathbb{Q}[X_1, \cdots, X_k]$ iff G contains some constant non-zero polynomial.

Many other operations are computable on Gröbner bases. A striking application was given by Müller-Olm and Seidl (Müller-Olm and Seidl, 2002) to the verification of so-called *polynomial programs*, namely programs in a simple imperative language with **if** conditionals, **while** loops, sequencing, and assignments of the form $x := e$, where e is an expression built using the numerical operators $+$, $-$, $*$ (i.e., a polynomial), or the special wildcard '?', meaning any value whatsoever. The same paper gives a sizable collection of useful computable operations on Gröbner bases.

polynomial programs

The paper (Goubault-Larrecq, 2010) briefly explains how Müller-Olm and Seidl's approach can also be seen as an analogue of the standard backwards algorithm for WSTS, applied to a topological variant of WSTS whose state space is $\mathbb{C}^k$ with the $\mathbb{Q}$-Zariski topology. It also shows how topology allows one to go much beyond, for example to verify networks of polynomial programs that communicate through lossy channels of discrete messages.

1.11.2 Better quasi-orderings

Similarly to Noetherian spaces, better quasi-orderings (bqo) also solve the conundrum posed by Rado's structure, in the sense that if $\leq$ is bqo on X then $\leq^\flat$ is bqo on $\mathbb{P}(X)$.

The notion is due to Nash-Williams (Nash-Williams, 1968), and is more complex than that of wqo. One should note that all total well-founded orders are bqo, that all bqos are wqo, but the converse fails. There are analogues of Higman's lemma and Kruskal's theorem, due to Nash-Williams. In fact, even *infinite* sequences and infinite trees are bqo under natural generalizations of the embedding quasi-orderings, assuming a better quasi-ordered set of letters, resp. of function symbols.

In particular, all the data types mentioned in Section 1.2 are bqo, to the possible exception of $\mathcal{G}$, for which the result is unknown.

There seems to be only one application of bqos in computer science: in (Abdulla and Nylén, 2000), the authors develop a theory of existential constraints, which applies to the verification of timed networks and timed Petri nets, and rests on bqo theory to show that the collection of constraints is bqo, hence wqo under the implication quasi-ordering. That paper also gives a readable introduction to the basics of bqo theory. For a more comprehensive account, see (Marcone, 1994). We just recap the definition(s) and a few properties.

The simplest possible "definition" is as follows. An ω^2-wqo is a qo such that $\leq^\flat$ is wqo on $\mathbb{P}(X)$. This implies wqo since $\leq^\flat$ well-founded means $\leq$ wqo. Conversely, Rado's structure is wqo but not ω^2-wqo. In fact, a wqo is ω^2-wqo iff it does not contain Rado's structure, see (Laver, 1976, Theorem 1.7) and (Jančar, 1999). One can then define an ω^3-wqo as a qo X such that $\mathbb{P}^2(X) = \mathbb{P}(\mathbb{P}(X))$ is wqo, and so on. We can also iterate this in the transfinite, and require that the colimit of all $\mathbb{P}^\alpha(X)$, α countable ordinal, is wqo: this is a bqo. This idea is formalized in (Forster, 2003).

The above intuitive idea was Nash-Williams' original idea. However, with that definition we cannot even prove that a finite qoset is bqo under equality, hence one usually prefers a different definition. One of the simplest definitions is due to Péquignot (Péquignot, 2015, Definition 2.49), and is as follows. We first generalize infinite sequences to *multi-sequences*, which are families indexed not by $n \in \mathbb{N}$, but by infinite subsets of $\mathbb{N}$.

Given an infinite subset A of $\mathbb{N}$, we write $[A]^\omega$ for the set of infinite sequences of elements of A. We usually think of elements **s** of $[A]^\omega$ as increasing sequences: for example, $\{2, 3, 5, 7, 11, \cdots\}$ is viewed as the increasing sequence $2 < 3 < 5 < 7 < 11 < \cdots$. The *shift map* sends **s** to the sequence $_*\mathbf{s}$ obtained by removing the first element from **s**. For example, $_*\{2, 3, 5, 7, 11, \cdots\} = \{3, 5, 7, 11, \cdots\}$.

shift map

multi-sequence

locally constant

A *multi-sequence* in a qoset D is any family $(x_\mathbf{s})_{\mathbf{s}\in[A]^\omega}$ of elements of D, where A is an infinite subset of $\mathbb{N}$. It is *locally constant* if and only if every $x_\mathbf{s}$ only depends on a finite prefix of **s**: formally, for every $\mathbf{s} \in [A]^\omega$, there is a finite prefix $\mathbf{s}_0$ of **s** such that, for every $\mathbf{t} \in [A]^\omega$ such that $\mathbf{s}_0$ is a prefix of **t**, $x_\mathbf{t} = x_\mathbf{s}$ (let us call $\mathbf{s}_0$ a *witness of local constancy* for **s**). A multi-sequence $(x_\mathbf{s})_{\mathbf{s}\in[A]^\omega}$ is *good* if and only if $x_\mathbf{s} \leq x_{_*\mathbf{s}}$ for some $\mathbf{s} \in [A]^\omega$. Finally, D is *bqo* if and only if every locally constant multi-sequence in D is good.

witness of local constancy

good

bqo

One can think of $[A]^\omega$ as an infinite, countably-branching tree; its elements are the branches, and we share branches through their common prefixes, so its vertices are finite increasing sequences of natural numbers. Given a locally constant multi-sequence $(x_\mathbf{s})_{\mathbf{s}\in[A]^\omega}$, if we go down the tree, starting from the root, along any branch **s** until we find the first witness $\mathbf{s}_0$ of local constancy for **s**. The collection of such witnesses forms a so-called *block*, namely a set B of elements of $[A]^{<\omega}$ (the family of finite subsets of A) such that every $\mathbf{s} \in [A]^\omega$ has a unique finite prefix in B, and such that $\bigcup B$ is infinite.

block

This leads to the standard definition of bqos. We equate each $\mathbf{s} \in [\mathbb{N}]^{<\omega}$ with the sorted list $n_1 < n_2 < \cdots < n_k$ of its elements. We write $\mathbf{s} \lhd \mathbf{t}$ iff there is an infinite subset **u** of $\mathbb{N}$ such that **s** is a finite prefix of **u** and **t** is a finite prefix of $_*\mathbf{u}$. (Beware that $\lhd$ is not transitive.) An *array* is any family $(x_\mathbf{s})_{\mathbf{s}\in B}$, where B is a block. It is *good* iff one can find $\mathbf{s} \lhd \mathbf{t}$ in B such that $x_\mathbf{s} \leq x_\mathbf{t}$. A *bqo* is a qo on X such that every array is good.

array

good

bqo

One has the following results:

- every total well-founded qoset is bqo, every bqo is ω^2-wqo, every ω^2-wqo is wqo, every wqo is well-founded;
- in particular, $\mathbb{N}$ with its usual ordering is bqo;
- every finite qo is bqo;

- every finite disjoint sum, every finite product of bqos is bqo;
- if D is bqo, then so are D^*, $D^{\circledast}$, $\mathcal{P}_f(D)$, $\mathcal{H}_{\text{fin}}(D)$;
- if D is bqo, then so is $T(D)$;
- (not available on wqos) if D is bqo, then the set of *infinite*, countable words over D is bqo under embedding;
- (not available on wqos) if D is bqo, then $\mathbb{P}(D)$ is bqo under $\leq^\flat$, $\mathcal{H}(D)$ and *Idl*(D) are bqo under inclusion;
- (not available on wqos) if D is bqo, then the set of *infinite*, countable trees is bqo under homeomorphic embedding.

It is unknown whether the set of finite undirected graphs is bqo under the minor relation.

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 non-negative reals $(\mathbb{R}_+, \leq)$;

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

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

prefix ordering

lexicographic ordering

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

subsets

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

Smyth's ordering

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

Exercise 1.2 (Lexicographic Sum and Product). Let $(A_1, \leq_1)$ and $(A_2, \leq_2)$ be two qos. The *lexicographic sum* $A_1 + A_2$ is the qo $(A_1 + A_2, \leq_{\leq_{\text{lex}}})$ with same support set $\{1\} \times A_1 \cup \{2\} \times A_2$ as for the disjoint sum but with an ordering defined with

lexicographic sum

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

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

lexicographic product

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

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

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

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

Exercise 1.3. Is every wqo necessarily countable? Is every well partial order necessarily countable? Give a proof or a counter-example.

Exercise 1.4 (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 1.5 (Characterization of wqos, V)**.** The goal of this exercise is to give yet another characterization of wqos: a partial order is wqo if and only if all its linearizations are well-founded, a result from (Wolk, 1967). We also include some related material.

We will use Zorn's Lemma. This states that in an inductive partial ordered set, every element is less than or equal to some maximal element. An *inductive* poset is one where every non-empty totally ordered subset has an upper bound.

inductive

(1) We consider the set $Ord(X)$ of all partial orderings on X, and we order it by inclusion: $(\leq_1) \subseteq (\leq_2)$ iff for all $x, y \in X$ such that $x \leq_1 y$, then $x \leq_2 y$. Given a non-empty totally ordered family of partial orders $\leq_i$, $i \in I$ (not necessarily countable) on X, show that their union $\leq$ (namely, $x \leq y$ iff $x \leq_i y$ for some $i \in I$) is again a partial order. Hence $Ord(X)$ is inductive.

(2) If $\leq$ is a partial order on X such that x, y are incomparable (you can in fact merely assume that $y \not\leq x$), exhibit a larger partial order $\leq'$ in $Ord(X)$ (namely, $(\leq) \subseteq (\leq')$) such that $x \leq' y$.

Szpilrajn's Theorem

(3) Deduce *Szpilrajn's Theorem*: every partial ordering has a total extension.

(4) Let $\leq$ be a qo on a set X, and assume a bad sequence x_n, $n \in \mathbb{N}$. Define another qo $\leq'$ by $x \leq' y$ iff $x \leq y$, or $x \leq x_j$ and $x_i \leq y$ for some $i < j$. Verify that $\leq'$ is indeed a qo, extends $\leq$, and that $x_0 \geq' x_1 \geq' \cdots \geq' x_n \geq' \cdots$. Check also that $\leq'$ is a partial order if $\leq$ is one, and that in that case x_n, $n \in \mathbb{N}$, is an infinite decreasing chain in $\leq'$.

(5) Let $\leq$ be a partial order on a set X. Show that $\leq$ is wqo iff *all* its linear extensions are well-founded.

Exercise 1.6. Let Σ be a finite set $\{a_1, a_2, \cdots, a_n\}$, quasi-ordered by equality $=$. Letting $\leq$ be that order, we have defined the qo $\leq_{\circledast}$ on $\Sigma^{\circledast}$.

There is a bijection between the set $\Sigma^{\circledast}$ of multisets over Σ and $\mathbb{N}^n$, which maps every multiset μ to the vector whose kth component is the number of occurrences of a_k in μ. What can you say of $\leq_{\circledast}$ vs. the componentwise ordering on $\mathbb{N}^n$? Deduce a relation between Corollary 1.21 and (Dickson's Lemma, 1913).

sparser-than ordering

Exercise 1.7 ($\mathbb{Z}^k$, $\leq_{\text{sparse}}$)**.** We consider the *sparser-than ordering*. Given $\mathbf{a} = (a_1, \ldots, a_k)$ and $\mathbf{b} = (b_1, \ldots, b_k)$ two tuples in $\mathbb{Z}^k$, we define

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

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

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

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

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

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

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

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

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

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

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

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

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

(7) Show that $(R^\omega, \leq_{R,\omega})$, the ω-sequence extension of Rado's structure $(R, \leq_R)$ —see Definition 1.27—, 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 1.9 (Higman's Lemma for Matrices?)**.** A quasi-ordering $(A, \leq)$ leads to a natural notion of embedding on 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 (Mat$[A], \leq_{\text{Mat}}$) wqo?

Exercise 1.10 (Ordering Powersets)**.** Recall from Exercise 1.1 the definition of Smyth's ordering on the powerset $\mathcal{P}(A)$: if $(A, \leq)$ is a qo and $U, V \subseteq A$ we let: Hoare ordering

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

There also exists the *Hoare ordering*:

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

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

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

(3) Express $\sqsubseteq_{\mathrm{S}}$ in terms of $\sqsubseteq_{\mathrm{H}}$ (and conversely), 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, $(\mathcal{P}(A), \sqsubseteq_{\mathrm{H}})$ is well-founded. (wqo.7)

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

(6) Show that $(\mathcal{P}_f(R), \sqsubseteq_{\mathrm{S}})$ and $(\mathcal{P}(R), \sqsubseteq_{\mathrm{H}})$ are *not* wqos, where R is Rado's structure —see Definition 1.27.

Exercise 1.11. Let $\leq$ be wqo on X. We see the elements of X as letters, so that X^* is the set of finite words over X. A *regular expression* R is given by the following grammar:

$$\begin{array}{rcll} R, S, T, \ldots & ::= & x & (x \in X) \\ & | & \varepsilon & \text{empty word} \\ & | & RS & \text{concatenation} \\ & | & 0 & \text{empty set} \\ & | & R + S & \text{union} \\ & | & R^* & \text{Kleene star} \end{array}$$

Each regular expression R defines a language $[\![R]\!]$, that is, a subset of X^*, in the obvious way. In particular $[\![0]\!]$ is the empty set, $[\![\varepsilon]\!]$ is the one-element set containing just the empty word, again written ε, and $[\![RS]\!] = \{uv \mid u \in [\![R]\!], v \in [\![S]\!]\}$. By definition, a *regular language* on X is a language of the form $[\![R]\!]$ for some regular expression R.

(1) We define a variant of (a subcase of) regular expressions as follows.

The *literals* are of the form $x^?$ or $A^{\downarrow *}$ where A is a finite subset of X. They denote languages defined by: $[\![x^?]\!] = \{\varepsilon\} \cup \{y \in X \mid y \leq x\}$ (we equate letters y with words of length 1); $[\![A^{\downarrow *}]\!]$ is the set of words all of whose letters are in $\downarrow A$.

The *products* P are concatenations $L_1 L_2 \cdots L_n$ of literals. $[\![P]\!]$ is defined in the obvious way. Notice that $[\![P]\!] = \{\varepsilon\}$ if $n = 0$, and accordingly we shall write ε for the empty product. It should be clear that every product P is downwards-closed with respect to the embedding qo $\leq_*$ (formally, $[\![P]\!]$ is downwards-closed).

Define the *support* of a literal and of a product as the following set of letters: the support of $x^?$ is the set $\{x\}$, the support of $A^{\downarrow *}$ is A, the support of a product is the union of the supports of its literals. For finitely many products $P_1, \ldots, P_n$, show that the language that one may denote by $(P_1 + \cdots + P_n)^*$, of concatenations of arbitrarily many words in $\bigcup_{i=1}^{n} [\![P_i]\!]$, coincides with $[\![S^{\downarrow *}]\!]$, where S is the union of the supports of $P_1, \ldots, P_n$.

(2) Show that, for every regular expression R, there are finitely many products P_i such that $[\![R]\!] = \bigcup_i [\![P]\!]_i$. You may also observe that the construction is effective, i.e., given by an algorithm.

(3) Define the qo $\sqsubseteq$ on literals, then on products, inductively by the following rules:

1. (Literals) $x^? \sqsubseteq y^?$ if $x \leq y$ in X;
2. $x^? \sqsubseteq A^{\downarrow *}$ if $x \leq y$ for some $y \in A$;
3. $A^{\downarrow *} \sqsubseteq B^{\downarrow *}$ if $A \leq^\flat B$, in other words if for every $x \in A$, there is a $y \in B$ such that $x \leq y$;
4. (Products) $\varepsilon \sqsubseteq P$ for every product P;
5. $LP \sqsubseteq L'P'$ if $L \not\sqsubseteq L'$ and $LP \sqsubseteq P'$;
6. $x^?P \sqsubseteq y^?P'$ if $x \leq y$ and $P \sqsubseteq P'$;
7. $LP \sqsubseteq B^{\downarrow *}P'$ if $L \sqsubseteq B^{\downarrow *}$ and $P \sqsubseteq B^{\downarrow *}P'$.

To avoid a lengthy verification, we will admit that $\sqsubseteq$ characterizes exactly inclusion of (languages of) products: $P \sqsubseteq P'$ iff $[\![P]\!] \subseteq [\![P']\!]$. (In particular, inclusion of products is decidable.)

Let Y be the disjoint sum of X and $\mathcal{P}_f(X)$, quasi-ordered in a suitable way, and f be a suitable surjective monotone map from Y^* to the set of products quasi-ordered by $\sqsubseteq$. Show that products are wqo under $\sqsubseteq$, i.e., under language inclusion.

(4) Show that $[\![P]\!]$ is directed for every product P.

(5) A *simple regular expression* (*SRE*) is a finite sum of products. We will also admit that inclusion of (languages of) SREs is characterized by $P_1 + \ldots + P_m \sqsubseteq P'_1 + \cdots + P'_n$ iff for every i, there is a j such that $P_i \sqsubseteq P'_j$. Think of using Lemma 1.31!

simple regular expression

(6) Show that inclusion of SREs is again wqo.

(7) Conclude that $\leq^\flat$ is wqo on the set of regular languages on X, extending Proposition 1.23, which was the similar result on the smaller set of finite subsets of X. The result is due to Intrigila and Varricchio (Intrigila and Varricchio, 2000, Proposition 2.1). Their proof is completely different, and proceeds by a direct minimal bad sequence argument: read it and compare!

Exercise 1.12. Let Σ be a finite alphabet, and let A be *any* downwards-closed subset of Σ^* with respect to $\leq_*$, where $\leq$ is equality on Σ. Show that A is regular.

Hint: given any word w, show that $\uparrow w$ is regular. From that, deduce that any upwards-closed subset of Σ^* is regular, and conclude.

Exercise 1.13. The following is due to Vityniotis, Coquand, and Wahlstedt (Vytiniotis et al., 2012). The *full* relation Full on a set X is defined by x Full y for all $x, y \in X$. For a binary relation R on X, and a point $z \in X$, the relation $z \triangleright R$ is defined by $x\ (z \triangleright R)\ y$ iff $x\ R\ y$ or $z\ R\ x$.

Define the predicate "*almost full*" by the following rules, inductively:

almost full

1. Full is almost full;

2. For every binary relation R on X such that for every $z \in X$, $z \triangleright R$ is almost full, R itself is almost full.

well

Say that a binary relation R is *well* iff it has no bad sequence, i.e., for every sequence x_n, $n \in \mathbb{N}$, one can find $i < j$ such that $x_i \; R \; x_j$. Hence a wqo is a well qo, but we are not requiring transitivity here.

Show that a binary relation R is almost full iff it is well.

Exercise 1.14. Let $\to$ be a terminating relation on X. Show that $\not\to$ is well. Does the converse implication hold, i.e., is it true that if $\not\to$ is almost full then $\to$ terminates? What about if $\to$ is transitive?

Well Structured Transition Systems

Exercise 1.15. A *multiset vector addition system with states* (MVASS) is a transition system whose states are multisets M of vectors in $\mathbb{N}^d$ (d fixed but arbitrary), and whose only two rules are:

$$\begin{array}{rcll} M \uplus \{|x|\} & \to & M \uplus \{|x + a_i|\} & \text{if } x + a_i \geq 0 \\ M \uplus \{|x, y|\} & \to & M \uplus \{|x + y|\} & \end{array}$$

where $a_1, \ldots, a_n$ are fixed vectors in $\mathbb{Z}^d$. Show that coverability is decidable for MVASS. Of course we order the set $(\mathbb{N}^d)^{\circledast}$ of states by $\leq_{\circledast}$, where $\leq$ is the componentwise ordering on $\mathbb{N}^d$.

MVASS are closely related to the notion of BVASS found in the literature, and which has uses in computational linguistics, in cryptographic protocol verification, in linear logic, and in verification of XPath queries.

Exercise 1.16. Consider a deterministic Turing machine $\mathcal{M}$, and let X be its set of configurations. Let $\to$ be the (one-step) transition relation of $\mathcal{M}$: $x \to y$ if running $\mathcal{M}$ for exactly one step from configuration x, we arrive at configuration y. For each $x \in X$, let $\ell(x)$ be the length of the unique (complete) computation $x \to x_1 \to \cdots \to x_n \to \cdots$ of $\mathcal{M}$ starting with configuration x; $\ell(x) = +\infty$ if $\mathcal{M}$ does not terminate starting from $\mathcal{M}$. Let $x \leq y$ iff $\ell(x) \leq \ell(y)$ in $\mathbb{N}_\omega = \mathbb{N} \cup \{+\infty\}$.

(1) Show that $(\mathcal{M}, \to, \leq)$ is a WSTS.

(2) Given a universal Turing machine $\mathcal{M}$, why is the coverability problem of $(\mathcal{M}, \to, \leq)$ undecidable?

(3) Show that every universal Turing machine $\mathcal{M}$ can be modified so as to yield another universal Turing machine $\mathcal{M}'$ such that $(\mathcal{M}', \to, \leq)$ has an effective pred-basis.

(4) In the lectures, we have seen that every WSTS satisfying certain conditions has a decidable coverability problem. List those conditions, and say which ones fail in WSTS of the form $(\mathcal{M}', \to, \leq)$, where $\mathcal{M}'$ is as in the previous question.

Exercise 1.17 (Transitive Compatibility)**.** We relax the (compatibility) condition to a weaker notion of compatibility, but show that Term remains decidable in this setting. Consider the following replacement for (compatibility):

transitive compatibility

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

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

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

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

reflexive transitive compatibility

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

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

(1) Show that, if U is upward-closed, then $\mathit{Pre}^*_\exists(U)$ is also upward-closed. Does Lemma 1.37 still hold?

(2) Let K_0 be a finite basis of U. Lift *pb*—recall the *effective pred-basis* function from page 18—to operate on finite sets. The sequence

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

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

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

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

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

strict compatibility

$$s \rightarrow s' \wedge s < t \text{ implies } \exists t',\, s' < t',\, t \rightarrow t' \;. \tag{sc}$$

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 $\mathit{Post}^*(s_0)$ is infinite.

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

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

[Fin] Finiteness

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

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

Exercise 1.20 (Downward WSTSs)**.** Let $\langle S, \to\rangle$ be a transition system and $(S, \leq)$ be a wqo. The definition of compatibility is also known as "upward-compatibility", by contrast with its dual *reflexive downward compatibility* :

reflexive downward compatibility

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

downward WSTS

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

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

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

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

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

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

Program Termination

Exercise 1.22. Let $Y_1, \ldots, Y_n$ be finitely many wqos. We write their qos $\leq_1, \ldots, \leq_n$.

(1) Let X be a set with a binary relation $\to$, and assume there are ranking functions $\rho_k \colon X \to Y_k$, $1 \leq k \leq n$, such that for all $x, x' \in X$, if $x \to^+ x'$ then for some k, $\rho_k(x) \not\leq_k \rho_k(x')$. Show that $\to$ is terminating. (This is the basis of the so-called *size-change principle* in automated termination checking of computer programs.)

(2) Show that the conclusion of the last question does not hold if we only assume that for all $x, x' \in X$, if $x \to x'$ then for some k, $\rho_k(x) \not\leq_k \rho_k(x')$.

Exercise 1.23 (Disjunctive Termination Arguments)**.** Assume that a binary relation R verifies (1.9) on page 20—not necessarily a qo—, 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 1.39.

Exercise 1.25 (A WSTS for Relevant Implication)**.** Prove that $\mathcal{S}$ defined by equations (1.17) and (1.18) is a WSTS with effective pred-basis and decidable ordering.

⋆ **Exercise 1.26** (Proof Search for Relevant Implication)**.** The purpose of this exercise is to find an alternative algorithm for RI. The key idea in this algorithm is to remove (Con) from $\mathbf{R}_{\supset}$ and apply contractions only when needed, i.e. modify the rules ($\supset_\mathsf{L}$) and ($\supset_\mathsf{R}$) to contract their conclusion, but only inasmuch as could not be obtained by first contracting their premises. Doing so we define an alternative proof system $\mathbf{R}'_{\supset}$ that includes the unmodified (Ax) and ($\supset_\mathsf{R}$), and a modified version of ($\supset_\mathsf{L}$):

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

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

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

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

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

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

Karp & Miller Trees

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

Exercise 1.28 (Covering)**.** The aim of this exercise is to complete the proof of Theorem 1.42 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 1.41. ⋆⋆

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

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

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

(2) In order to prove that C satisfies Definition 1.41.1, we will prove a stronger statement. For an ω-marking $\mathbf{y}$ in $\mathbb{N}^d_\omega$, first define

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

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

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

Karp & Miller graph

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

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

ω-node

(3) Let us prove that C satisfies Definition 1.41.2. The idea is that we can find reachable configurations of $\mathcal{V}$ that agree with $\mathbf{y}$ on its finite components, and that can be made arbitrarily high on its ω-components. For this, we focus on the graph nodes where new ω values are introduced by step 2, which we call *ω-nodes.*

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

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

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

Other applications.

learning from positive data

Exercise 1.29. A *set system* is just a set X together with a collection of subsets of X, which we call the *properties*, and such that every finite subset of X is included in some property. The setting of *learning from positive data*, given a set system, is as follows. There are two players, Teacher and Learner, who play in turn. Let $P_0 = \emptyset$. At turn $n \geq 1$, Teacher presents fresh data x_n—by fresh we mean $x_n \notin P_{n-1}$—and Learner submits a property P_n that is meant to explain all previous data—formally, x_1, x_2, …, x_n are all in P_n. Teacher is assumed to follow a given, but unknown strategy, which is a partial function that maps histories $x_1, P_1, x_2, P_2, \cdots, x_{n-1}, P_{n-1}$ to the next data $x_n \notin P_n$. We say that Teacher is *stumped* when he cannot produce x_n, i.e., when its strategy is not defined on the current length $2n$ history. We say that Learner has a *winning strategy* iff whatever Teacher's strategy, Teacher is eventually stumped. If that is the case, then we say that the set system is *learnable from positive data.* (Positive data only requires that the example data $x_1, \ldots, x_n, \ldots$, should all be in the final propery P_n when Teacher is stumped. Negative data would also require Learner to produce properties that *do not* contain certain other data $y_1, \ldots, y_m, \ldots$)

Assume that X is equipped with a qo $\leq$. Show that, if $\leq$ is a wqo, then every set system on X whose properties are all upwards-closed is learnable from positive data.

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 (1952b), although Dickson's Lemma already appeared (in a rather different form) in (Dickson, 1913a). The proofs of Higman's Lemma (Higman, 1952a) and of Kruskal (Kruskal, 1960) using a "minimal bad sequence" argument is due to Nash-Williams (1963). The reader will find more information in the survey of Milner (1985), which also covers *better quasi orders* (bqo). See Lovász (2006) for an exposition of Robertson-Seymour, its underlying ideas, and its consequences in graph theory.

better quasi order

Well Structured Transition Systems have been developed in different directions by Finkel (1987, 1990) and Abdulla et al. (1996), before a unifying theory finally emerged in the works of Abdulla et al. (2000) and Finkel and Schnoebelen (2001)—the latter being our main source for this chapter and exercises 1.17 to 1.20. More recent developments are concerned with the algorithmics of downward-closed sets (Finkel and Goubault-Larrecq, 2009, 2012) and of games (Abdulla et al., 2008; Bertrand and Schnoebelen, 2013).

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

Relevance Logic. The reader will find a good general exposition on relevance logic in the chapter of Dunn and Restall (2002), and in particular a discussion of decidability issues in their Section 4, from which Exercise 1.26 is taken (credited to Kripke, 1959). Both the approach in the exercise and that of the main text scale to larger fragments like the conjunctive implicative fragment $\mathbf{R}_{\supset,\wedge}$, but Urquhart (1984) proved the undecidability of the full relevance logic **R** and its variants the *entailment logic* **E** and *ticket logic* **T**. Urquhart (1999) proved $\mathbf{R}_{\supset,\wedge}$ to be Ackermannian (see CRI on page 142), while RI was shown to be 2ExpTime-complete only very recently (Schmitz, 2016b). The decidability of implicative ticket logic $\mathbf{T}_{\supset}$ was recently proven by Padovani (2013), and its complexity is unknown.

Karp & Miller Trees and vector addition systems were first defined by Karp and Miller (1969). Coverability trees are used in a large number of algorithms and decision procedures on VAS, although their worst-case size can be Ackermannian in the size of the input VAS (Cardoza et al., 1976). Quite a few of these problems, including termination and coverability, can actually be solved in ExpSpace instead (Rackoff, 1978a; Blockelet and Schmitz, 2011), but finite equivalences are an exception (Mayr and Meyer, 1981; Jančar, 2001); see FCP on page 140. 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.

More WSTS. We have already seen Petri nets (with n *places*), which are transition systems on $S = \mathbb{N}^n$, defined by finitely many rules of the form $x \to x + b$, where b is from a finite set of *transitions* in $\mathbb{Z}^n$, which apply iff $y + c \geq 0$. Petri nets are WSTS by Dickson's

Lemma, and even strongly monotonic. They are effective. One can enrich the model: Petri nets with reset arcs, with transfer arcs, etc. All are special cases of the *affine* systems considered in (Finkel et al., 2004), which are transition systems on $S = \mathbb{N}^n$ defined by finitely many rules $x \to Ax + b$, where A is an $n \times n$-matrix with coefficients in $\mathbb{N}$ and $b \in \mathbb{Z}^n$, where again such a rule is fireable iff $Ax + b \geq 0$. They are all WSTS.

The complexity of Petri net coverability is EXPSPACE-complete (Rackoff, 1978b). This is proved by a technique specific to Petri nets, not by the standard backward algorithm, which is not optimal in that case. I am not aware of known bounds for the more general affine systems.

Lossy channel systems are finite networks of finite automata that communicate through FIFO (first-in, first-out) channels that may spontaneously lose any letter they contain. Without the latter, we would have channel systems, whose reachability is undecidable. In lossy channel systems, reachability is equivalent to coverability. The state space is $\prod_{i=1}^{n} Q_i \times \prod_{j=1}^{m} \Sigma_j^*$, where Q_i is the finite state space of the ith automaton, Σ_j is the finite alphabet of the jth channel (all ordered by $=$). This is wqo by Higman's Lemma and the fact that finite products of wqos are wqo. See Abdulla and Jonsson (1993) for details.

The verification of concurrent programs with so-called weak memory consistency—which is how actual, modern machines, do work!—is equivalent to the verification of lossy channel systems, a discovery due to Atig et al. (2010), see also Abdulla et al. (2015); Bouajjani et al. (2015).

Data nets generalize Petri nets and much more (Lazic et al., 2007). Those are effective WSTS with state space $(\mathbb{N}^n)^*$.

Timed Petri nets (Abdulla et al., 2004b) are Petri nets with integer durations, and can be quotiented without loss of information. The quotient is done by coalescing so-called *regions*, in the sense of timed automata, to single points. The quotient system is an effective WSTS, whose state space is $(P \times I)^{\circledast} \times ((P \times I)^{\circledast})^* \times P^{\circledast}$, where P is the finite set of places of the underlying Petri net, I is a finite interval $\{0, 1, \cdots, T_{\max}\}$ of durations (both ordered by $=$).

Vector addition systems with states (VASS) are like Petri nets except the state space is $Q \times \mathbb{N}^n$ for some finite set Q of states, and transitions can change state (Hopcroft and Pansiot, 1979). They are again effective WSTS. They are recursively equivalent to Petri nets. Coverability is again EXPSPACE-complete for them.

Branching VASS (BVASS) are VASS enriched with new binary rules of the form $(q_1, x), (q_2, y)$ - $(q_3, x + y)$, where $q_1, q_2, q_3 \in Q$. Those rules apply to all $x, y \in \mathbb{N}^n$. BVASS (Verma and Goubault-Larrecq, 2005), also known as VATA (de Groote et al., 2004) are not WSTS, because they are not even transition systems. However, they can be encoded as transition systems over $(Q \times \mathbb{N}^n)^{\circledast}$, called *multiset VASS* (MVASS), by rules of the form $m \uplus \{\!|(q_1, x)|\!\} \to m \uplus \{\!|(q_2, x + b)|\!\}$ (Petri net rule; b constant in $\mathbb{Z}^n$; rule fireable if $x + b \geq 0$) and $m \uplus \{\!|(q_1, x), (q_2, y)|\!\} \to m \uplus \{\!|(q_3, x + y)|\!\}$. MVASS are effective WSTS. The two cited papers give applications, one to verification of cryptographic protocols, the other one to the (yet unsolved) decidability of the multiplicative exponential fragment of linear logic. Those systems had been invented earlier in computational linguistics under the name of multiset-valued linear indexed grammars (Rambow, 1994), see (Schmitz, 2010). Since their inception into computer science (Verma and Goubault-Larrecq, 2005; de Groote et al., 2004), a variety of applications were found, sometimes requiring yet newer extensions, to database theory and the verification question for XPath queries (Jacquemard et al., 2016), to observational equivalence for fragments of the ML language (Cotton-Barratt et al., 2017), or to the decidability of implicational relevance logic (Schmitz, 2016a). The latter uses the result that BVASS coverability is 2-EXPTIME-complete (Demri et al., 2012).

Finally, one should also mention various decision problems for variously expressive process algebras (Hüchting et al., 2013; Acciai and Boreale, 2010; Wies et al., 2010). In the latter reference, process algebras are abstracted as trees, and this is the only known instance of WSTS requiring Kruskal's Theorem to date. It also uses a forward procedure.

2

COMPLEXITY UPPER BOUNDS

As seen in Chapter 1, many algorithms rely on well quasi orderings for their proof of termination. Although it is true that the classical proofs of Dickson's Lemma, Higman's Lemma (Higman, 1952a), 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 $(\mathrm{a}, \mathrm{b})$
$\mathrm{c} \longleftarrow 1$
while $\mathrm{a} > 0 \land \mathrm{b} > 0$
    $l : \langle \mathrm{a}, \mathrm{b}, \mathrm{c} \rangle \longleftarrow \langle \mathrm{a} - 1, \mathrm{b}, 2\mathrm{c} \rangle$
  or
    $r : \langle \mathrm{a}, \mathrm{b}, \mathrm{c} \rangle \longleftarrow \langle 2\mathrm{c}, \mathrm{b} - 1, 1 \rangle$
end
```

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

Recall the SIMPLE program from Figure 1.1 on page 19, repeated here in Figure 2.1. We argued on page 19 that, in any run, the sequence of values taken by $\mathtt{a}$ and $\mathtt{b}$

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

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

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

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

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

of length

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

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

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

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

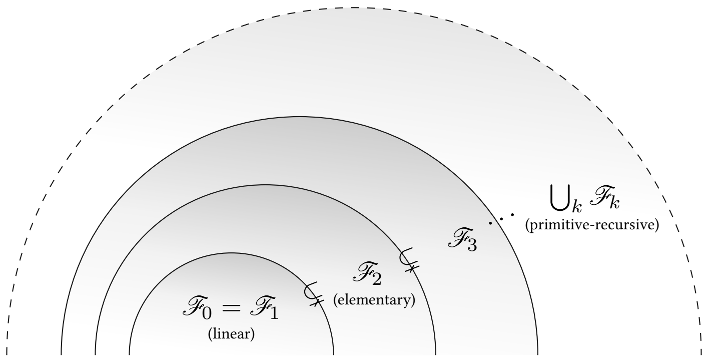


Figure 2.2: The Grzegorczyk hierarchy of primitive-recursive functions.

2.1 The Length of Controlled Bad Sequences

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

2.1.1 Controlled Sequences

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

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

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

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

1. the first is a *norm* $|.|_A \colon A \to \mathbb{N}$ on the elements to represent their size. We always assume $A_{\leq n} \overset{\text{def}}{=} \{a \in A \mid |a|_A \leq n\}$ 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* $|\langle m,n\rangle|_{\mathbb{N}^2} \overset{\text{def}}{=} |\langle m,n\rangle|_\infty = \max(m,n)$; normed wqo

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

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 $n \in \mathbb{N}$ $\overset{\text{def}}{\Leftrightarrow}$

controlled sequence

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

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

Definition 2.1 (Basic nwqos). We note $[k]$ the nwqo $(\{0, \ldots, k-1\}, \leq, |.|_{[k]})$ defined over the initial segment of the natural numbers, where $|j|_{[k]} \overset{\text{def}}{=} j$ for all $0 \leq j < k$, and Γ_k the generic k-elements nwqo $(\{a_0, \ldots, a_{k-1}\}, =, |.|_{\Gamma_k})$ where $|a_j|_{\Gamma_k} \overset{\text{def}}{=} 0$ 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 Kőnig's Lemma, this tree of bad sequences is therefore finite, of some height $L_{g,n,A}$ representing the length of the maximal (g,n)-controlled bad sequence(s) over A. In the following, since we are mostly interested in this length as a function of the initial norm, we will see this as a length $L_{g,A}(n)$ depending on n (and parameterized by g and A), or as a *length function* $L_{g,A} : \mathbb{N} \to \mathbb{N}$. Our purpose will then be to obtain complexity bounds on $L_{g,A}$.

length function

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

2.1.2 POLYNOMIAL NWQOS

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

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

nwqo isomorphism

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

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

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

empty nwqo

singleton nwqo

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

disjoint sum

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

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

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

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

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

cartesian product

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

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

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

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

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

naturals nwqo

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

polynomial nwqo

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

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

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

Γ_1 is neutral for $\times$:

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

$+$ is associative and commutative:

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

$\times$ is associative and commutative:

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

and $\times$ distributes over $+$:

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

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

polynomial normal form

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

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

2.1.3 Subrecursive Functions

We already witnessed with SIMPLE that the complexity of some programs implementable as monotone counter systems can be quite high—more than a tower of exponentials $2^{2^{\cdot^{\cdot^{2}}}}\}^{b \text{ times}}$ for $\textsc{simple}(2, b)$ in Equation (2.2) on page 54, which is a non-elementary function of b. However there is a vast space of functions that are non-elementary but recursive—and even primitive recursive, which will be enough for our considerations.

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

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

The lower levels correspond to reasonable classes, $\mathscr{F}_0 = \mathscr{F}_1$ being the class of linear functions, and $\mathscr{F}_2$ that of elementary functions. Starting at level 1, the hierarchy is strict in that $\mathscr{F}_k \subsetneq \mathscr{F}_{k+1}$ for $k > 0$ (see Figure 2.2 on page 55).

fast-growing function

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

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

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

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

Observe that

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

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

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

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

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

honest function

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

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

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

We delay more formal details on $(\mathscr{F}_k)_k$ until Section 2.4 on page 68 and Exercise 2.3 on page 76 and turn instead to the main theorem of the chapter.

2.1.4 Upper Bounds for Dickson's Lemma

Length Function Theorem

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

The Length Function Theorem is especially tailored to give upper bounds for VASS configurations (recall Example 2.5 on page 57), but can also be used for VASS extensions. For instance, the runs of SIMPLE can be described by bad sequences in $\mathbb{N}^2$, of form described by Equation (2.1) on page 54. As these sequences are controlled by the linear function $g(x) = 2x$ in $\mathscr{F}_1$, the Length Function Theorem with $p = \gamma = 1$ entails the existence of a bounding function in $\mathscr{F}_3$ on the length of any run of SIMPLE, which matches the non-elementary length of the example run we provided in (2.2).

2.2 Applications

Besides providing complexity upper bounds for various problems, the results presented in this chapter also yield new "combinatorial" algorithms: we can now employ an algorithm that looks for a witness of *bounded size*. We apply this technique in this section to the two WSTS algorithms presented in Section 1.9.

Exercise 2.4 investigates the application of the Length Function Theorem to the program termination proofs of Section 1.10.1, and Exercise 2.14 to the Karp & Miller trees of Section 1.10.3. 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 1.34), lossy counter machines (Section 3.1), reset machines (Section 3.5), or other examples of well-structured counter machines with transitions controlled by $g(x) = x + b$ for some b—which is a function in $\mathscr{F}_1$—, with d counters, and with p states, the Length Function Theorem yields an upper bound in $\mathscr{F}_{d+1}$ on the length of controlled bad sequences. This is improved to $\mathscr{F}_d$ by Corollary 2.36 on page 74. When b or p is part of the input, this rises to $\mathscr{F}_{d+1}$, and when d is part of the input, to F_ω, which asymptotically dominates every primitive-recursive function.

2.2.1 Termination Algorithm

Let us consider the Termination problem of Section 1.9.1. Let $\mathcal{S} = \langle S, \rightarrow, \leq \rangle$ be a WSTS over a normed wqo $(S, \leq, |.|)$ where the norm $|.|$ is also the size for a concrete representation of elements in S, let s_0 be an initial state in S with $n = |s_0| + 1$, and let $g(|s|)$ be an upper bound on the space required to compute some s' from s verifying $s \rightarrow s'$. We can reasonably expect g to be increasing and honest, and use it as a control over sequences of states: we compute an upper bound

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

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

Because any run of $\mathcal{S}$ of length $\ell \overset{\text{def}}{=} f(n) + 1$ is necessarily good, we can replace the algorithm in the proof of Proposition 1.36 by an algorithm that looks for a finite witness of non-termination of form

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

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

2.2.2 Coverability Algorithm

Recall that the algorithm of Section 1.9.2 for WSTS coverability of t from s, relied on the saturation of a sequence (1.3) on page 18 of subsets of S. In order to derive an upper complexity bound on this problem, we look instead at how long we might have to wait until this sequence proves coverability, i.e. consider the length of

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

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

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

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

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

There is therefore a sequence

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

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

2.3 Bounding the Length Function

This section and the next together provide a proof for the Length Function Theorem. The first part of this proof investigates the properties of bad controlled sequences and derives by induction over polynomial nwqos a *bounding function* $M_{g,A}(n)$ on the length of (g,n)-controlled bad sequences over A (see Proposition 2.20 on page 68). The second part, detailed in Section 2.4, studies the properties of the $M_{g,A}$ functions, culminating with their classification in the Grzegorczyk hierarchy.

2.3.1 Residual nwqos and a Descent Equation

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

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

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

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

of length $n + 1$.

Formally, (2.30) only proves one direction of (2.29), which is that $L_{g,\mathbb{N}}(n) \geq n$; an argument for the converse inequality could use roughly the following lines: in any (g, n)-controlled bad sequence of natural integers $k, l, m, \ldots$ over $\mathbb{N}$, once the first element $k \leq n$ has been fixed, the remaining elements $l, m, \ldots$ have to be chosen inside a finite set $\{0, \ldots, k-1\}$ of cardinal k—or the sequence would be good. Thus this suffix, which itself has to be bad, is of length at most

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

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

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

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

Item 1 motivates the following definition:

residual nwqo

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

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

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

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

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

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

Proposition 2.11 (Descent Equation). Descent Equation

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

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

Example 2.12. Let us consider the case of $L_{g,[k]}(n)$ for $k \leq n + 1$: by induction on k, we can see that

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

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

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

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

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

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

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

2.3.2 Reflecting nwqos

The reader might have noticed that Example 2.13 does not quite follow the reasoning that led to (2.29) on page 62: although we started by decomposing bad sequences into a first element and a suffix as in the Descent Equation, we rather used (2.31) to treat the suffix by seeing it as a bad sequence over Γ_n and to deduce the value of $L_{g,\mathbb{N}}(n)$. However, as already mentioned in Example 2.3 on page 57, $\Gamma_n \not\equiv [n]$ in general.

We can reconcile the analyses made for (2.29) on page 62 and in Example 2.13 by noticing that bad sequences are never shorter in Γ_n than in $[n]$. We will prove this formally using *reflections*, which let us simplify instances of the Descent Equation by replacing all A/a for $a \in A_{\leq n}$ by a single (or a few) A' that is larger than any of the considered A/a's—but still reasonably small compared to A, so that a well-founded inductive reasoning remains possible.

nwqo reflection

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

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

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

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

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

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

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

Reflections preserve controlled bad sequences. Let $h\colon A \hookrightarrow B$, consider a sequence $s = a_0, a_1, \ldots$ over A, and write $h(s)$ for $h(a_0), h(a_1), \ldots$, a sequence over B. Then by (2.36), $h(s)$ is bad when s is, and by (2.37), it is (g,n)-controlled when s is. Hence we can complete the picture of the monotonicity properties of L started in Remark 2.2 on page 56:

Proposition 2.16 (Monotonicity of L in A).

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

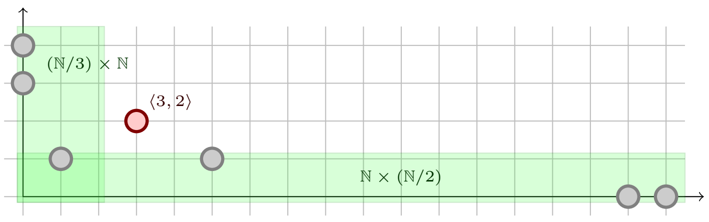

$(\mathbb{N}/3)\times\mathbb{N}$
$\langle 3,2\rangle$
$\mathbb{N}\times(\mathbb{N}/2)$

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

This is the last missing piece for deducing (2.29) from (2.31): since $[k]\hookrightarrow\Gamma_k$, $L_{g,[k]}(n)\leq L_{g,\Gamma_k}(n)$ by Proposition 2.16—the converse inequality holds for $k\leq n+1$, 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 Precongruence). Reflections are compatible with product and sum:

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

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

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

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

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

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

and reflections are not required.

The case of Cartesian products $A\times B$ is different: Let $g(x)=2x$ and consider the following $(g,3)$-controlled bad sequence over $\mathbb{N}^2$

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

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

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

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

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

We could have put $\langle 1,1\rangle$ in either $\mathbb{N} \times (\mathbb{N}/2)$ or $(\mathbb{N}/3) \times \mathbb{N}$ but we had no choice for the other elements of the suffix. Observe that the two subsequences $\langle 0,4\rangle\langle 0,3\rangle$ and $\langle 5,1\rangle, \langle 17,0\rangle, \langle 1,1\rangle, \langle 16,0\rangle$ are indeed bad, but not necessarily $(g, g(3))$-controlled: $|\langle 17,0\rangle| = 17 \geq 12 = g(g(3))$. However, we do not see them as *independent* sequences but consider their disjoint sum instead, so that their elements inherit their positions from the original sequence, and indeed the suffix sequence in (2.43) is $(g, g(3))$-controlled.

By a straightforward generalisation of the argument:

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

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

2.3.3 A Bounding Function

It is time to wrap up our analysis of L. We first combine the inductive residuation and reflection operations into *derivation relations* ∂_n: intuitively, the relation $A\ \partial_n\ A'$ is included in the relation "$A/a \hookrightarrow A'$ for some $a \in A_{\leq n}$" (see Lemma 2.19 for the formal statement). More to the point, the derivation relation captures a *particular* way of reflecting residuals, which enjoys some good properties: for every n, given A a nwqo in *polynomial normal form* (recall Remark 2.7 on page 58), $\partial_n A$ is a *finite* set of polynomial nwqos also in PNF, defined inductively by

nwqo derivation

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

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

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

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

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

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

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

The definition of ∂_n verifies:

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

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

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

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

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

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

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

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

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

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

in accordance with (2.47).

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

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

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

bounding function

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

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

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

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

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

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

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

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

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

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

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

by definition of M. □

2.4 ⋆ Classification in the Grzegorczyk Hierarchy

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

2.4.1 Maximal Order Types

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

order type

maximal order type

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

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

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

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

Ordinal Terms. Let ε_0 be the supremum of the family of ordinals $\{0, 1, \omega, \omega^\omega, \omega^{\omega^\omega}, \ldots\}$ (in other words ε_0 is the smallest solution of the equation $\omega^x = x$). It is well-known that ordinals below ε_0 can be written down in a canonical way as ordinal terms in *Cantor Normal Form* (CNF), i.e. sums

Cantor Normal Form

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

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

successor ordinal

limit ordinal

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

natural sum

natural product

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

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

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

Fact 2.23. *For any wqos A and B*

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

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

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

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

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

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

Equation (2.56) restates (2.46) and (2.47) using maximal order types, while (2.57) and (2.58) mirror respectively (2.49) and (2.50) but work in ω^ω; one easily obtains the following slight variation of Proposition 2.20:

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

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

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

Example 2.26. One can check that

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

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

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

2.4.2 The Cichoń Hierarchy

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

fundamental sequence

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

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

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

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

ordinal predecessor

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

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

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

Observe that predecessors of ordinals in ω^ω are very similar to our derivatives: for $d = 0$, $P_n(\omega^d) = 0$ and otherwise $P_n(\omega^d) = \omega^{d-1} \cdot n + P_n(\omega^{d-1})$, which is somewhat similar to (2.56), and more generally (2.64) is reminiscent of (2.57) but chooses a particular strategy: always derive the ω^d summand with the smallest d. The relationship will be made more precise in Section 2.4.3 on the following page.

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

Cichoń hierarchy

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

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

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

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

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

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

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

$$H_{\omega\cdot 2}(x) = H_{\omega+x+1}(x) = H_\omega(2x+1) + x = 3x + 1\,, \tag{2.67}$$
$$H_{\omega^2}(x) = (2^{x+1} - 1)(x+1)\,. \tag{2.68}$$

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

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

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

2.4.3 Monotonicity

One obstacle subsists before we can finally prove the Length Function Theorem: the functions $M_{g,\alpha}$ and h_α are not monotone in the parameter α. Indeed, $\alpha' < \alpha$ does *not* imply $M_{g,\alpha'}(n) \leq M_{g,\alpha}(n)$ for all n: witness the case $\alpha = \omega$ and $\alpha' = n+2$: $M_{g,\omega}(n) = 1+M_{g,n}(g(n)) = 1+n$ but $M_{g,n+2}(n) = n+2$ by Example 2.26. Similarly with h_α, as seen with Example 2.28, $h_{x+2}(x) = x+2 > x+1 = h_\omega(x)$, although $x + 2 < \omega$.

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

structural ordering

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

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

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

The structural ordering rules out the previous examples, as $x + 2 \not\sqsubseteq \omega$ for any x. This refined ordering yields the desired monotonicity property for M—see Lemma 2.31 next (it can also be proven for h; see Exercise 2.11)—but let us first introduce some notation: we write $\alpha' = \partial_{d,n}\alpha$ if $\alpha = \gamma \oplus \omega^d$ and $\alpha' = \gamma \oplus \partial_n\omega^d$. Then (2.58) can be rewritten as

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

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

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

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

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

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

□

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

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

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

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

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

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

The claim itself is easy to check using (2.70): abusing notations for the cases of $d = 0$ or $d' = 0$,

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

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

Let us conclude with a comparison between derivatives and predecessors:

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

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

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

and the statement holds. Otherwise, by (2.70)

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

from which we deduce the result by Lemma 2.31. □

2.4.4 Wrapping Up

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

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

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

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

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

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

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

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

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

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

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

which is optimal (recall examples 2.26 and 2.28).

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 $d, x \geq 1$: super-homogeneous function

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

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

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

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

Corollary 2.36. *Let $g(x) = x + b$ for some $b > 0$, and fix $d, p \geq 0$. Then $L_{g,\mathbb{N}^d \times \Gamma_p}$ is bounded by a function in $\mathscr{F}_d$.*

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

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

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

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

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

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

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

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

How good are these upper bounds? We already noted that they were optimal for $\mathbb{N}$ in (2.75), and the sequence (2.1) extracted from the successive configurations of SIMPLE was an example of a bad sequence with length function in $\mathscr{F}_3$. Exercise 2.15 generalises SIMPLE to arbitrary dimensions d and control functions g and shows that a length $g_{\omega^d}(n)$ can be reached using the lexicographic ordering; this is very close to the upper bounds found for instance in (2.75) and Corollary 2.35. The next chapter will be devoted to complexity lower bounds, showing that for many decision problems, the enormous generic upper bounds we proved here are actually unavoidable.

Exercises

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

Exercise 2.2 (Fast-Growing Functions). ⋆

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

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

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

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

⋆ Grzegorczyk hierarchy zero function sum function projection function substitution

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

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

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

limited primitive recursion

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

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

primitive recursion

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

cut-off subtraction

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

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

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

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

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

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

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

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

Exercise 2.4 (Complexity of `while` Programs)**.** Consider a program like SIMPLE that consists of a loop with variables ranging over $\mathbb{Z}$ and updates of linear complexity. Assume we obtain a k-ary disjunctive termination argument like (1.9) on page 20, where we synthesised linear ranking functions ρ_j into $\mathbb{N}$ for each T_j.

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

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

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

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

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

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

Exercise 2.8 (Well Foundedness of ∂)**.** Recall that, when working with terms in CNF, the *ordinal ordering* $<$, which is a well ordering over ordinals, has a syntactic characterisation akin to a lexicographic ordering: ordinal ordering

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

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

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

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

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

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. $\star$ r-good sequence r-bad sequence

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

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

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

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

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

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

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

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

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

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

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

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

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

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

★★ lexicographic ordering

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

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

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

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

for all control functions g and initial norms n.

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

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

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

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

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

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

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

(3) Deduce (¶).

(4) Show that, if $(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \ldots, (\mathbf{x}_\ell, c_\ell)$ is the run of $\textsc{lex}_d(g, n)$ 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 (Higman, 1952a), and so do Cichoń and Tahhan Bittar (1998) and Weiermann (1994); furthermore the latter provides upper bounds for the more general Kruskal (Kruskal, 1960).

The hierarchy $(\mathscr{F}_k)_{k\geq 2}$ described as the Grzegorczyk hierarchy in Section 2.1.3 and Section 2.4 is actually due to Löb and Wainer (1970); its relationship with the original Grzegorczyk hierarchy $(\mathscr{E}^k)_k$ (Grzegorczyk, 1953) is that $\mathscr{F}_k = \mathscr{E}^{k+1}$ for all $k \geq 2$. There are actually some difference between our definition of $(F_k)_k$ and that of Löb and Wainer (1970), but it only impacts low indices $k < 2$, and our definition follows contemporary presentations. Maximal order types were defined by de Jongh and Parikh (1977), where the reader will find a proof of Fact 2.23. The Cichoń hierarchy was first published in (Cichoń and Tahhan Bittar, 1998), where it was called the *length hierarchy*. More material on subrecursive hierarchies can be found in textbooks (Rose, 1984; Fairtlough and Wainer, 1998; Odifreddi, 1999) and in Appendix A. Fact 2.29 is proven there as Equation (A.25), and Fact 2.30 is a consequence of lemmas A.7, A.10, and A.17. Exercises 2.4 and 2.12 are taken from (Figueira et al., 2011). Equation (¶) in Exercise 2.15 relates the Cichoń hierarchy with the length of bad sequences for the lexicographic ordering over $\mathbb{N}^d$, i.e. with the length of decreasing sequences over ω^d; this holds more generally for any ordinal below ε_0 along with an appropriate norm, see (Schmitz, 2014).

3

COMPLEXITY LOWER BOUNDS

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

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

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

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

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

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

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

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

3.1 Counter Machines

counter machine

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

Minsky machine

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

3.1.1 Extended Counter Machines

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

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

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

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

3.1.2 Operational Semantics

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

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

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

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

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

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

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

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

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

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

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

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

3.1.3 Lossy Counter Machines

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

Technically, it is more convenient to see lossy machines as counter machines with a different operational semantics (and not as a special class of machines): thus it is possible to use simultaneously the two semantics and relate them. Incrementing errors too are handled by introducing a different operational semantics, see Exercise 3.4.

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

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

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

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

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

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

$$M \vdash \sigma \rightarrow_{\text{std}} \sigma' \text{ implies } M \vdash \sigma \rightarrow_{\text{lossy}} \sigma'. \quad (3.3)$$

3.1.4 Behavioural Problems on Counter Machines

We consider the following decision problems:

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

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

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

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

3.2 Hardy Computations

Hardy hierarchy

The *Hardy hierarchy* $(H^\alpha : \mathbb{N} \rightarrow \mathbb{N})_{\alpha<\varepsilon_0}$ is a hierarchy of ordinal-indexed functions, much like the *Cichoń hierarchy* introduced in Section 2.4.2. Its definition and properties are the object of Exercise 2.13 on page 77, but let us recall the following:

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

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

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

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

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

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

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

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

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

They are also monotone in their parameter relatively to the *structural ordering* defined in Section 2.4.3 on page 72:

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

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

Hardy computation

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

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

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

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

3.2.1 Encoding Hardy Computations

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

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

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

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

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

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

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

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

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

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

3.2.2 Implementing Hardy Computations with Counter Machines

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

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

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

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

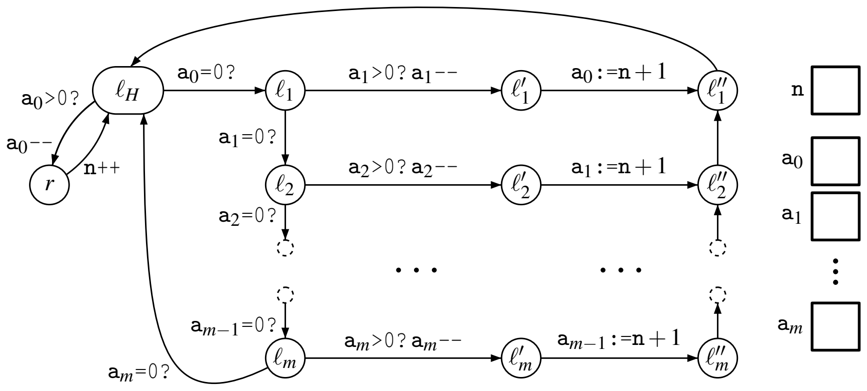


Figure 3.1: $M_H(m)$, a counter machine that implements $\xrightarrow{H}$.

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

The rules (D1–D2) can also be used from right to left. Used this way, they implement backward Hardy computations, i.e. they *invert* H. This is implemented by another counter machine, $M_{H^{-1}}(m) = (Loc_{H^{-1}}, C, \Delta_{H^{-1}})$, or $M_{H^{-1}}$ for short, defined pictorially in Figure 3.2.

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

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

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

3.3 Minsky Machines on a Budget

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

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

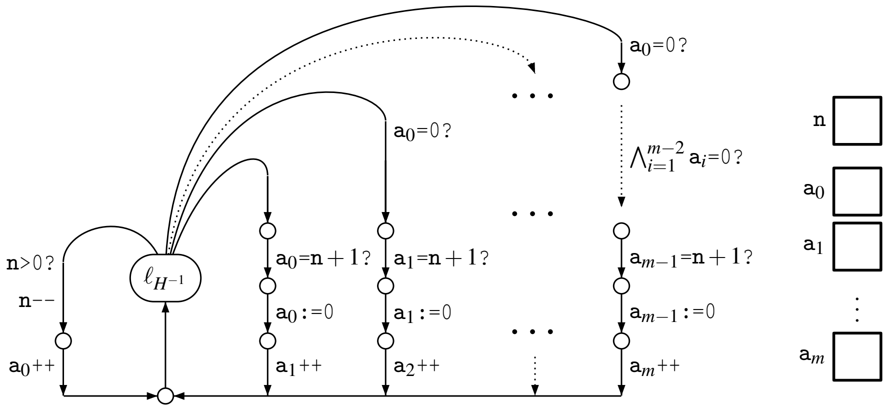


Figure 3.2: $M_{H^{-1}}(m)$, a counter machine that implements $\xrightarrow{H}{}^{-1}$.

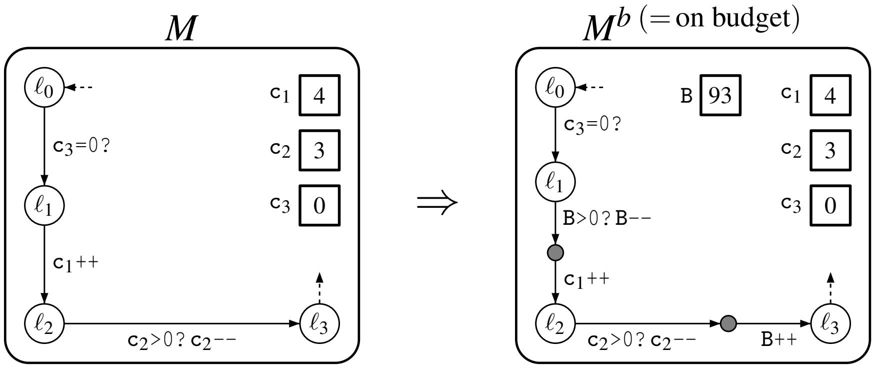


Figure 3.3: From M to M^b (schematically).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.4 Ackermann-Hardness for Lossy Counter Machines

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

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

Proposition 3.12. *The following are equivalent:*

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

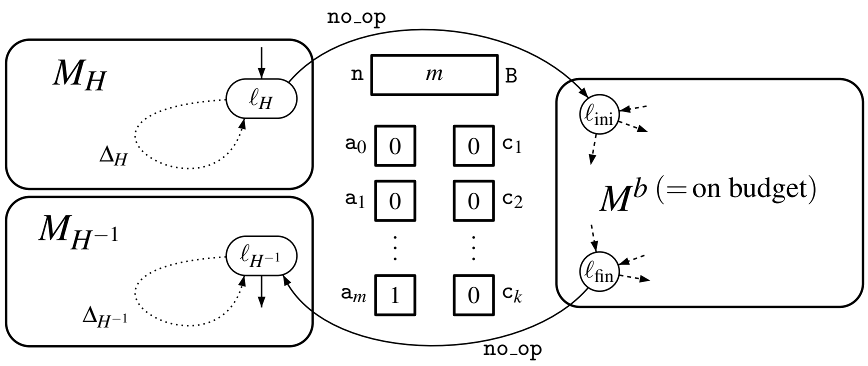


Figure 3.4: Constructing $M(m)$ from M^b, M_H and $M_{H^{-1}}$.

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

Proof Sketch.

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

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

  The first part yields $H^{\alpha(1,\mathbf{0})}(m) \geq H^{\alpha(\mathbf{a})}(x)$ (by Lemma 3.5.3), the third part $H^{\alpha(\mathbf{a}')}(x') \geq H^{\alpha(1,\mathbf{0})}(m)$ (by Lemma 3.6.3), and the middle part $B \geq B' + |\mathbf{c}|$ (by Corollary 3.11.1). Lossiness further implies $x \geq B$, $B' \geq x'$ and $\mathbf{a} \geq \mathbf{a}'$. Now, the only way to reconcile $H^{\alpha(\mathbf{a})}(x) \leq H^{\alpha(1,\mathbf{0})}(m) = Ack(m) \leq H^{\alpha(\mathbf{a}')}(x')$, $\mathbf{a}' \leq \mathbf{a}$, $x' \leq x$, and the monotonicity of F (Lemma 3.4) is by concluding $x = B = B' = x' = Ack(m)$ and $\mathbf{c} = \mathbf{0}$. Then the middle part of the run witnesses $M^b \vdash (\ell_{\text{ini}}, Ack(m), \mathbf{0}) \rightarrow^*_{\text{lossy}} (\ell_{\text{fin}}, Ack(m), \mathbf{0})$.
- "*2 ⇒ 5*" is Corollary 3.11.2.
- "*5 ⇒ 3*" is given by Lemma 3.9.
- "*3 ⇒ 4*" is obtained by stringing together reliable runs of the components, relying on lemmas 3.5.1 and 3.6.1 for the reliable runs of M_H and $M_{H^{-1}}$.

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

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

3.5 Handling Reset Petri Nets

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

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

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

3.5.1 Replacing Zero-Tests with Resets

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

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

Lemma 3.13.

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

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

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

Proposition 3.14. *The following are equivalent:*

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

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

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

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

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

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

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

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

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

3.5.2 From Extended to Minsky Machines

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

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

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

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

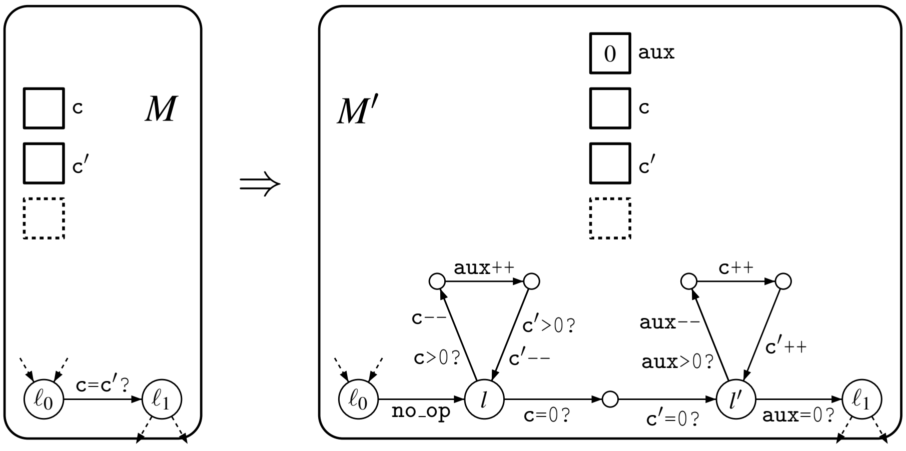


Figure 3.5: From M to M': eliminating equality tests.

2. $C' = C + \{\texttt{aux}\}$ 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 $\ell, \ell' \in Loc$ and vectors $\mathbf{c}, \mathbf{c}' \in \mathbb{N}^C$, the following holds:

$$M \vdash (\ell, \mathbf{c}) \rightarrow^*_{\text{std}} (\ell', \mathbf{c}') \text{ iff } M' \vdash (\ell, \mathbf{c}, 0) \rightarrow^*_{\text{std}} (\ell', \mathbf{c}', 0), \tag{3.11}$$
$$M \vdash (\ell, \mathbf{c}) \rightarrow^*_{\text{lossy}} (\ell', \mathbf{c}') \text{ iff } M' \vdash (\ell, \mathbf{c}, 0) \rightarrow^*_{\text{lossy}} (\ell', \mathbf{c}', 0). \tag{3.12}$$

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

We show here how to eliminate equality tests and leave the elimination of resets and copies as Exercise 3.2. Figure 3.5 shows, on a schematic example, how the transformation is defined.

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

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

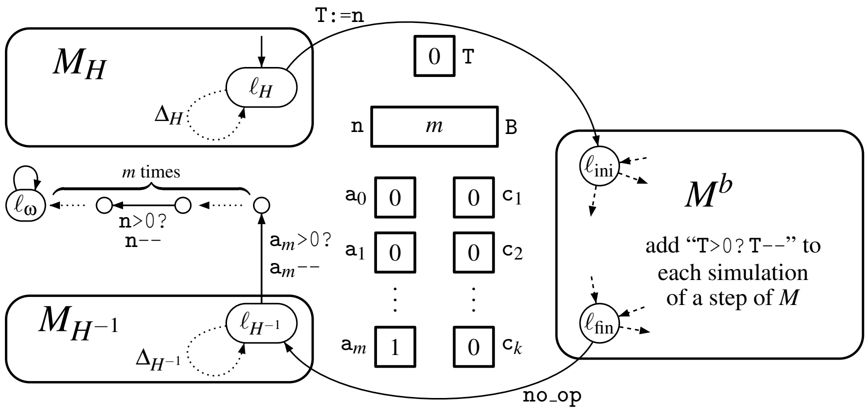


Figure 3.6: Hardness for termination: A new version of $M(m)$.

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

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

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

3.6 Hardness for Termination

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

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

Proposition 3.15. *The following are equivalent:*

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

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

Exercises

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

Exercise 3.2 (From Extended to Minsky Machines)**.** Complete the translation from extended counter machines to Minsky machines given in Section 3.5.2: provide gadgets for equality tests and resets.

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

transfer machine

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

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

Exercise 3.4 (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 $M \vdash \sigma \rightarrow_{\mathrm{inc}} \sigma'$, and defined by:

incrementing counter machine

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

Incrementation errors are thus the symmetrical mirror of losses.

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

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

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

Bibliographic Notes

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

We refer the reader to (Mayr, 2000; Schnoebelen, 2010b) for decidability issues for lossy counter machines. Reset nets (Araki and Kasami, 1976; Ciardo, 1994) are Petri nets extended with reset arcs that empty a place when the relevant transition is fired. Transfer nets (Ciardo, 1994) are instead extended with transfer arcs that move all the tokens from a place to another upon transition firing. Decidability issues for Transfer nets and Reset nets are investigated by Dufourd et al. (1999); interestingly, some problems are harder for Reset nets than for Transfer nets, although there exists an easy reduction from one to the others as far as the Hardness Theorem is concerned (see Exercise 3.3).

Using lossy counter machines, hardness results relying on the first half of the Hardness Theorem have been derived for a variety of logics and automata dealing with data words or data trees (Demri, 2006; Demri and Lazić, 2009; Jurdziński and Lazić, 2007; Figueira and Segoufin, 2009; Tan, 2010). Actually, these used reductions from counter machines with *incrementation* errors (see Exercise 3.4); although reachability for incrementing counter machines is Ackermann-hard, this does not hold for termination (Bouyer et al., 2012). Ackermann-hardness has also been shown by reductions from Reset and Transfer nets, relying on the second half of the Hardness Theorem (e.g. Amadio and Meyssonnier, 2002; Bresolin et al., 2012). The reader will find an up-to-date list of decision problems complete for Ackermann complexity in Appendix B.

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

4

IDEALS

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

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

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

4.1 Ideals of WQOs

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

principal filter

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

principal ideals

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

4.1.1 Prime Decompositions of Order-Closed Subsets

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

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

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

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

Proof. We prove 1, the downward-case being identical.
"$\Rightarrow$": By Definition 4.1 (and by induction on n) U prime and $U = U_1 \cup \cdots \cup U_n$ imply $U \subseteq U_i$ for some i, hence $U = U_i$.
"$\Leftarrow$": We check that U meets the criterion for being prime: assume that $U \subseteq U_1 \cup U_2$ for some $U_1, U_2 \in Up$. Then, letting $U'_i \overset{\text{def}}{=} U_i \cap U$ for $i = 1, 2$ gives $U = U'_1 \cup U'_2$. The assumption of the lemma entails that $U = U'_i$ for some $i \in \{1, 2\}$. Then $U \subseteq U_i$. □

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

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

Proof of 2. By well-founded induction on D. For this, recall that $(Down(X), \subseteq)$ is well-founded (Corollary 1.9). If D is empty, it is an empty, hence finite, union of primes. If $D \neq \emptyset$ is not prime, then by Lemma 4.2 it can be written as $D = D_1 \cup \cdots \cup D_n$ with $D \supsetneq D_i$ for all $i = 1, \ldots, n$. By induction hypothesis each D_i is a finite union of primes. Hence D is too. □

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

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

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

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

4.1.2 Ideals

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

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

We observe that every principal ideal $\downarrow x$ is up directed and is therefore an ideal, and similarly every principal filter $\uparrow x$ is down directed and is therefore a filter. Conversely, when $(X, \leq)$ is a wqo, any filter F is a principal filter: since it is upward-closed, by the Characterization of wqos, II 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 *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: *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: *Idl*$(\mathbb{N}) = \{\downarrow n \mid n \in \mathbb{N}\} \cup \{\mathbb{N}\}$.

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

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

1. *The up primes are exactly the filters.*

2. *The down primes are exactly the ideals.*

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

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

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

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

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

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

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

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

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

4.2 Effective Well Quasi-Orders and Ideals

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

This requires some basic computational assumptions on the wqos at hand. Such assumptions are often summarised informally as "*the wqo* $(X, \leq)$ *is effective*" and their precise meaning is often defined at a later stage, when one gives sufficient conditions based on the algorithm one is describing. This is just what we did with Proposition 1.36 or Proposition 1.38 in Section 1.9. Sometimes the effectiveness assumptions are not spelled out formally, e.g., when one has in mind applications where the wqo is $(\mathbb{N}^k, \leq_\times)$ or $(\Sigma^*, \leq_*)$ which are obviously "effective" under all expected understandings.

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

4.2.1 Effective WQOs

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

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

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

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

(ID) Inclusion of ideals is decidable.

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

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

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

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

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

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

Note also that the chosen representation for *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.

ideally effective wqo

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

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

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

(XF) Providing a filter decomposition of X,

(XI) Providing an ideal decomposition of X,

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

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

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

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

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

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

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

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

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

Proof. We explain how to obtain the missing procedures:

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

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

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

1. Initialise $U := \emptyset$;

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

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

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

Finally, the algorithm terminates since it builds a strictly increasing sequence of upward closed sets, which must be finite (see Corollary 1.9).

(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 Do we? 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 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 $(\omega{+}1) \smallsetminus \{0\}$, i.e., the set of non-null ordinals up to and including ω. A natural data structure for *Idl*$(\mathbb{N})$ is thus to use the number "n" to denote $\downarrow n$ and the symbol "ω" to denote $\mathbb{N}$.[1]

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

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

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

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

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

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

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

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

4.3.2 Sums and Products

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

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

Cartesian products behave equally nicely:

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

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

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

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

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

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

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

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

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

phic sum
product

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

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

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

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

4.3.3 Sequence Extensions

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

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

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

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

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

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

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

[2] See also Exercise 1.2.

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. $\square$

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

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

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

atoms

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

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

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

To show that these products generate all of *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 $X^* \smallsetminus \uparrow w$ is a product, and that (2) finite unions of products are closed under intersection.

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

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

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

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

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

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

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

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

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

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

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

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

$$
\begin{array}{rl}
\varepsilon\subseteq P & \text{always}\\
(I+\varepsilon).P\subseteq\varepsilon & \text{never}\\
D^*.P\subseteq\varepsilon & \text{iff } D=\emptyset\wedge P\subseteq\varepsilon\\
(I+\varepsilon).P\subseteq(I'+\varepsilon).P' & \text{iff } \big[I\subseteq I'\wedge P\subseteq P'\big]\vee\big[(I+\varepsilon).P\subseteq P'\big]\\
(I+\varepsilon).P\subseteq D^*.P' & \text{iff } \big[I\subseteq D\wedge P\subseteq P'\big]\vee\big[(I+\varepsilon).P\subseteq P'\big]\\
D^*.P\subseteq(I'+\varepsilon).P' & \text{iff } \big[D=\emptyset\wedge P\subseteq(I+\varepsilon).P'\big]\vee\big[D^*.P\subseteq P'\big]\\
D^*.P\subseteq D'^*.P' & \text{iff } \big[D\subseteq D'\wedge P\subseteq D'^*.P'\big]\vee\big[D^*.P\subseteq P'\big]
\end{array}
$$

Note that the above equalities directly lead to a dynamic programming procedure that will perform at most n^2 tests between atoms when comparing two ideals that are denoted by products of at most n atoms each.

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

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

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

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

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

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

4.4 Some Algorithmic Applications of Ideals

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

4.4.1 Forward Coverability

Our first ideal-based algorithm for coverability is a *forward coverability* one: it relies on the effectiveness of *Post* computations to avoid requiring effective pred-bases. Compared to the backward algorithm of Section 1.9.2 on page 17, 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

ideally *Post*-effective

Post computations as defined in (1.1) on page 16 can be carried over ideals. More precisely, if we define

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

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

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

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

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

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

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

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

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

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

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

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

4.4.2 Dual Backward Coverability

Our second algorithm for Cover proceeds like the backward algorithm of Section 1.9.2 on page 17, 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 (1.3) on page 18. In other words, given a Cover instance comprising a WSTS $\langle S, \rightarrow, \leq \rangle$ and two states s and t, the algorithm computes

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

where

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

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

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

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

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

VAS Coverability

Proposition 4.26 does not improve over Proposition 1.38, 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 61 down to 2ExpTime. We begin by investigating the general properties of the dual backward algorithm, before focusing on the case of VAS.

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

proper ideal

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

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

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

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

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

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

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

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

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

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

ω-set

ω-monotone

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

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

Control. As in Chapter 2, in order to derive combinatorial statements, we need to restrict our attention to *controlled* sequences of downward-closed subsets. We use as in Section 2.1 the infinity norm for ideal representations in $(\mathbb{N} \cup \{\omega\})^d$: $\|\mathbf{v}\| \stackrel{\text{def}}{=} \max_{1 \leq i \leq d | \mathbf{v}(i) < \omega} \mathbf{v}(i)$, and the maximal norm $\|D\| \stackrel{\text{def}}{=} \max \|I_j\|$ over the ideals in the canonical decomposition of downward-closed sets $D = \bigcup_j I_j$. We can observe that the sequence $D_0, D_1, \ldots$ constructed by the dual backward algorithm according to (4.12) is controlled. Let $\|\mathbf{A}\| \stackrel{\text{def}}{=} \max_{\mathbf{a} \in \mathbf{A}} \|\mathbf{a}\|$; we rely for this on propositions 4.16 and 4.18:

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

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

Upper Bound. We are now in position to state a refinement of Corollary 1.11 for ω-monotone controlled descending chains. For a control function $g\colon \mathbb{N} \to \mathbb{N}$, define the function $\widetilde{g}\colon \mathbb{N}^2 \to \mathbb{N}$ by induction on its first argument:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Exercises

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

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

(2) Is the reciprocal true?

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

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

Exercise 4.3 (Rado's Structure)**.** (1) What are the ideals of $(R, \leq_R)$, Rado's structure, seen in Definition 1.27 ?

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

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

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

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

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

Exercise 4.5 (Ideals of Cartesian Products)**.** (1) Prove the first part of Proposition 4.18: the ideals of $(A_1 \times A_2, \leq_\times)$ are exactly the sets of the form $I_1 \times I_2$ for $I_1 \in$ *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*$(A_1 +_{\text{lex}} A_2, \subseteq)$ is isomorphic to *Idl*$(A_1, \subseteq) +_{\text{lex}}$ *Idl*$(A_2, \subseteq)$.

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

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

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

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

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

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

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

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

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

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

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

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

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

Bibliographic Notes

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

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

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

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

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

Appendix

SUBRECURSIVE FUNCTIONS

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

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

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

A.1 Ordinal Terms

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

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

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

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

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

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

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

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

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

A.2 Fundamental Sequences and Predecessors

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

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

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

We assume ω_x to be the value in x of some monotone and expansive function s, typically $s(x) = x$—which we will hold as the standard one—or $s(x) = x + 1$.

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

We will see in Section A.6 how different choices for ω_x influence the hierarchies of functions built from them, in a simple case. Observe that, if $s(x) > 0$, then $\lambda_x > 0$.

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

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

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

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

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

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

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

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

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

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

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

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

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

A.3 Pointwise Ordering and Lean Ordinals

Pointwise ordering. An issue with ordinal-indexed hierarchies is that they are typically *not* monotonic in their ordinal index. A way to circumvent this problem is to refine the ordinal ordering; an especially useful refinement is $\prec_x$ defined for $x \in \mathbb{N}$ as the smallest transitive relation satisfying (see Dennis-Jones and Wainer (1984); Fairtlough and Wainer (1992); Cichoń and Tahhan Bittar (1998)):

$$\alpha \prec_x \alpha + 1\,, \qquad\qquad \lambda_x \prec_x \lambda\,. \tag{A.5}$$

In particular, using induction on α, one immediately sees that

$$0 \preccurlyeq_x \alpha\,, \tag{A.6}$$

$$P_x(\alpha) \prec_x \alpha\,. \tag{A.7}$$

The inductive definition of $\prec_x$ implies

$$\alpha' \prec_x \alpha \text{ iff } \begin{cases} \alpha = \beta + 1 \text{ is a successor and } \alpha' \preccurlyeq_x \beta, \text{ or} \\ \alpha = \lambda \text{ is a limit and } \alpha' \preccurlyeq_x \lambda_x. \end{cases} \tag{A.8}$$

Obviously $\prec_x$ is a restriction of $<$, the strict linear quasi-ordering over ordinal terms. For example, $\omega_x \prec_x \omega$ but $\omega_x + 1 \not\prec_x \omega$, although $\text{ord}(\omega_x + 1)$ is by definition a finite ordinal, smaller than $\text{ord}(\omega)$.

The $\prec_x$ relations are linearly ordered themselves

$$\prec_0 \subseteq \cdots \subseteq \prec_x \subseteq \prec_{x+1} \subseteq \cdots \tag{A.9}$$

and, over terms in CNF, $<$ can be recovered by

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

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

Lemma A.2. *For all α, α', γ in Ω and all x in $\mathbb{N}$*

$$\alpha' \prec_x \alpha \textit{ implies } \gamma + \alpha' \prec_x \gamma + \alpha \, , \tag{A.11}$$

$$\omega_x > 0 \textit{ and } \alpha' \prec_x \alpha \textit{ imply } \omega^{\alpha'} \prec_x \omega^{\alpha} \, . \tag{A.12}$$

Proof. All proofs are by induction over α (NB: the case $\alpha = 0$ is impossible).
(A.11): For the successor case $\alpha = \beta + 1$, this goes through

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

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

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

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

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

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

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

□

Lemma A.2 shows that $\prec_x$ is left congruent for $+$ and congruent for ω-exponentiation. One can observe that it is *not* right congruent for $+$; consider for instance the terms $\omega_x + 1$ and $\omega + 1$: one can see that $\omega_x + 1 \not\prec_x \omega + 1$. Indeed, from $\omega + 1$ the only way of descending through $\succ_x$ is $\omega + 1 \succ_x \omega \succ_x \omega_x$, but $\omega_x \not\succcurlyeq_x \omega_x + 1$ since $\prec_x \subseteq <$ for terms in CNF(ε_0).

Lemma A.3. *Let λ be a limit ordinal in Ω and $x < y$ in $\mathbb{N}$. Then $\lambda_x \preccurlyeq_y \lambda_y$, and if furthermore $\omega_x > 0$, then $\lambda_x \preccurlyeq_x \lambda_y$.*

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

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

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

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

Corollary A.4. *Let α, β in Ω and x, y in $\mathbb{N}$. If $x \leq y$ then $\alpha \prec_x \beta$ implies $\alpha \prec_y \beta$.*

In other words, $\prec_x \subseteq \prec_{x+1} \subseteq \prec_{x+2} \subseteq \cdots$ as claimed in (A.9).

If s is strictly increasing, i.e. if $\omega_x < \omega_{x+1}$ for all x, then the statement of Lemma A.3 can be strengthened to $\lambda_x \prec_y \lambda_y$ and $\lambda_y \prec_x \lambda_y$ when $\omega_x > 0$, and this hierarchy becomes strict at every level x: indeed, $\omega_{x+1} \prec_{x+1} \omega$ but $\omega_{x+1} \prec_x \omega$ would imply $\omega_{x+1} \preccurlyeq_x \omega_x$, contradicting $\prec_x \subseteq <$.

Lean Ordinals. Let k be in $\mathbb{N}$. We say that an ordinal α in CNF(ε_0) is *k-lean* if it only uses coefficients $\leq k$, or, more formally, when it is written under the strict form $\alpha = \omega^{\beta_1} \cdot c_1 + \cdots + \omega^{\beta_m} \cdot c_m$ with $c_i \leq k$ and, inductively, with k-lean β_i, this for all $i = 1, \ldots, m$. Observe that only 0 is 0-lean, and that any term in CNF is k-lean for some k.

A value k of particular importance for lean ordinal terms is $k = \omega_x - 1$: observe that this is the coefficient value introduced when we compute a predecessor ordinal at x. Stated differently, $(\omega_x - 1)$-leanness is an invariant of predecessor computations: if α is $(\omega_x - 1)$-lean, then $P_x(\alpha)$ is also $(\omega_x - 1)$-lean.

Leanness also provides a very useful characterisation of the $\prec_x$ relation in terms of the ordinal ordering over terms in CNF:

Lemma A.5. *Let x be in $\mathbb{N}$, and α in* $\mathrm{CNF}(\varepsilon_0)$ *be $(\omega_x - 1)$-lean. Then:*

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

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

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

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

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

$$\alpha < \gamma \text{ implies } \alpha \preccurlyeq_x P_x(\gamma)\ .$$

Proof. If $\alpha = 0$, we are done so we assume $\alpha > 0$ and hence $\omega_x > 1$, thus $\alpha = \sum_{i=1}^m \omega^{\beta_i} \cdot c_i$ with $m > 0$. Working with terms in CNF allows us to employ the syntactic characterisation of $<$ given in (A.14).

We prove the claim by induction on γ, considering two cases:

1. if $\gamma = \gamma' + 1$ is a successor then $\alpha < \gamma$ implies $\alpha \leq \gamma'$, hence $\alpha \overset{ih}{\preccurlyeq_x} \gamma' \overset{(A.2)}{=} P_x(\gamma)$.

2. if γ is a limit, we claim that $\alpha < \gamma_x$, from which we deduce $\alpha \overset{ih}{\preccurlyeq_x} P_x(\gamma_x) \overset{(A.2)}{=} P_x(\gamma)$. We consider three subcases for the claim:

   (a) if $\gamma = \omega^\lambda$ with λ a limit, then $\alpha = \sum_{i=1}^m \omega^{\beta_i} \cdot c_i < \gamma$ implies $\beta_1 < \lambda$, hence $\beta_1 \overset{ih}{\preccurlyeq_x} P_x(\lambda) = P_x(\lambda_x) < \lambda_x$, since β_1 is $(\omega_x - 1)$-lean. Thus $\alpha < \omega^{\lambda_x} = (\omega^\lambda)_x = \gamma_x$.

   (b) if $\gamma = \omega^{\beta+1}$ then $\alpha < \gamma$ implies $\beta_1 < \beta + 1$, hence $\beta_1 \leq \beta$. Now $c_1 \leq \omega_x - 1$ since α is $(\omega_x - 1)$-lean, hence $\alpha < \omega^{\beta_1} \cdot (c_1 + 1) \leq \omega^{\beta_1} \cdot \omega_x \leq \omega^\beta \cdot \omega_x = (\omega^{\beta+1})_x = \gamma_x$.

   (c) if $\gamma = \gamma' + \omega^\beta$ with $0 < \gamma', \beta$, then either $\alpha \leq \gamma'$, hence $\alpha < \gamma' + (\omega^\beta)_x = \gamma_x$, or $\alpha > \gamma'$, and then α can be written as $\alpha = \gamma' + \alpha'$ with $\alpha' < \omega^\beta$. In that case $\alpha' \overset{ih}{\preccurlyeq_x} P_x(\omega^\beta) \overset{(A.2)}{=} P_x((\omega^\beta)_x) < (\omega^\beta)_x$, hence $\alpha = \gamma' + \alpha' \overset{(A.14)}{<} \gamma' + (\omega^\beta)_x \overset{(A.1)}{=} (\gamma' + \omega^\beta)_x = \gamma_x$. □

A.4 Ordinal Indexed Functions

Let us recall several classical hierarchies from (Cichoń and Wainer, 1983; Cichoń and Tahhan Bittar, 1998). All the functions we define are over natural numbers. We introduce "relativized" versions of the hierarchies, which employ a unary *control function* $h : \mathbb{N} \to \mathbb{N}$; the "standard" hierarchies then correspond to the special case where the successor function $h(x) = x + 1$ is picked. We will see later in Section A.7 how hierarchies with different control functions can be related.

Hardy Functions. We define the functions $(h^\alpha)_{\alpha\in\Omega}$, each $h^\alpha \colon \mathbb{N} \to \mathbb{N}$, by inner iteration:

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

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

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

Cichoń Functions. Again for a unary h, we can define a variant $(h_\alpha)_{\alpha\in\Omega}$ of the Hardy functions called the *length hierarchy* by Cichoń and Tahhan Bittar (1998) and defined by inner and outer iteration:

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

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

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

Those hierarchies are the most closely related to the hierarchies of functions we define for the length of bad sequences.

Fast Growing Functions. Last of all, the *fast growing functions* $(f_\alpha)_{\alpha\in\Omega}$ are defined through

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

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

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

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

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

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

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

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

Proof. We only prove (A.21); (A.22) and (A.23) can be proven similarly.

By transfinite induction over α. For a successor ordinal $\alpha+1$, $h_{\alpha+1}(x) = 1 + h_\alpha(h(x)) = 1 + h_{P_x(\alpha+1)}(h(x))$. For a limit ordinal λ, $h_\lambda(x) = h_{\lambda_x}(x) \overset{ih}{=} 1 + h_{P_x(\lambda_x)}(h(x)) \overset{(A.2)}{=} 1 + h_{P_x(\lambda)}(h(x))$, where the ind. hyp. can applied since $0 < \lambda_x < \lambda$. □

Lemma A.7. *Let $h(x) > x$ for all x. Then for all α in Ω and x in $\mathbb{N}$ with $\omega_x > 0$,*

$$h_\alpha(x) \leq h^\alpha(x) - x\ .$$

Proof. By induction over α. For $\alpha = 0$, $h_0(x) = 0 = x - x = h^0(x) - x$. For $\alpha > 0$,

$$\begin{aligned}
h_\alpha(x) &= 1 + h_{P_x(\alpha)}(h(x)) && \text{(by Lemma A.6)}\\
&\leq 1 + h^{P_x(\alpha)}(h(x)) - h(x) && \text{(by ind. hyp. since } P_x(\alpha) < \alpha)\\
&\leq h^{P_x(\alpha)}(h(x)) - x && \text{(since } h(x) > x)\\
&= h^\alpha(x) - x\ . && \text{(by (A.22))}
\end{aligned}$$

□

Using the same argument, one can check that in particular for $h(x) = x + 1$,

$$H_\alpha(x) = H^\alpha(x) - x\ . \tag{A.24}$$

Lemma A.8. *For all α, γ in Ω, and x,*

$$h^{\gamma+\alpha}(x) = h^\gamma(h^\alpha(x))\ .$$

Proof. By transfinite induction on α. For $\alpha = 0$, $h^{\gamma+0}(x) = h^\gamma(x) = h^\gamma(h^0(x))$. For a successor ordinal $\alpha+1$, $h^{\gamma+\alpha+1}(x) = h^{\gamma+\alpha}(h(x)) \overset{ih}{=} h^\gamma(h^\alpha(h(x))) = h^\gamma\big(h^{\alpha+1}(x)\big)$. For a limit ordinal λ, $h^{\gamma+\lambda}(x) = h^{(\gamma+\lambda)_x}(x) = h^{\gamma+\lambda_x}(x) \overset{ih}{=} h^\gamma\big(h^{\lambda_x}(x)\big) = h^\gamma\big(h^\lambda(x)\big)$. □

Remark A.9. Some care should be taken with Lemma A.8: $\gamma+\alpha$ is not necessarily a term in CNF. See Remark A.15 on page 130 for a related discussion.

Lemma A.10. *For all β in Ω, and r, x in $\mathbb{N}$,*

$$h^{\omega^\beta \cdot r}(x) = f_\beta^r(x)\ .$$

Proof. In view of Lemma A.8 and $h^0 = f^0 = Id_\mathbb{N}$, it is enough to prove $h^{\omega^\beta} = f_\beta$, i.e., the $r = 1$ case. We proceed by induction over β.

For the base case. $h^{\omega^0}(x) = h^1(x) \overset{(A.19)}{=} f_0(x)$.

For a successor $\beta+1$. $h^{\omega^{\beta+1}}(x) \overset{(A.15)}{=} h^{(\omega^{\beta+1})_x}(x) = h^{\omega^\beta \cdot \omega_x}(x) \overset{ih}{=} f_\beta^{\omega_x}(x) \overset{(A.19)}{=} f_{\beta+1}(x)$.

For a limit λ. $h^{\omega^\lambda}(x) \overset{(A.15)}{=} h^{\omega^{\lambda_x}}(x) \overset{ih}{=} f_{\lambda_x}(x) \overset{(A.19)}{=} f_\lambda(x)$. □

A.5 Pointwise Ordering and Monotonicity

We set to prove in this section the main monotonicity and expansiveness properties of our various hierarchies.

Lemma A.11 (Cichoń and Tahhan Bittar, 1998). *Let h be an expansive monotone function. Then, for all α, α' in Ω and x, y in $\mathbb{N}$,*

$$x < y \textit{ implies } h_\alpha(x) \leq h_\alpha(y) \,, \tag{A.25}$$

$$\alpha' \prec_x \alpha \textit{ implies } h_{\alpha'}(x) \leq h_\alpha(x) \,. \tag{A.26}$$

Proof. Let us first deal with $\alpha' = 0$ for (A.26). Then $h_0(x) = 0 \leq h_\alpha(x)$ for all α and x.

Assuming $\alpha' > 0$, the proof now proceeds by simultaneous transfinite induction over α.

For 0. Then $h_0(x) = 0 = h_0(y)$ and (A.26) holds vacuously since $\alpha' \prec_x \alpha$ is impossible.

For a successor $\alpha + 1$. For (A.25), $h_{\alpha+1}(x) = 1 + h_\alpha(h(x)) \overset{ih(\text{A.25})}{\leq} 1 + h_\alpha(h(y)) = h_{\alpha+1}(y)$ where the ind. hyp. on (A.25) can be applied since h is monotone.

For (A.26), we have $\alpha' \preccurlyeq_x \alpha \prec_x \alpha + 1$, hence $h_{\alpha'}(x) \overset{ih(\text{A.26})}{\leq} h_\alpha(x) \overset{ih(\text{A.25})}{\leq} h_\alpha(h(x)) \overset{(\text{A.17})}{=} h_{\alpha+1}(x)$ where the ind. hyp. on (A.25) can be applied since $h(x) \geq x$.

For a limit λ. For (A.25), $h_\lambda(x) = h_{\lambda_x}(x) \overset{ih(\text{A.25})}{\leq} h_{\lambda_x}(y) \overset{ih(\text{A.26})}{\leq} h_{\lambda_y}(y) = h_\lambda(y)$ where the ind. hyp. on (A.26) can be applied since $\lambda_x \prec_y \lambda_y$ by Lemma A.3.

For (A.26), we have $\alpha' \preccurlyeq_x \lambda_x \prec_x \lambda$ with $h_{\alpha'}(x) \overset{ih(\text{A.26})}{\leq} h_{\lambda_x}(x) = h_\lambda(x)$. □

Essentially the same proof can be carried out to prove the same monotonicity properties for h^α and f_α. As the monotonicity properties of f_α will be handy in the remainder of the section, we prove them now:

Lemma A.12 (Löb and Wainer, 1970). *Let h be a function with $h(x) \geq x$. Then, for all α, α' in Ω, x, y in $\mathbb{N}$ with $\omega_x > 0$,*

$$f_\alpha(x) \geq h(x) \geq x \,. \tag{A.27}$$

$$\alpha' \prec_x \alpha \textit{ implies } f_{\alpha'}(x) \leq f_\alpha(x) \,, \tag{A.28}$$

$$x < y \textit{ and } h \textit{ monotone imply } f_\alpha(x) \leq f_\alpha(y) \,. \tag{A.29}$$

Proof of (A.27). By transfinite induction on α. For the base case, $f_0(x) = h(x) \geq x$ by hypothesis. For the successor case, assuming $f_\alpha(x) \geq h(x)$, then by induction on $n > 0$, $f_\alpha^n(x) \geq h(x)$: for $n = 1$ it holds since $f_\alpha(x) \geq h(x)$, and for $n+1$ since $f_\alpha^{n+1}(x) = f_\alpha(f_\alpha^n(x)) \geq f_\alpha(x)$ by ind. hyp. on n. Therefore $f_{\alpha+1}(x) = f_\alpha^{\omega_x}(x) \geq x$ since $\omega_x > 0$. Finally, for the limit case, $f_\lambda(x) = f_{\lambda_x}(x) \geq x$ by ind. hyp. □

Proof of (A.28). Let us first deal with $\alpha' = 0$. Then $f_0(x) = h(x) \leq f_\alpha(x)$ for all $x > 0$ and all α by (A.27).

Assuming $\alpha' > 0$, the proof proceeds by transfinite induction over α. The case $\alpha = 0$ is impossible. For the successor case, $\alpha' \preccurlyeq_x \alpha \prec_x \alpha + 1$ with $f_{\alpha+1}(x) = f_\alpha^{\omega_x - 1}(f_\alpha(x)) \overset{\text{(A.27)}}{\geq} f_\alpha(x) \overset{ih}{\geq} f_{\alpha'}(x)$. For the limit case, we have $\alpha' \preccurlyeq_x \lambda_x \prec_x \lambda$ with $f_{\alpha'}(x) \overset{ih}{\leq} f_{\lambda_x}(x) = f_\lambda(x)$. □

Proof of (A.29). By transfinite induction over α. For the base case, $f_0(x) = h(x) \leq h(y) = f_0(y)$ since h is monotone. For the successor case, $f_{\alpha+1}(x) = f_\alpha^{\omega_x}(x) \overset{\text{(A.27)}}{\leq} f_\alpha^{\omega_y}(x) \overset{ih}{\leq} f_\alpha^{\omega_y}(y) = f_{\alpha+1}(y)$ using $\omega_x \leq \omega_y$. For the limit case, $f_\lambda(x) = f_{\lambda_x}(x) \overset{ih}{\leq} f_{\lambda_x}(y) \overset{\text{(A.28)}}{\leq} f_{\lambda_y}(y) = f_\lambda(y)$, where (A.28) can be applied thanks to Lemma A.3. □

A.6 Different Fundamental Sequences

The way we employ ordinal-indexed hierarchies is as *standard* ways of classifying the growth of functions, allowing to derive meaningful complexity bounds for algorithms relying on wqos for termination. It is therefore quite important to use a standard assignment of fundamental sequences in order to be able to compare results from different sources. The definition provided in (A.1) is standard, and the two choices $\omega_x = x$ and $\omega_x = x + 1$ can be deemed as "equally standard" in the literature. We employed $\omega_x = x + 1$ in the rest of the notes, but the reader might desire to compare this to bounds using e.g. $\omega_x = x$—as seen in Lemma A.13, this is possible for strictly increasing h.

A bit of extra notation is needed: we want to compare the Cichoń hierarchies $(h_{s,\alpha})_{\alpha\in\Omega}$ for different choices of s. Recall that s is assumed to be monotone and expansive, which is true of the identity function *id*.

Lemma A.13. *Let α in Ω. If $s(h(x)) \leq h(s(x))$ for all x, then $h_{s,\alpha}(x) \leq h_{id,\alpha}(s(x))$ for all x.*

Proof. By induction on α. For 0, $h_{s,0}(x) = 0 = h_{id,0}(s(x))$. For a successor ordinal $\alpha + 1$, $h_{s,\alpha+1}(x) = 1 + h_{s,\alpha}(h(x)) \overset{ih}{\leq} 1 + h_{id,\alpha}(s(h(x))) \overset{\text{(A.25)}}{\leq} 1 + h_{id,\alpha}(h(s(x))) = h_{id,\alpha+1}(s(x))$ since $s(h(x)) \leq h(s(x))$. For a limit ordinal

λ, $h_{s,\lambda}(x) = h_{s,\lambda_x}(x) \overset{ih}{\leq} h_{id,\lambda_x}(s(x)) \overset{(A.26)}{\leq} h_{id,\lambda_{s(x)}}(s(x)) = h_{id,\lambda}(s(x))$ where $s(x) \geq x$ implies $\lambda_x \prec_{s(x)} \lambda_{s(x)}$ by Lemma A.3 and allows to apply (A.26). □

A simple corollary of Lemma A.13 for $s(x) = x + 1$ is that, if h is strictly monotone, then $h(x+1) \geq 1 + h(x)$, and thus $h_{s,\alpha}(x) \leq h_{id,\alpha}(x+1)$, i.e. the Cichoń functions for the two classical assignments of fundamental sequences are tightly related and will always fall in the same classes of subrecursive functions. This also justifies not giving too much importance to the choice of s—within reasonable limits.

A.7 Different Control Functions

As in Section A.6, if we are to obtain bounds in terms of a *standard* hierarchy of functions, we ought to provide bounds for $h(x) = x + 1$ as control. We are now in position to prove a statement of Cichoń and Wainer (1983):

Lemma A.14. *For all γ and α in Ω, if h is monotone eventually dominated by F_γ, then f_α is eventually dominated by $F_{\gamma+\alpha}$.*

Proof. By hypothesis, there exists x_0 (which we can assume wlog. verifies $x_0 > 0$) s.t. for all $x \geq x_0$, $h(x) \leq F_\gamma(x)$. We keep this x_0 constant and show by transfinite induction on α that for all $x \geq x_0$, $f_\alpha(x) \leq F_{\gamma+\alpha}(x)$, which proves the lemma. Note that $\omega_x \geq x \geq x_0 > 0$ and thus that we can apply Lemma A.12.

For the base case 0*:* for all $x \geq x_0$, $f_0(x) = h(x) \leq F_\gamma(x)$ by hypothesis.

For a successor ordinal $\alpha + 1$*:* we first prove that for all n and all $x \geq x_0$,

$$f_\alpha^n(x) \leq F_{\gamma+\alpha}^n(x) \ . \tag{A.30}$$

Indeed, by induction on n, for all $x \geq x_0$,

$$\begin{aligned}
f_\alpha^0(x) &= x = F_{\gamma+\alpha}^0(x)\\
f_\alpha^{n+1}(x) &= f_\alpha(f_\alpha^n(x))\\
&\leq f_\alpha\big(F_{\gamma+\alpha}^n(x)\big) && \text{(by (A.29) on f_α and the ind. hyp. on n)}\\
&\leq F_{\gamma+\alpha}\big(F_{\gamma+\alpha}^n(x)\big)\\
&\quad \text{(since by (A.27) $F_{\gamma+\alpha}(x) \geq x \geq x_0$ and by ind. hyp. on α)}\\
&= F_{\gamma+\alpha}^{n+1}(x) \ .
\end{aligned}$$

Therefore

$$\begin{aligned}
f_{\alpha+1}(x) &= f_\alpha^x(x)\\
&\leq F_{\gamma+\alpha}^x(x) && \text{(by (A.30) for $n = x$)}\\
&= F_{\gamma+\alpha+1}(x) \ .
\end{aligned}$$

For a limit ordinal λ*:* for all $x \geq x_0$, $f_\lambda(x) = f_{\lambda_x}(x) \overset{ih}{\leq} F_{\gamma+\lambda_x}(x) = F_{(\gamma+\lambda)_x}(x) = F_{\gamma+\lambda}(x)$. □

Remark A.15. Observe that the statement of Lemma A.14 is one of the few instances in this appendix where ordinal term notations matter. Indeed, nothing forces $\gamma + \alpha$ to be an ordinal term in CNF. Note that, with the exception of Lemma A.5, all the definitions and proofs given in this appendix are compatible with arbitrary ordinal terms in Ω, and not just terms in CNF, so this is not a formal issue.

The issue lies in the intuitive understanding the reader might have of a term "$\gamma + \alpha$", by interpreting $+$ as the direct sum in ordinal arithmetic. This would be a mistake: in a situation where two different terms α and α' denote the same ordinal $ord(\alpha) = ord(\alpha')$, we do not necessarily have $F_\alpha(x) = F_{\alpha'}(x)$: for instance, $\alpha = \omega^{\omega^0}$ and $\alpha' = \omega^0 + \omega^{\omega^0}$ denote the same ordinal ω, but $F_\alpha(2) = F_2(2) = 2^2 \cdot 2 = 2^3$ and $F_{\alpha'}(2) = F_3(2) = 2^{2^{2} \cdot 2} \cdot 2^2 \cdot 2 = 2^{11}$. Therefore, the results on ordinal-indexed hierarchies in this appendix should be understood *syntactically* on ordinal terms, and not semantically on their ordinal denotations.

The natural question at this point is: how do these new fast growing functions compare to the functions indexed by terms in CNF? Indeed, we should check that e.g. $F_{\gamma+\omega^p}$ with $\gamma < \omega^\omega$ is multiply-recursive if our results are to be of any use. The most interesting case is the one where γ is finite but α infinite (which will be used in the proof of Lemma A.17):

Lemma A.16. *Let* $\alpha \geq \omega$ *and* $0 < \gamma < \omega$ *be in* CNF(ε_0)*, and* $\omega_x \overset{\text{def}}{=} x$*. Then, for all* x*,* $F_{\gamma+\alpha}(x) \leq F_\alpha(x+\gamma)$*.*

Proof. We first show by induction on $\alpha \geq \omega$ that

Claim A.16.1. Let $s(x) \overset{\text{def}}{=} x + \gamma$. Then for all x, $F_{id,\gamma+\alpha}(x) \leq F_{s,\alpha}(x)$.

base case for ω*:* $F_{id,\gamma+\omega}(x) = F_{id,\gamma+x}(x) = F_{s,\omega}(x)$,

successor case $\alpha + 1$*:* with $\alpha \geq \omega$, an induction on n shows that $F^n_{id,\gamma+\alpha}(x) \leq F^n_{s,\alpha}(x)$ for all n and x using the ind. hyp. on α, thus $F_{id,\gamma+\alpha+1}(x) = F^x_{id,\gamma+\alpha}(x) \overset{(A.27)}{\leq} F^{x+\gamma}_{id,\gamma+\alpha}(x) \leq F^{x+\gamma}_{s,\alpha}(x) = F_{s,\alpha+1}(x)$,

limit case $\lambda > \omega$*:* $F_{id,\gamma+\lambda}(x) = F_{id,\gamma+\lambda_x}(x) \overset{ih}{\leq} F_{s,\lambda_x}(x) \overset{(A.28)}{\leq} F_{s,\lambda_{x+\gamma}}(x) = F_{s,\lambda}(x)$ where (A.28) can be applied since $\lambda_x \prec_x \lambda_{x+\gamma}$ by Lemma A.3 (applicable since $s(x) = x + \gamma > 0$).

Returning to the main proof, note that $s(x + 1) = x + 1 + \gamma = s(x) + 1$, allowing to apply Lemma A.13, thus for all x,

$$\begin{aligned}
F_{\mathit{id},\gamma+\alpha}(x) &\leq F_{s,\alpha}(x) && \text{(by the previous claim)}\\
&= H_s^{\omega^\alpha}(x) && \text{(by Lemma A.10)}\\
&\leq H_{\mathit{id}}^{\omega^\alpha}(s(x)) && \text{(by Lemma A.13 and (A.24))}\\
&= F_{\mathit{id},\alpha}(s(x))\;. && \text{(by Lemma A.10)}
\end{aligned}$$

□

A.8 Classes of Subrecursive Functions

We finally consider how some natural classes of recursive functions can be characterised by closure operations on subrecursive hierarchies. The best-known of these classes is the *extended Grzegorczyk hierarchy* $(\mathscr{F}_\alpha)_{\alpha\in\mathrm{CNF}(\varepsilon_0)}$ defined by Löb and Wainer (1970) on top of the fast-growing hierarchy $(F_\alpha)_{\alpha\in\mathrm{CNF}(\varepsilon_0)}$ for $\omega_x \overset{\text{def}}{=} x$.

Let us first provide some background on the definition and properties of $\mathscr{F}_\alpha$. The class of functions $\mathscr{F}_\alpha$ is the closure of the constant, addition, projection (including identity), and F_α functions, under the operations of

substitution: if $h_0, h_1, \ldots, h_n$ belong to the class, then so does the function f defined by

$$f(x_1,\ldots,x_n) = h_0(h_1(x_1,\ldots,x_n),\ldots,h_n(x_1,\ldots,x_n))\;,$$

limited primitive recursion: if h_1, h_2, and h_3 belong to the class, then so does the function f defined by

$$\begin{aligned}
f(0,x_1,\ldots,x_n) &= h_1(x_1,\ldots,x_n)\;,\\
f(y+1,x_1,\ldots,x_n) &= h_2(y,x_1,\ldots,x_n,f(y,x_1,\ldots,x_n))\;,\\
f(y,x_1,\ldots,x_n) &\leq h_3(y,x_1,\ldots,x_n)\;.
\end{aligned}$$

The hierarchy is strict for $\alpha > 0$, i.e. $\mathscr{F}_{\alpha'} \subsetneq \mathscr{F}_\alpha$ if $\alpha' < \alpha$, because in particular $F_{\alpha'} \notin \mathscr{F}_\alpha$. For small finite values of α, the hierarchy characterises some well-known classes of functions:

- $\mathscr{F}_0 = \mathscr{F}_1$ contains all the linear functions, like $\lambda x.x+3$ or $\lambda x.2x$, along with many simple ones like *cut-off subtraction*: $\lambda xy.x \dot{-} y$, which yields $x-y$ if $x \geq y$ and 0 otherwise,[2] or simple predicates like *odd*: $\lambda x.x \mod 2$,[3]

- $\mathscr{F}_2$ is exactly the set of elementary functions, like $\lambda x.2^{2^x}$,

[2]By limited primitive recursion; first define $\lambda x.x \dot{-} 1$ by $0 \dot{-} 1 = 0$ and $(y+1) \dot{-} 1 = y$; then $x \dot{-} 0 = x$ and $x \dot{-} (y+1) = (x \dot{-} y) \dot{-} 1$.

[3]By limited primitive recursion: $0 \mod 2 = 0$ and $(y+1) \mod 2 = 1 \dot{-} (y \mod 2)$.

- $\mathscr{F}_3$ contains all the tetration functions, like $\lambda x.\, \underbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}_{x \text{ times}}$, etc.

The union $\bigcup_{\alpha<\omega} \mathscr{F}_\alpha$ is the set of primitive-recursive functions, while F_ω is an Ackermann-like non primitive-recursive function. Similarly, $\bigcup_{\alpha<\omega^\omega} \mathscr{F}_\alpha$ is the set of multiply-recursive functions with F_{ω^ω} a non multiply-recursive function.

The following properties (resp. Theorem 2.10 and Theorem 2.11 in (Löb and Wainer, 1970)) are useful: for all α, unary f in $\mathscr{F}_\alpha$, and x,

$$\alpha > 0 \text{ implies } \exists p,\ f(x) \leq F_\alpha^p(x+1)\ , \tag{A.31}$$

$$\exists p,\ \forall x \geq p,\ f(x) \leq F_{\alpha+1}(x)\ . \tag{A.32}$$

Also note that by (A.31), if a unary function g is dominated by some function g' in $\mathscr{F}_\alpha$ 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 $(\mathscr{F}_\alpha)_\alpha$:

Lemma A.17. *For all $\gamma > 0$ and α, if h is monotone and eventually dominated by a function in $\mathscr{F}_\gamma$, then*

1. *if $\alpha < \omega$, f_α is dominated by a function in $\mathscr{F}_{\gamma+\alpha}$, and*
2. *if $\gamma < \omega$ and $\alpha \geq \omega$, f_α is dominated by a function in $\mathscr{F}_\alpha$.*

Proof of 1. We proceed by induction on $\alpha < \omega$.

For the base case $\alpha = 0$: we have $f_0 = h$ dominated by a function in $\mathscr{F}_\gamma$ by hypothesis.

For the successor case $\alpha = k+1$: by ind. hyp. f_k is dominated by a function in $\mathscr{F}_{\gamma+k}$, thus by (A.31) there exists p s.t. $f_k(x) \leq F_{\gamma+k}^p(x+1) = F_{\gamma+k}^p \circ F_0(x)$. By induction on n, we deduce

$$f_k^n(x) \leq (F_{\gamma+k}^p \circ F_0)^n(x)\ ; \tag{A.33}$$

Therefore,

$$\begin{aligned} f_{k+1}(x) &= f_k^x(x) && \text{(A.34)}\\ &\overset{\text{(A.33)}}{\leq} (F_{\gamma+k}^p \circ F_0)^x(x) && \text{(A.35)}\\ &\overset{\text{(A.29)}}{\leq} F_{\gamma+k}^{(p+1)x+1}((p+1)x+1) && \text{(A.36)}\\ &= F_{\gamma+k+1}((p+1)x+1)\ , \end{aligned}$$

where the latter function $x \mapsto F_{\gamma+k+1}((p+1)x+1)$ is defined by substitution from $F_{\gamma+k+1}$, successor, and $(p+1)$-fold addition, and therefore belongs to $\mathscr{F}_{\gamma+k+1}$. $\square$

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, $f_\alpha(x) \overset{\text{(A.29)}}{\leq} f_\alpha(x + x_0) \leq F_\alpha(x + x_0 + \gamma + 1)$ for all x, where the latter function $x \mapsto F_\alpha(x + x_0 + \gamma + 1)$ is in $\mathscr{F}_\alpha$. $\square$

Bestiary

PROBLEMS OF ENORMOUS COMPLEXITY

Because their main interest lies in characterising which problems are efficiently solvable, most textbooks in complexity theory concentrate on the frontiers between tractability and intractability, with less interest for the "truly intractable" problems found in ExpTime and beyond. Unfortunately, many natural decision problems are not that tame and require to explore the uncharted classes outside the exponential hierarchy.

This appendix is based on (Schmitz, 2016c) and borrows its title from a survey by Friedman (1999), where the reader will find many problems living outside Elementary. We are however not interested in "creating" new problems of enormous complexity, but rather in classifying already known problems in some important stops related to the extended Grzegorczyk hierarchy. Because we wanted this appendix to be reasonably self-contained, we will recall several definitions found elsewhere in these notes.

B.1 Fast-Growing Complexities

Exponential Hierarchy. Let us start where most accounts on complexity stop: define the class of exponential-time problems as

$$\textsc{ExpTime} \stackrel{\text{def}}{=} \bigcup_c \textsc{DTime}\left(2^{n^c}\right)$$

and the corresponding nondeterministic and space-bounded classes as

$$\textsc{NExpTime} \stackrel{\text{def}}{=} \bigcup_c \textsc{NTime}\left(2^{n^c}\right)$$

$$\textsc{ExpSpace} \stackrel{\text{def}}{=} \bigcup_c \textsc{Space}\left(2^{n^c}\right).$$

Problems complete for ExpTime, like corridor tiling games (Chlebus, 1986) or equivalence of regular tree languages (Seidl, 1990), are *known* not to be in PTime, hence the denomination "truly intractable" or "provably intractable" in the literature.

We can generalise these classes of problems to the *exponential hierarchy*

$$k\text{-}\textsc{ExpTime} \overset{\text{def}}{=} \bigcup_c \textsc{DTime}\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{n^c}}}}}}_{k \text{ times}}\right),$$

with the nondeterministic and space-bounded variants defined accordingly. The union of the classes in this hierarchy is the class of *elementary* problems:

$$\textsc{Elementary} \overset{\text{def}}{=} \bigcup_k k\text{-}\textsc{ExpTime} = \bigcup_c \textsc{DTime}\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{n}}}}}}_{c \text{ times}}\right).$$

Note that we could as easily define Elementary in terms of nondeterministic time bounds, space bounds, alternation classes, etc. Our interest in this appendix lies in the problems found outside this class, for which suitable hierarchies need to be used.

The Extended Grzegorczyk Hierarchy $(\mathscr{F}_\alpha)_{\alpha<\varepsilon_0}$ is an infinite hierarchy of classes of functions f with argument(s) and images in $\mathbb{N}$ (Löb and Wainer, 1970). At the heart of each $\mathscr{F}_\alpha$ lies the αth *fast-growing function* $F_\alpha\colon \mathbb{N} \to \mathbb{N}$, which is defined by

$$F_0(x) \overset{\text{def}}{=} x+1\,, \qquad F_{\alpha+1}(x) \overset{\text{def}}{=} F_\alpha^{x+1}(x) = \overbrace{F_\alpha(F_\alpha(\cdots F_\alpha}^{x+1 \text{ times}}(x)))\,,$$
$$F_\lambda(x) \overset{\text{def}}{=} F_{\lambda_x}(x)\,,$$

where $\lambda_x < \lambda$ is the xth element of the *fundamental sequence* for the limit ordinal λ, defined by

$$(\gamma+\omega^{\beta+1})_x \overset{\text{def}}{=} \gamma+\omega^\beta \cdot x\,, \quad (\gamma+\omega^\lambda)_x \overset{\text{def}}{=} \gamma+\omega^{\lambda_x}\,.$$

For instance,

$$F_1(x) = 2x+1\,, \qquad F_2(x) = 2^{x+1}(x+1)-1\,,$$
$$F_3(x) > 2^{\cdot^{\cdot^{\cdot^2}}}\Big\}{\scriptstyle x \text{ times}}\,,$$

F_ω is an Ackermannian function,

F_{ω^ω} is a hyper-Ackermannian function, etc.

For $\alpha \geq 2$, each level of the extended Grzegorczyk hierarchy can be characterised as a class of functions computable with bounded resources

$$\mathscr{F}_\alpha = \bigcup_c \textsc{FDTime}\left(F_\alpha^c(n)\right), \tag{B.1}$$

the choice between deterministic and nondeterministic or between time-bounded and space-bounded computations being once more irrelevant because F_2 is already a function of exponential growth. In particular, F_α^c belongs to $\mathscr{F}_\alpha$ for every α and fixed c.

Every function f in $\mathscr{F}_\alpha$ is *honest*, i.e. can be computed in time elementary in itself (Wainer, 1970)—this is a variant of the *time constructible* or *proper complexity* functions found in the literature, but better suited for the high complexities we are considering. Every f is also eventually bounded by $F_{\alpha'}$ if $\alpha < \alpha'$, i.e. there exists a rank $x_{f,\alpha}$ s.t. for all $x_1, \ldots, x_n$, if $\max_i x_i \geq x_{f,\alpha}$, then $f(x_1, \ldots, x_n) \leq F_{\alpha'}(\max_i x_i)$. However, for all $\alpha' > \alpha > 0$, $F_{\alpha'} \notin \mathscr{F}_\alpha$, and the hierarchy $(\mathscr{F}_\alpha)_{\alpha<\varepsilon_0}$ is strict for $\alpha > 0$.

Important Stops. Although some deep results have been obtained on the lower classes,[1] we focus here on the non-elementary classes, i.e. on $\alpha \geq 2$, where we find for instance

$$\begin{aligned}
\mathscr{F}_2 &= \text{FElementary} ,\\
\bigcup_k \mathscr{F}_k &= \text{FPrimitive-Recursive} ,\\
\bigcup_k \mathscr{F}_{\omega^k} &= \text{FMultiply-Recursive} ,\\
\bigcup_{\alpha<\varepsilon_0} \mathscr{F}_\alpha &= \text{FOrdinal-Recursive} .
\end{aligned}$$

We are dealing here with classes of functions, but writing $\mathscr{F}_\alpha^*$ for the restriction of $\mathscr{F}_\alpha$ to $\{0,1\}$-valued functions, we obtain the classification of decision problems displayed in Figure B.1.

Unfortunately, these classes are not quite satisfying for some interesting problems, which are *non* elementary (resp. non primitive-recursive, or non multiply-recursive, ...), but only *barely* so. The issue is that complexity classes like e.g. $\mathscr{F}_3^*$, which is the first class that contains non-elementary problems, are very large: $\mathscr{F}_3^*$ contains for instance problems that require space F_3^{100}, more than a hundred-fold compositions of towers of exponentials. As a result, hardness for $\mathscr{F}_3$ cannot be obtained for the classical examples of non-elementary problems.

We therefore introduce *smaller* classes:

$$\mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{p \in \cup_{\beta<\alpha} \mathscr{F}_\beta} \text{DTime}\left(F_\alpha(p(n))\right) . \tag{B.2}$$

[1]See Ritchie (1963) for a characterisation of FLinSpace, and for variants see e.g. Cobham (1965); Bellantoni and Cook (1992) for FPTime, or the chapter by Clote (1999) for a survey of these techniques.

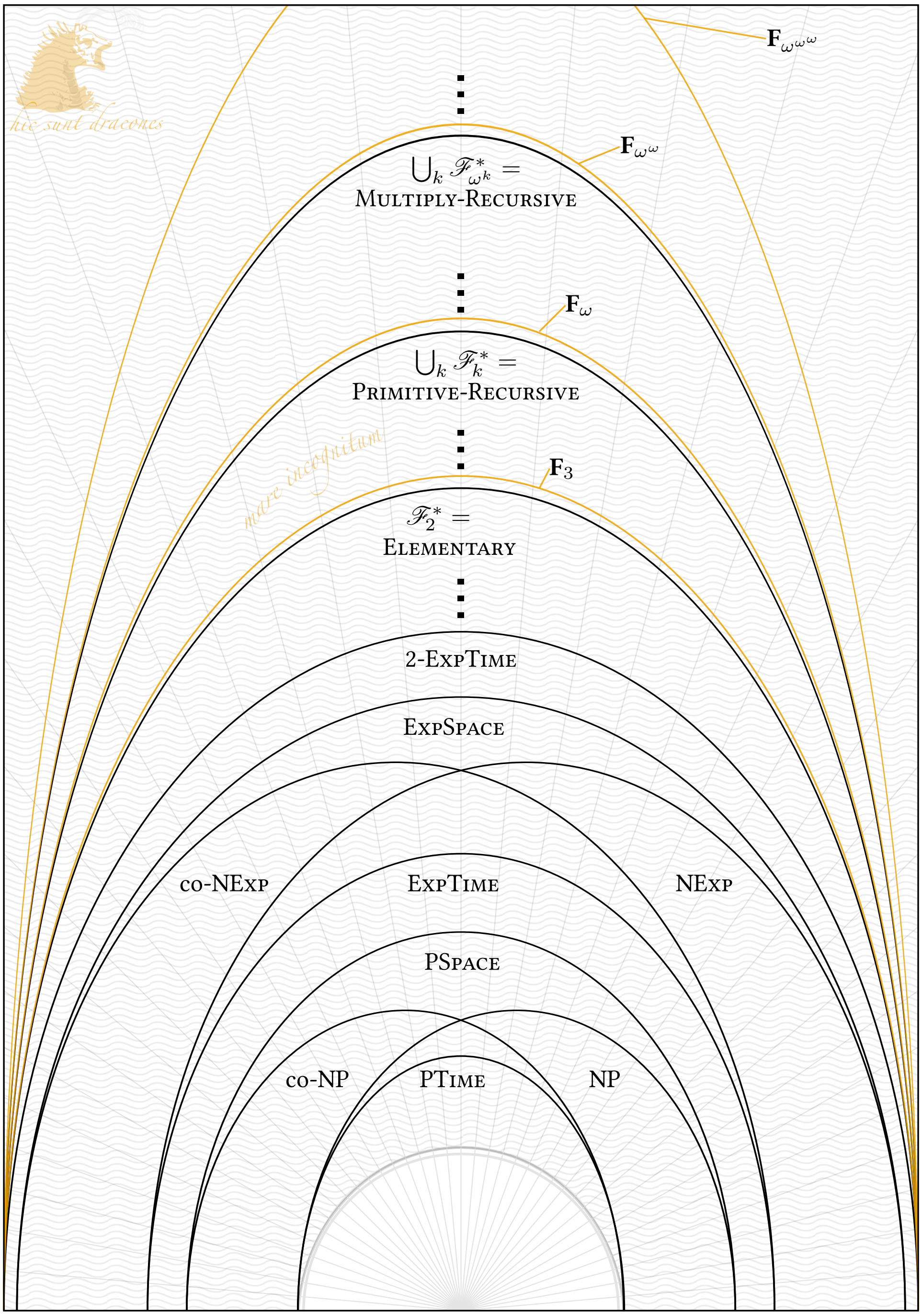


Figure B.1: Some complexity classes.

As previously, the choice of DTIME rather than NTIME or SPACE or ATIME is irrelevant for $\alpha \geq 3$. This yields for instance a class $\mathbf{F}_3$ of non-elementary decision problems closed under elementary reductions, a class $\mathbf{F}_\omega$ of Ackermannian problems closed under primitive-recursive reductions, a class $\mathbf{F}_{\omega^\omega}$ of hyper-Ackermannian problems closed under multiply-recursive reductions, etc.[2] We can name a few of these complexity classes:

$$\mathbf{F}_\omega = \text{ACKERMANN} \;,$$
$$\mathbf{F}_{\omega^\omega} = \text{HYPER-ACKERMANN} \;.$$

Of course, we could replace in (B.2) the class of reductions $\cup_{\beta<\alpha}\mathscr{F}_\beta$ by a more traditional one, like FLOGSPACE or FPTIME, or for $\alpha \geq \omega$ by primitive-recursive reductions in $\bigcup_k \mathscr{F}_k$ as done by Chambart (2011). However this definition better captures the intuition one can have of a problem being "complete for F_α."

A point worth making is that the extended Grzegorczyk hierarchy has multiple natural characterisations: as `loop` programs for $\alpha < \omega$ (Meyer and Ritchie, 1967), as ordinal-recursive functions with bounded growth (Wainer, 1970), as functions computable with restricted resources as in (B.1), as functions provably total in fragments of Peano arithmetic (Fairtlough and Wainer, 1998), etc.—which make the complexity classes we introduced here *meaningful*.

AN $\mathbf{F}_3$-COMPLETE EXAMPLE can be found in the seminal paper of Stockmeyer and Meyer (1973), and is quite likely already known by many readers. Define a *star-free expression* over some alphabet Σ as a term e with abstract syntax

star-free expression

$$e ::= a \mid \varepsilon \mid \emptyset \mid e+e \mid ee \mid \neg e$$

where a ranges over Σ and ε denotes the empty string. Such expressions are inductively interpreted as languages included in Σ^* by:

$$[\![a]\!] \stackrel{\text{def}}{=} \{a\} \qquad [\![\varepsilon]\!] \stackrel{\text{def}}{=} \{\varepsilon\} \qquad [\![\emptyset]\!] \stackrel{\text{def}}{=} \emptyset$$
$$[\![e_1+e_2]\!] \stackrel{\text{def}}{=} [\![e_1]\!] \cup [\![e_2]\!] \qquad [\![e_1e_2]\!] \stackrel{\text{def}}{=} [\![e_1]\!] \cdot [\![e_2]\!] \qquad [\![\neg e]\!] \stackrel{\text{def}}{=} \Sigma^* \smallsetminus [\![e]\!] \;.$$

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)

[2] An alternative class for $\alpha \geq 3$ is

$$\mathbf{F}'_\alpha \stackrel{\text{def}}{=} \bigcup_c \text{DTIME}\left(F_\alpha(n+c)\right) \;,$$

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* $(\mathscr{H}_\beta)_{\beta<\varepsilon_0}$ of functions (Wainer, 1972), which provides an infinite refinement of each $\mathscr{F}_\alpha$ as $\bigcup_{\beta<\omega^{\alpha+1}} \mathscr{H}_\beta$, but its classes lack both forms of robustness: any f in $\mathscr{H}_\beta$ is bounded by H^β the βth function of the *Hardy hierarchy*. What we define here as $\mathbf{F}_\alpha$ seems closer to $\bigcup_{\beta<\omega^\alpha\cdot 2} \mathscr{H}^*_\beta$.

show that this problem is hard for $2^{\cdot^{\cdot^{\cdot^2}}}\}^{\log n \text{ times}}$ space under FLogSpace reductions. Then, $\mathbf{F}_3$-hardness follows by an FElementary reduction from any Turing machine working in space $F_3(p(n))$ into a machine working in space $2^{\cdot^{\cdot^{\cdot^2}}}\}^{\log n \text{ times}}$. That the problem is in $\mathbf{F}_3$ can be checked using an automaton-based algorithm: construct automata recognising $[\![e_1]\!]$ and $[\![e_2]\!]$ respectively, using determinization to handle each complement operator at the expense of an exponential blowup, and check equivalence of the obtained automata in PSpace—the overall procedure is in space polynomial in $2^{\cdot^{\cdot^{\cdot^2}}}\}^{n \text{ times}}$, thus in $\mathbf{F}_3$.

B.2 Ackermann-Complete Problems

We gather here some decision problems that can be proven decidable in $\mathbf{F}_\omega$. The common trait of all these problems is their reliance on Dickson's Lemma over $\mathbb{N}^d$ for some d for decidability, and on the associated length function theorems (McAloon, 1984; Clote, 1986; Figueira et al., 2011; Abriola et al., 2015) for Ackermann upper bounds. The reader will find examples of $\mathbf{F}_\alpha$-complete problems for higher α in (Schmitz, 2016c).

B.2.1 Vector Addition Systems

Vector Addition Systems (VAS, and equivalently Petri nets), provided the first known Ackermannian decision problem: FCP.

A d-dimensional VAS is a pair $\langle \mathbf{v}_0, \mathbf{A}\rangle$ where $\mathbf{v}_0$ is an initial configuration in $\mathbb{N}^d$ and $\mathbf{A}$ is a finite set of transitions in $\mathbb{Z}^d$. A transition $\mathbf{u}$ in $\mathbf{A}$ can be applied to a configuration $\mathbf{v}$ in $\mathbb{N}^d$ if $\mathbf{v}' = \mathbf{v} + \mathbf{u}$ is in $\mathbb{N}^d$; the resulting configuration is then $\mathbf{v}'$. The complexity of decision problems for VAS usually varies from ExpSpace-complete (Lipton, 1976; Rackoff, 1978a; Blockelet and Schmitz, 2011) to $\mathbf{F}_\omega$-complete (Mayr and Meyer, 1981; Jančar, 2001) to undecidable (Hack, 1976; Jančar, 1995), via a key problem, whose exact complexity is unknown: VAS Reachability (Mayr, 1981; Kosaraju, 1982; Lambert, 1992; Leroux, 2011; Leroux and Schmitz, 2015).

[FCP] Finite Containment Problem

instance: Two VAS $\mathcal{V}_1$ and $\mathcal{V}_2$ known to have finite sets $\mathrm{Reach}(\mathcal{V}_1)$ and $\mathrm{Reach}(\mathcal{V}_2)$ of reachable configurations.

question: Is $\mathrm{Reach}(\mathcal{V}_1)$ included in $\mathrm{Reach}(\mathcal{V}_2)$?

lower bound: Mayr and Meyer (1981), from an F_ω-bounded version of Hilbert's Tenth Problem. A simpler reduction is given by Jančar (2001) from F_ω-MM the halting problem of F_ω-bounded Minsky machines.

upper bound: Originally McAloon (1984) and Clote (1986), or more generally using length function theorems for Dickson's Lemma (Figueira et al., 2011; Abriola et al., 2015).

comment: Testing whether the set of reachable configurations of a VAS is finite is ExpSpace-complete (Lipton, 1976; Rackoff, 1978a). FCP has been generalised by Jančar (2001) to a large range of behavioural relations between two VASs. Without the finiteness condition, these questions are undecidable (Hack, 1976; Jančar, 1995, 2001).

An arguably simpler problem on vector addition systems has recently been shown to be Ackermann-complete by Hofman and Totzke (2014). A *labelled vector addition system with states* (VASS) $\mathcal{V} = \langle Q, \Sigma, d, T, q_0, \mathbf{v}_0 \rangle$ is a VAS extended with a finite set Q of control states that includes a distinguished initial state q_0. The transitions in T of such systems are furthermore labelled with symbols from a finite alphabet Σ: transitions are then defined as quadruples $q \xrightarrow{a,\mathbf{u}} q'$ for a in Σ and $\mathbf{u}$ in $\mathbb{Z}^d$. Such a system defines an infinite labelled transition system $\langle Q \times \mathbb{N}^d, \rightarrow, (q_0, \mathbf{v}_0) \rangle$ where $(q, \mathbf{v}) \xrightarrow{a} (q', \mathbf{v} + \mathbf{u})$ if $q \xrightarrow{a,\mathbf{u}} q'$ is in T and $\mathbf{v} + \mathbf{u} \geq \mathbf{0}$. The *set of traces* of $\mathcal{V}$ is the set of finite sequences $L(\mathcal{V}) \stackrel{\text{def}}{=} \{a_1 \cdots a_n \in \Sigma^* \mid \exists (q, \mathbf{v}) \in Q \times \mathbb{N}^d . (q_0, \mathbf{v}_0) \xrightarrow{a_1 \cdots a_n} (q, \mathbf{v})\}$.

[1VASSU] One-Dimensional VASS Universality
instance: A one-dimensional labelled VASS $\mathcal{V} = \langle Q, \Sigma, 1, T, q_0, \mathbf{x}_0 \rangle$.
question: Does $L(\mathcal{V}) = \Sigma^*$, i.e. is every finite sequence over Σ a trace of $\mathcal{V}$?
lower bound: Hofman and Totzke (2014) by reduction from reachability in gainy counter machines, see LCM.
upper bound: Hofman and Totzke (2014) using length function theorems for Dickson's Lemma.
comment: One-dimensional VASS are also called "one counter nets" in the literature. More generally, the *inclusion* problem $L \subseteq L(\mathcal{V})$ for some rational language L is still Ackermann-complete.

B.2.2 Energy Games

A problem with a similar flavour to 1VASSU considers instead games played on (multi-)weighted graphs. Given a finite directed graph $G = (V, E)$, whose vertices $V = V_1 \uplus V_2$ are partitioned into Player 1 vertices and Player 2 vertices, and whose edges E are labelled by vectors in $\mathbb{Z}^d$ for some dimension d—representing discrete energy consumption and replenishment from d sources—, an initial vertex v of G, and an initial credit $\mathbf{u}$ in $\mathbb{N}^d$, we may consider an *energy objective* for Player 1: does she have a strategy ensuring an infinite play starting from v such that, at all times, $\mathbf{u}$ plus the vector labels of all the edges used so far is componentwise non-negative? (Note that the input to this problem could equivalently be seen as a labelled VASS with a partition of its control states into Player 1 and Player 2 states.) This problem is known to be 2ExpTime-complete (Jurdziński et al., 2015).

We are interested here in a variant where additionally Player 1 has only *partial observation*, meaning that she does not know the exact current vertex, but is only given an equivalence class of vertices that contains it. This variant of the game is still decidable (Degorre et al., 2010), and was shown $\mathbf{F}_\omega$-complete by Pérez (2016):

[POE] Energy Games with Partial Observation
instance: A multi-weighted finite game graph $G = (V, E)$ with labelling $\lambda\colon E \to \mathbb{Z}^d$, $v \in V$, $\mathbf{u} \in \mathbb{N}^d$, and $\equiv \subseteq V \times V$ an equivalence relation on vertices.
question: Does Player 1 have a winning strategy for the energy objective and compatible with $\equiv$, when starting from v with initial credit $\mathbf{u}$?
lower bound: Pérez (2016) by reduction from reachability in gainy counter machines, see LCM.
upper bound: Pérez (2016) by applying length function theorems for Dickson's Lemma to the decision algorithm of Degorre et al. (2010).
comment: Hardness already holds in dimension $d = 1$ and in the case of a *blind* game, i.e. where $v \equiv v'$ for all $v, v' \in V$.

B.2.3 Unreliable Counter Machines

A *lossy counter machine* (LCM) is syntactically a Minsky machine, but its operational semantics are different: its counter values can decrease nondeterministically at any moment during execution. See chapters 2 and 3 for details.

[LCM] Lossy Counter Machines Reachability
instance: A lossy counter machine M and a configuration σ.
question: Is σ reachable in M with lossy semantics?
lower bound: Schnoebelen (2010a), by a direct reduction from F_ω-bounded Minsky machines. The first proofs were given independently by Urquhart in 1999 and Schnoebelen in 2002.
upper bound: Length function theorems for Dickson's Lemma.
comment: Completeness also holds for terminating LCMs (meaning that every computation starting from the initial configuration terminates), coverability in Reset or Transfer Petri nets, and for reachability in *gainy* counter machines, where counter values can increase nondeterministically.

B.2.4 Relevance Logics

Relevance Logics provide different semantics of implication, where a fact B is said to follow from A, written "$A \to B$", only if A is actually *relevant* in the deduction of B. This excludes for instance $A \to (B \to A)$, $(A \wedge \neg A) \to B$, etc.—see Dunn and Restall (2002) for more details. Although the full logic $\mathbf{R}$ is undecidable (Urquhart, 1984), its conjunctive-implicative fragment $\mathbf{R}_{\to,\wedge}$ is decidable, and Ackermann-complete:

[CRI] Conjunctive Relevant Implication
instance: A formula A of $\mathbf{R}_{\to,\wedge}$.
question: Is A a theorem of $\mathbf{R}_{\to,\wedge}$?

lower bound: Urquhart (1999), from a variant of LCM: the emptiness problem of *alternating expansive counter machines*, for which he proved $\mathbf{F}_\omega$-hardness directly from F_ω-MM the halting problem in F_ω-bounded Minsky machines.
upper bound: Urquhart (1999) using length function theorem for Dickson's Lemma.
comment: Hardness also holds for any intermediate logic between $\mathbf{R}_{\to,\wedge}$ and $\mathbf{T}_{\to,\wedge}$, which might include some undecidable fragments. The related *contractive propositional linear logic* LLC and its additive-multiplicative fragment MALLC are also Ackermann-complete (Lazić and Schmitz, 2015b).

B.2.5 Data Logics & Register Automata

Data Logics and Register Automata are concerned with structures like words or trees with an additional equivalence relation over the positions. The motivation for this stems in particular from XML processing, where the equivalence stands for elements sharing the same *datum* from some infinite data domain $\mathbb{D}$. Enormous complexities often arise in this context, both for automata models (register automata and their variants, when extended with alternation or histories) and for logics (which include logics with *freeze* operators and XPath fragments)—the two views being tightly interconnected.

[A1RA] Emptiness of Alternating 1-Register Automata
instance: An A1RA $\mathcal{A}$.
question: Is the data language $L(\mathcal{A})$ empty?
lower bound: Demri and Lazić (2009), from reachability in gainy counter machines LCM.
upper bound: Demri and Lazić (2009), by reducing to reachability in gainy counter machines LCM.
comment: There exist many variants of the A1RA model, and hardness also holds for the corresponding data logics (e.g. Jurdziński and Lazić, 2007; Demri and Lazić, 2009; Figueira and Segoufin, 2009; Tan, 2010; Figueira, 2012; Tzevelekos and Grigore, 2013). The complexity rises to $\mathbf{F}_{\omega^\omega}$ in the case of linearly ordered data (Ouaknine and Worrell, 2007), and even to $\mathbf{F}_{\varepsilon_0}$ for data logics using multiple attributes with a hierarchical policy (Decker and Thoma, 2016).

B.2.6 Metric Temporal Logic

Metric Temporal Logic (MTL) allows to reason on *timed words* over $\Sigma \times \mathbb{R}$, where Σ is a finite alphabet and the real values are non decreasing *timestamps* on events (Koymans, 1990). When considering infinite timed words, one usually focuses on *non-Zeno* words, where the timestamps are increasing and unbounded. MTL is an extension of linear temporal logic where temporal modalities are decorated with real intervals constraining satisfaction; for instance, a timed word w satisfies the formula $\mathsf{F}_{[3,\infty)}\varphi$ at position i, written $w, i \models \mathsf{F}_{[3,\infty)}\varphi$, only if φ holds at some

timed words

position $j > i$ of w with timestamp $\tau_j - \tau_i \geq 3$. The *safety* fragment of MTL restricts the intervals decorating "until" modalities to be right-bounded.

[SMTL] Satisfiability of Safety Metric Temporal Logic
instance: A safety MTL formula φ.
question: Does there exist an infinite non-Zeno timed word w s.t. $w, 0 \models \varphi$?
lower bound: Lazić et al. (2016), by a direct reduction from F_ω-bounded Turing machines.
upper bound: Lazić et al. (2016) by resorting to length function theorems for Dickson's Lemma.
comment: The complexity bounds are established through reductions to and from the *fair termination* problem for insertion channel systems, which Lazić et al. (2016) show to be Ackermann-complete.

B.2.7 Ground Term Rewriting

A *ground term rewrite system with state* (sGTRS) maintains a finite ordered labelled tree along with a control state from some finite set. While most questions about ground term rewrite systems are decidable (Dauchet and Tison, 1990), the addition of a finite set of control states yields a Turing-powerful formalism. Formally, a sGTRS $\langle Q, \Sigma, R\rangle$ over a ranked alphabet Σ and a finite set of states Q is defined by a finite set of rules $R \subseteq (Q \times T(\Sigma))^2$ of the form $(q, t) \to (q', t')$ acting over pairs of states and trees, which rewrite a configuration $(q, C[t])$ into $(q', C[t'])$ in any context C.

Hague (2014) adds *age* labels in $\mathbb{N}$ to every node of the current tree. In the initial configuration, every tree node has age zero, and at each rewrite step $(q, C[t]) \to (q', C[t'])$, in the resulting configuration the nodes in t' have age zero, and the nodes in C see their age increment by one if $q \neq q'$ or remain with the same age as in $(q, C[t])$ if $q = q'$. A *senescent* sGTRS with *lifespan* k in $\mathbb{N}$ restricts rewrites to only occur in subtrees of age at most k, i.e. when matching $C[t]$ the age of the root of t is $\leq k$.

[SGTRS] State Reachability in Senescent Ground Term Rewrite Systems
instance: A senescent sGTRS $\langle Q, \Sigma, R\rangle$ with lifespan k, two states q_0 and q_f in Q, and an initial tree t_0 in $T(\Sigma)$.
question: Does there exist a tree t in $T(\Sigma)$ such that (q_f, t) is reachable from (q_0, t_0)?
lower bound: Hague (2014), from coverability in reset Petri nets, see LCM.
upper bound: Hague (2014), by reducing to coverability in reset Petri nets, see LCM.

B.2.8 Interval Temporal Logics

Interval Temporal Logics provide a formal framework for reasoning about temporal intervals. Halpern and Shoham (1991) define a logic with modalities expressing the basic relationships that can hold between two temporal intervals, $\langle B\rangle$ for "begun by", $\langle E\rangle$ for "ended by", and their inverses $\langle \bar{B}\rangle$ and $\langle \bar{E}\rangle$. This logic, and even small fragments of it, has an undecidable satisfiability problem, thus prompting the search for decidable restrictions and variants. Montanari et al. (2010) show that the logic with relations $A\bar{A}B\bar{B}$—where $\langle A\rangle$ expresses that the two intervals "meet", i.e. share an endpoint—, has an $\mathbf{F}_\omega$-complete satisfiability problem over finite linear orders:

[ITL] Finite Linear Satisfiability of $A\bar{A}B\bar{B}$
instance: An $A\bar{A}B\bar{B}$ formula φ.
question: Does there exist an interval structure $\mathcal{S}$ over some finite linear order and an interval I of $\mathcal{S}$ s.t. $\mathcal{S}, I \models \varphi$?
lower bound: Montanari et al. (2010), from reachability in lossy counter machines (LCM).
upper bound: Montanari et al. (2010), by reducing to reachability in lossy counter machines (LCM).
comment: Hardness already holds for the fragments $\bar{A}B$ and $\bar{A}\bar{B}$ (Bresolin et al., 2012).

REFERENCES

Abdulla, P.A., Čerāns, K., Jonsson, B., and Tsay, Y.K., 1996. General decidability theorems for infinite-state systems. In *LICS'96*, pages 313–321. IEEE. doi:10.1109/LICS.1996.561359. Cited on page 49.

Abdulla, P.A., Čerāns, K., Jonsson, B., and Tsay, Y.K., 2000. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1–2):109–127. doi:10.1006/inco.1999.2843. Cited on page 49.

Abdulla, P.A., Collomb-Annichini, A., Bouajjani, A., and Jonsson, B., 2004a. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65. doi:10.1023/B:FORM.0000033962.51898.1a. Cited on page 117.

Abdulla, P.A., Bouajjani, A., and d'Orso, J., 2008. Monotonic and downward closed games. *Journal of Logic and Computation*, 18(1):153–169. doi:10.1093/logcom/exm062. Cited on page 49.

Abdulla, P.A. and Jonsson, B., 1993. Verifying programs with unreliable channels. In *LICS'93*, pages 160–170. Cited on page 50.

Abdulla, P.A. and Nylén, A., 2000. Better is better than well: On efficient verification of infinite-state systems. In *LICS'00*, pages 132–140. Cited on page 37.

Abdulla, P.A., Deneux, J., Mahata, P., and Nylén, A., 2004b. Forward reachability analysis of timed Petri nets. In Lakhnech, Y. and Yovine, S., editors, *FORMATS/FTRTFT*, pages 343–362. Springer Verlag LNCS 3253. ISBN 3-540-23167-6. http://springerlink.metapress.com/openurl.asp?genre=article{&}issn=0302-9743{&}volume=3253{&}spage=343. Cited on page 50.

Abdulla, P.A., Aronis, S., Atig, M.F., Jonsson, B., Leonardsson, C., and Sagonas, K., 2015. Stateless model checking for TSO and PSO. In Baier, C. and Tinelli, C., editors, *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9035 of *Lecture Notes in Computer Science*, pages 353–367. Springer. ISBN 978-3-662-46680-3. doi:10.1007/978-3-662-46681-0_28. Cited on page 50.

Abriola, S., Figueira, S., and Senno, G., 2015. Linearizing well-quasi orders and bounding the length of bad sequences. *Theoretical Computer Science*, 603:3–22. doi:10.1016/j.tcs.2015.07.012. Cited on pages 79, 140.

Acciai, L. and Boreale, M., 2010. Spatial and behavioral types in the pi-calculus. *Information and Computation*, 208(10):1118–1153. Cited on page 51.

Adams, W.W. and Loustaunau, P., 1994. *An introduction to Gröbner bases*, volume 3 of *Graduate Studies in Mathematics*. American Mathematical Society. 289 pages. Cited on page 31.

Amadio, R. and Meyssonnier, Ch., 2002. On decidability of the control reachability problem in the asynchronous π-calculus. *Nordic Journal of Computing*, 9(2):70–101. Cited on page 96.

Araki, T. and Kasami, T., 1976. Some decision problems related to the reachability problem for Petri nets. *Theoretical Computer Science*, 3(1):85–104. doi:10.1016/0304-3975(76)90067-0. Cited on page 96.

Atig, M.F., Bouajjani, A., Burckhardt, S., and Musuvathi, M., 2010. On the verification problem for weak memory models. In Hermenegildo, M.V. and Palsberg, J., editors, *Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010,* *Madrid, Spain, January 17-23, 2010*, pages 7–18. ACM. ISBN 978-1-60558-479-9. doi:10.1145/1706299.1706303. Cited on page 50.

Bellantoni, S. and Cook, S., 1992. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2(2):97–110. doi:10.1007/BF01201998. Cited on page 137.

Berghofer, S., 2004. A constructive proof of Higman's lemma in Isabelle. In *TYPES'03*, pages 66–82. Springer Verlag LNCS 3085. Cited on page 29.

Bertrand, N. and Schnoebelen, Ph., 2013. Computable fixpoints in well-structured symbolic model checking. *Formal Methods in System Design*, 43(2):233–267. doi:10.1007/s10703-012-0168-y. Cited on page 49.

Blass, A. and Gurevich, Y., 2008. Program termination and well partial orderings. *ACM Transactions on Computational Logic*, 9(3):1–26. doi:10.1145/1352582.1352586. Cited on page 49.

Blockelet, M. and Schmitz, S., 2011. Model-checking coverability graphs of vector addition systems. In *MFCS 2011*, volume 6907 of *Lecture Notes in Computer Science*, pages 108–119. Springer. doi:10.1007/978-3-642-22993-0_13. Cited on pages 49, 140.

Blondin, M., Finkel, A., and McKenzie, P., 2014. Handling infinitely branching WSTS. In *ICALP 2014*, volume 8573 of *Lecture Notes in Computer Science*, pages 13–25. doi:10.1007/978-3-662-43951-7_2. Cited on page 117.

Blondin, M., Finkel, A., and McKenzie, P., 2016. Well behaved transition systems. Preprint. http://arxiv.org/abs/1608.02636. Cited on page 117.

Blondin, M., Finkel, A., and Goubault-Larrecq, J., 2017. Forward analysis for WSTS, Part III: Karp-Miller trees. In Lokam, S. and Ramanujam, R., editors, *Proceedings of the 36th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'17)*, Leibniz International Proceedings in Informatics, pages 16:1–16:15, Kanpur, India. Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2017.16. Cited on page 26.

Bouajjani, A., Calin, G., Derevenetc, E., and Meyer, R., 2015. Lazy TSO reachability. In Egyed, A. and Schaefer, I., editors, *Fundamental Approaches to Software Engineering - 18th International Conference, FASE 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9033 of *Lecture Notes in Computer Science*, pages 267–282. Springer. ISBN 978-3-662-46674-2. doi:10.1007/978-3-662-46675-9_18. Cited on page 50.

Bouyer, P., Markey, N., Ouaknine, J., Schnoebelen, Ph., and Worrell, J., 2012. On termination and invariance for faulty channel machines. *Formal Aspects of Computing*, 24(4):595–607. doi:10.1007/s00165-012-0234-7. Cited on page 96.

Bresolin, D., Della Monica, D., Montanari, A., Sala, P., and Sciavicco, G., 2012. Interval temporal logics over finite linear orders: The complete picture. In *ECAI 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 199–204. IOS Press. doi:10.3233/978-1-61499-098-7-199. Cited on pages 96, 145.

Buchberger, B., 1965. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal.* PhD thesis, Universität Innsbrück. Cited on page 34.

Buchberger, B. and Loos, R., 1982-1983. Algebraic simplification. In Buchberger, B., Collins, G.E., Loos, R., and Albrecht, R., editors, *Computer Algebra, Symbolic and Algebraic Computation.* Springer Verlag. Cited on page 31.

Cardoza, E., Lipton, R., and Meyer, A.R., 1976. Exponential space complete problems for Petri nets and commutative subgroups. In *STOC'76*, pages 50–54. ACM Press. doi:10.1145/800113.803630. Cited on page 49.

Chambart, P. and Schnoebelen, Ph., 2008. The ordinal recursive complexity of lossy channel systems. In *LICS 2008*, pages 205–216. IEEE. doi:10.1109/LICS.2008.47. Cited on page 96.

Chambart, P., 2011. *On Post's Embedding Problem and the complexity of lossy channels.* PhD thesis, ENS Cachan. http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/chambart-these11.pdf. Cited on page 139.

Charbit, P., 2016. Introduction to graphs minors, tree decompositions and FPT algorithms. https:

//www.irif.fr/~charbit/Enseignement/2016/coursmineurs2016.pdf. Cited on page 12.

Chlebus, B.S., 1986. Domino-tiling games. *Journal of Computer and System Sciences*, 32(3):374–392. doi:10.1016/0022-0000(86)90036-X. Cited on page 135.

Ciardo, G., 1994. Petri nets with marking-dependent arc cardinality: Properties and analysis. In *Petri nets '94*, volume 815 of *Lecture Notes in Computer Science*, pages 179–198. Springer. doi:10.1007/3-540-58152-9_11. Cited on page 96.

Cichoń, E.A. and Wainer, S.S., 1983. The slow-growing and the Grzecorczyk hierarchies. *Journal of Symbolic Logic*, 48(2):399–408. doi:10.2307/2273557. Cited on pages 119, 125, 129.

Cichoń, E.A. and Tahhan Bittar, E., 1998. Ordinal recursive bounds for Higman's Theorem. *Theoretical Computer Science*, 201(1–2):63–84. doi:10.1016/S0304-3975(97)00009-1. Cited on pages 79, 119, 121, 125, 127.

Clote, P., 1999. Computation models and function algebras. In Griffor, E.R., editor, *Handbook of Computability Theory*, volume 140 of *Studies in Logic and the Foundations of Mathematics*, chapter 17, pages 589–681. Elsevier. doi:10.1016/S0049-237X(99)80033-0. Cited on page 137.

Clote, P., 1986. On the finite containment problem for Petri nets. *Theoretical Computer Science*, 43: 99–105. doi:10.1016/0304-3975(86)90169-6. Cited on pages 79, 140.

Cobham, A., 1965. The intrinsic computational difficulty of functions. In *International Congress for Logic, Methodology and Philosophy of Science*, volume 2, pages 24–30. North-Holland. Cited on page 137.

Cook, B., Podelski, A., and Rybalchenko, A., 2011. Proving program termination. *Communications of the ACM*, 54:88–98. doi:10.1145/1941487.1941509. Cited on page 49.

Coquand, T. and Fridlender, D. A proof of Higman's lemma by structural induction. http://www.cse.chalmers.se/~coquand/open1.ps. Unpublished note. Cited on page 29.

Cotton-Barratt, C., Murawski, A.S., and Ong, C.H.L., 2017. ML and extended branching VASS. In *Proc. European Symposium on Programming (ESOP'17)*, pages 314–340. Springer Verlag LNCS 10201. Cited on page 50.

Dauchet, M. and Tison, S., 1990. The theory of ground rewrite systems is decidable. In *LICS '90*, pages 242–248. IEEE. doi:10.1109/LICS.1990.113750. Cited on page 144.

de Groote, P., Guillaume, B., and Salvati, S., 2004. Vector addition tree automata. In *Proc. 19th Annual IEEE Symposium on Logics in Computer Science.* IEEE Computer Society Press. Cited on page 50.

de Jongh, D.H.J. and Parikh, R., 1977. Well-partial orderings and hierarchies. *Indagationes Mathematicae*, 39(3):195–207. doi:10.1016/1385-7258(77)90067-1. Cited on page 79.

Decker, N. and Thoma, D., 2016. On freeze LTL with ordered attributes. In Jacobs, B. and Löding, C., editors, *FoSSaCS 2016*, volume 9634 of *Lecture Notes in Computer Science*, pages 269–284. Springer. doi:10.1007/978-3-662-49630-5_16. Cited on page 143.

Degorre, A., Doyen, L., Gentilini, R., Raskin, J.F., and Toruńczyk, S., 2010. Energy and mean-payoff games with imperfect information. In *CSL 2010*, volume 6247 of *Lecture Notes in Computer Science*, pages 260–274. doi:10.1007/978-3-642-15205-4_22. Cited on page 142.

Demri, S., 2006. Linear-time temporal logics with Presburger constraints: An overview. *Journal of Applied Non-Classical Logics*, 16(3–4):311–347. doi:10.3166/jancl.16.311-347. Cited on page 96.

Demri, S. and Lazić, R., 2009. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic*, 10(3). doi:10.1145/1507244.1507246. Cited on pages 96, 143.

Demri, S., Jurdziński, M., Lachish, O., and Lazić, R., 2012. The covering and boundedness problems for branching vector addition systems. *Journal of Computer and System Sciences*, 79(1):23–38. doi:10.1016/j.jcss.2012.04.002. Cited on page 51.

Dennis-Jones, E. and Wainer, S., 1984. Subrecursive hierarchies via direct limits. In Börger, E., Oberschelp, W., Richter, M., Schinzel, B., and Thomas, W., editors, *Computation and Proof Theory*, volume 1104 of *Lecture Notes in Mathematics*, pages 117–128. Springer. doi:10.1007/BFb0099482. Cited on pages 120, 121.

Dershowitz, N., 1979. A note on simplification orderings. *Inf. Proc. Letters*, 9(5):212–215. Cited on page 26.

Dershowitz, N. and Manna, Z., 1979. Proving termination with multiset orderings. *Communications of the ACM*, 22:465–476. Cited on page 32.

Dershowitz, N., 1982. Orderings for term-rewriting systems. *Theor. Comp. Sci.*, 17(3):279–301. Cited on page 26.

Dickson, L.E., 1913a. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422. doi:10.2307/2370405. Cited on page 49.

Dickson, L.E., 1913b. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422. doi:doi:10.2307/2370405. https://ia801908.us.archive.org/28/items/jstor-2370405/2370405.pdf. Cited on page 30.

Downey, R. and Fellows, M.R., 1999. *Parameterized Complexity*. Springer, New York. Cited on pages 29, 30.

Dufourd, C., Jančar, P., and Schnoebelen, Ph., 1999. Boundedness of reset P/T nets. In *ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 301–310. Springer. doi:10.1007/3-540-48523-6_27. Cited on page 96.

Dunn, J.M. and Restall, G., 2002. Relevance logic. In Gabbay, D.M. and Guenthner, F., editors, *Handbook of Philosophical Logic*, volume 6, pages 1–128. Kluwer Academic Publishers. http://consequently.org/papers/rle.pdf. Cited on pages 49, 142.

Erdős, P. and Tarski, A., 1943. On families of mutually exclusive sets. *Annals of Mathematics*, 44:315–329. Cited on page 14.

Fairtlough, M.V.H. and Wainer, S.S., 1992. Ordinal complexity of recursive definitions. *Information and Computation*, 99(2):123–153. doi:10.1016/0890-5401(92)90027-D. Cited on pages 120, 121.

Fairtlough, M. and Wainer, S.S., 1998. Hierarchies of provably recursive functions. In Buss, S., editor, *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, chapter III, pages 149–207. Elsevier. doi:10.1016/S0049-237X(98)80018-9. Cited on pages 79, 119, 139.

Fellows, M.R. and Langston, M.A., 1987. Nonconstructive advances in polynomial-time complexity. *Information Processing Letters*, 26(3):157–162. Cited on page 30.

Fellows, M.R. and Langston, M.A., 1988. Nonconstructive tools for proving polynomial-time decidability. *Journal of the ACM*, 35(3):727–739. Cited on page 13.

Figueira, D. and Segoufin, L., 2009. Future-looking logics on data words and trees. In *MFCS 2009*, volume 5734 of *Lecture Notes in Computer Science*, pages 331–343. Springer. doi:10.1007/978-3-642-03816-7_29. Cited on pages 96, 143.

Figueira, D., Figueira, S., Schmitz, S., and Schnoebelen, Ph., 2011. Ackermannian and primitive-recursive bounds with Dickson's Lemma. In *LICS 2011*, pages 269–278. IEEE. doi:10.1109/LICS.2011.39. Cited on pages 79, 140.

Figueira, D., 2012. Alternating register automata on finite words and trees. *Logical Methods in Computer Science*, 8(1):22. doi:10.2168/LMCS-8(1:22)2012. Cited on page 143.

Finkel, A., 1987. A generalization of the procedure of Karp and Miller to well structured transition systems. In *ICALP'87*, volume 267 of *Lecture Notes in Computer Science*, pages 499–508. Springer. doi:10.1007/3-540-18088-5_43. Cited on page 49.

Finkel, A., 1990. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–179. doi:10.1016/0890-5401(90)90009-7. Cited on page 49.

Finkel, A. and Schnoebelen, Ph., 2001. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92. doi:10.1016/S0304-3975(00)00102-X. Cited on page 49.

Finkel, A., McKenzie, P., and Picaronny, C., 2004. A well-structured framework for analysing Petri nets extensions. *Information and Computation*, 195(1–2):1–29. doi:10.1016/j.ic.2004.01.005. Cited on page 50.

Finkel, A. and Goubault-Larrecq, J., 2009. Forward analysis for WSTS, part I: Completions. In *STACS 2009*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 433–444. LZI.

doi:10.4230/LIPIcs.STACS.2009.1844. Cited on page 49.

Finkel, A. and Goubault-Larrecq, J., 2012. Forward analysis for WSTS, part II: Complete WSTS. *Logical Methods in Computer Science*, 8(4:28). doi:10.2168/LMCS-8(3:28)2012. Cited on pages 49, 117.

Forster, T., 2003. Better-quasi-orderings and coinduction. *Theoretical Computer Science*, 309(1–3): 111–123. Cited on page 38.

Friedman, H.M., 1999. Some decision problems of enormous complexity. In *LICS 1999*, pages 2–13. IEEE. doi:10.1109/LICS.1999.782577. Cited on page 135.

Friedman, H.M., 2001. Long finite sequences. *Journal of Combinatorial Theory, Series A*, 95(1):102–144. doi:10.1006/jcta.2000.3154. Cited on page 79.

Geser, A., 1996. A proof of Higman's lemma by open induction. Technical Report MIP-9606, Universität Passau. ftp://ftp.uni-passau.de/pub/local/parallel/papers/higman.ps.Z. Cited on page 29.

Goubault-Larrecq, J., 2009. On a generalization of a result by Valk and Jantzen. Research Report LSV-09-09, Laboratoire Spécification et Vérification, ENS Cachan, France. http://www.lsv.ens-cachan.fr/Publis/publis.php?onlykey=LSV:09:09. Cited on page 117.

Goubault-Larrecq, J., Halfon, S., Karandikar, P., Narayan Kumar, K., and Ph. Schnoebelen, 2016. The ideal approach to computing closed subsets in well-quasi-orderings. In preparation. Cited on page 117.

Goubault-Larrecq, J., 2001. Well-founded recursive relations. In Fribourg, L., editor, *Proceedings of the 15th International Workshop on Computer Science Logic (CSL'01)*, volume 2142 of *Lecture Notes in Computer Science*, pages 484–497, Paris, France. Springer. http://www.lsv.ens-cachan.fr/Publis/PAPERS/PS/Gou-csl2001.ps. Cited on page 26.

Goubault-Larrecq, J., 2010. Noetherian spaces in verification. In Abramsky, S., Meyer auf der Heide, F., and Spirakis, P., editors, *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP'10) – Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 2–21, Bordeaux, France. Springer. doi:10.1007/978-3-642-14162-1_2. Cited on pages 36, 37.

Goubault-Larrecq, J., 2013. *Non-Hausdorff Topology and Domain Theory—Selected Topics in Point-Set Topology*, volume 22 of *New Mathematical Monographs*. Cambridge University Press. http://www.cambridge.org/9781107034136. Cited on page 36.

Grzegorczyk, A., 1953. Some classes of recursive functions. *Rozprawy Matematyczne*, 4. http://matwbn.icm.edu.pl/ksiazki/rm/rm04/rm0401.pdf. Cited on page 79.

Hack, M., 1976. The equality problem for vector addition systems is undecidable. *Theoretical Computer Science*, 2(1):77–95. doi:10.1016/0304-3975(76)90008-6. Cited on pages 140, 141.

Haddad, S., Schmitz, S., and Schnoebelen, Ph., 2012. The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In *LICS 2012*, pages 355–364. IEEE. doi:10.1109/LICS.2012.46. Cited on page 96.

Hague, M., 2014. Senescent ground tree rewriting systems. In *CSL-LICS 2014*, pages 48:1–48:10. ACM Press. doi:10.1145/2603088.2603112. Cited on page 144.

Halpern, J.Y. and Shoham, Y., 1991. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962. doi:10.1145/115234.115351. Cited on page 144.

Higman, G., 1952a. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 2(7):326–336. Cited on pages 7, 49, 53, 79.

Higman, G., 1952b. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(2):326–336. doi:10.1112/plms/s3-2.1.326. Cited on page 49.

Hofman, P. and Totzke, P., 2014. Trace inclusion for one-counter nets revisited. In Ouaknine, J., Potapov, I., and Worrell, J., editors, *RP 2014*, volume 8762 of *Lecture Notes in Computer Science*, pages 151–162. Springer. doi:10.1007/978-3-319-11439-2_12. Cited on page 141.

Hopcroft, J.E. and Pansiot, J.J., 1979. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135–159. Cited on page 50.

Howell, R.R., Rosier, L.E., Huynh, D.T., and Yen, H.C., 1986. Some complexity bounds for problems concerning finite and 2-dimensional vector addition systems with states. *Theoretical Computer Science*, 46:107–140. doi:10.1016/0304-3975(86)90026-5. Cited on page 79.

Hüchting, R., Majumdar, R., and Meyer, R., 2013. A theory of name boundedness. In D'Argenio, P.R. and Melgratti, H.C., editors, *CONCUR 2013 - Concurrency Theory - 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8052 of *Lecture Notes in Computer Science*, pages 182–196. Springer. ISBN 978-3-642-40183-1. doi: 10.1007/978-3-642-40184-8_14. Cited on page 51.

Intrigila, B. and Varricchio, S., 2000. On the generalization of Higman and Kruskal's theorems to regular languages and rational trees. *Acta Informatica*, 36:817–835. Cited on page 43.

Jacquemard, F., Ségoufin, L., and Dimino, J., 2016. $fo^2(<, +1, \sim)$ on data trees, data tree automata and branching vector addition systems. *Logical Methods in Computer Science*, 12(2). Cited on page 50.

Jančar, P., 1999. A note on well quasi-orderings for powersets. *Inf. Proc. Let.*, 72(5-6):155–160. Cited on page 37.

Jančar, P., 1995. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301. doi:10.1016/0304-3975(95)00037-W. Cited on pages 140, 141.

Jančar, P., 2001. Nonprimitive recursive complexity and undecidability for Petri net equivalences. *Theoretical Computer Science*, 256(1–2):23–30. doi:10.1016/S0304-3975(00)00100-6. Cited on pages 49, 140, 141.

Jurdziński, M. and Lazić, R., 2007. Alternation-free modal mu-calculus for data trees. In *LICS 2007*, pages 131–140. IEEE. doi:10.1109/LICS.2007.11. Cited on pages 96, 143.

Jurdziński, M., Lazić, R., and Schmitz, S., 2015. Fixed-dimensional energy games are in pseudo-polynomial time. In *ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 260–272. Springer. doi:10.1007/978-3-662-47666-6_21. Cited on page 141.

Kabil, M. and Pouzet, M., 1992a. Une extension d'un théorème de P. Jullien sur les âges de mots. *RAIRO Theoretical Informatics and Applications*, 26(5):449–482. http://archive.numdam.org/article/ITA_1992__26_5_449_0.pdf. Cited on page 117.

Kabil, M. and Pouzet, M., 1992b. Une extension d'un théorème de P. Julien sur les âges de mots. *Informatique théorique et applications*, 26(5):449–482. Cited on page 15.

Karp, R.M. and Miller, R.E., 1969. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195. doi:10.1016/S0022-0000(69)80011-5. Cited on pages 49, 117.

Kawarabayashi, K.i., Kobayashi, Y., and Reed, B., 2012. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435. Cited on page 13.

Ketonen, J. and Solovay, R., 1981. Rapidly growing Ramsey functions. *Annals of Mathematics*, 113 (2):27–314. doi:10.2307/2006985. Cited on page 79.

Kosaraju, S.R., 1982. Decidability of reachability in vector addition systems. In *STOC'82*, pages 267–281. ACM Press. doi:10.1145/800070.802201. Cited on page 140.

Koymans, R., 1990. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299. doi:10.1007/BF01995674. Cited on page 143.

Kripke, S.A., 1959. The problem of entailment. In *ASL 1959*, volume 24(4) of *Journal of Symbolic Logic*, page 324. http://www.jstor.org/stable/2963903. Abstract. Cited on page 49.

Kruskal, J.B., 1960. Well-quasi-ordering, the tree theorem, and vazsonyi's conjecture. *Transactions of the American Mathematical Society*, 95(2):210–225. Cited on pages 11, 49, 79.

Kruskal, J.B., 1972. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13(3):297–305. doi:10.1016/0097-3165(72)90063-5. Cited on page 49.

Lambert, J.L., 1992. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99(1):79–104. doi:10.1016/0304-3975(92)90173-D. Cited on page 140.

Laver, R., 1976. Well-quasi-orderings and sets of finite sequences. *Mathematical Proceedings of the Cambridge Philosophical Society*, 79(1):1–10. Cited on page 37.

Lazic, R., Newcomb, T.C., Ouaknine, J., Roscoe, A.W., and Worrell, J., 2007. Nets with tokens which carry data. In *Proc. 28th Intl. Conf. Applications and Theory of Petri Nets and Other Models of Concurrency (ICATPN'07)*, pages 301–320. Springer Verlag LNCS 4546. Cited on page 50.

Lazić, R. and Schmitz, S., 2015a. The ideal view on Rackoff's coverability technique. In *RP 2015*, volume 9328 of *Lecture Notes in Computer Science*, pages 1–13. Springer. doi:10.1007/978-3-319-24537-9_8. Cited on page 117.

Lazić, R. and Schmitz, S., 2015b. Non-elementary complexities for branching VASS, MELL, and extensions. *ACM Transactions on Computational Logic*, 16(3:20):1–30. doi:10.1145/2733375. Cited on page 143.

Lazić, R. and Schmitz, S., 2016. The complexity of coverability in ν-Petri nets. In *LICS 2016*. ACM. hal.inria.fr:hal-01265302. To appear. Cited on page 117.

Lazić, R., Ouaknine, J.O., and Worrell, J.B., 2016. Zeno, Hercules and the Hydra: Safety metric temporal logic is Ackermann-complete. *ACM Transactions on Computational Logic*, 17(3). doi:10.1145/2874774. Cited on page 144.

Leroux, J., 2011. Vector addition system reachability problem: a short self-contained proof. In *POPL 2011*, pages 307–316. ACM Press. doi:10.1145/1926385.1926421. Cited on page 140.

Leroux, J. and Schmitz, S., 2015. Demystifying reachability in vector addition systems. In *LICS 2015*, pages 56–67. IEEE. doi:10.1109/LICS.2015.16. Cited on page 140.

Lipton, R.J., 1976. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University. http://www.cs.yale.edu/publications/techreports/tr63.pdf. Cited on pages 140, 141.

Löb, M. and Wainer, S., 1970. Hierarchies of number theoretic functions, I. *Archiv für Mathematische Logik und Grundlagenforschung*, 13:39–51. doi:10.1007/BF01967649. Cited on pages 79, 127, 131, 132, 136.

Lovász, L., 2006. Graph minor theory. *Bulletin of the American Mathematical Society*, 43(1):75–86. doi:10.1090/S0273-0979-05-01088-8. Cited on page 49.

Marché, C., 1996. Normalized rewriting: An alternative to rewriting modulo a set of equations. *J. Symb. Comput.*, 21(3):253–288. doi:10.1006/jsco.1996.0011. Cited on page 36.

Marcone, A., 1994. Foundations of bqo theory. *Transactions of the American Mathematical Society*, 345(2):641–660. Cited on page 37.

Mayr, E.W., 1981. An algorithm for the general Petri net reachability problem. In *STOC'81*, pages 238–246. ACM Press. doi:10.1145/800076.802477. Cited on page 140.

Mayr, E.W. and Meyer, A.R., 1981. The complexity of the finite containment problem for Petri nets. *Journal of the ACM*, 28(3):561–576. doi:10.1145/322261.322271. Cited on pages 49, 140.

Mayr, R., 2000. Undecidable problems in unreliable computations. In *LATIN 2000*, volume 1776 of *Lecture Notes in Computer Science*, pages 377–386. Springer. doi:10.1007/10719839_37. Cited on page 96.

McAloon, K., 1984. Petri nets and large finite sets. *Theoretical Computer Science*, 32(1–2):173–183. doi:10.1016/0304-3975(84)90029-X. Cited on pages 79, 140.

Meyer, A.R. and Ritchie, D.M., 1967. The complexity of loop programs. In *ACM '67*, pages 465–469. doi:10.1145/800196.806014. Cited on page 139.

Milner, E.C., 1985. Basic WQO- and BQO-theory. In Rival, I., editor, *Graphs and Order. The Role of Graphs in the Theory of Ordered Sets and Its Applications*, volume 147 of *NATO ASI Series*, pages 487–502. D. Reidel Publishing. doi:10.1007/978-94-009-5315-4_14. Cited on page 49.

Montanari, A., Puppis, G., and Sala, P., 2010. Maximal decidable fragments of Halpern and Shoham's modal logic of intervals. In *ICALP 2010*, volume 6199 of *Lecture Notes in Computer Science*, pages 345–356. Springer. doi:10.1007/978-3-642-14162-1_29. Cited on page 145.

Müller-Olm, M. and Seidl, H., 2002. Polynomial constants are decidable. In Hermenegildo, M.V. and Puebla, G., editors, *Proc. 9th International Symposium on Static Analysis (SAS'02)*, pages 4–19. Springer-Verlag LNCS 2477. ISBN 3-540-44235-9. Cited on page 37.

Murthy, C.R. and Russell, J.R., 1990. A constructive proof of Higman's lemma. In *LICS'90*, pages 257–267. IEEE Computer Society Press. Cited on page 29.

Nash-Williams, C.S.J.A., 1968. On better-quasi-ordering transfinite sequences. *Proc. Camb. Phil. Soc.*, 64:273–290. Cited on pages 7, 37.

Nash-Williams, C.St.J.A., 1963. On well-quasi-ordering finite trees. *Math. Proc. Cambridge Philos. Soc.*, 59(4):833–835. doi:10.1017/S0305004100003844. Cited on page 49.

Odifreddi, P.G., 1999. *Classical Recursion Theory, vol. II*, volume 143 of *Studies in Logic and the Foundations of Mathematics*. Elsevier. doi:10.1016/S0049-237X(99)80040-8. Cited on pages 79, 119.

Ogawa, M., 2003. A linear time algorithm for monadic querying of indefinite data over linearly ordered domains. *Inf. Comput.*, 186(2):236–259. Cited on pages 27, 29.

Ouaknine, J.O. and Worrell, J.B., 2007. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, 3(1):8. doi:10.2168/LMCS-3(1:8)2007. Cited on page 143.

Padovani, V., 2013. Ticket Entailment is decidable. *Mathematical Structures in Computer Science*, 23 (3):568–607. doi:10.1017/S0960129512000412. Cited on page 49.

Péquignot, Y., 2015. *Better Quasi-Order: Ideals and Spaces—Ainsi de Suite. . . .* PhD thesis, Université Paris Diderot and Université de Lausanne. Cited on page 38.

Pérez, G.A., 2016. The fixed initial credit problem for energy games with partial-observation is ACK-complete. Preprint. http://arxiv.org/abs/1512.04255. Cited on page 142.

Podelski, A. and Rybalchenko, A., 2004. Transition invariants. In *LICS 2004*, pages 32–41. IEEE. doi:10.1109/LICS.2004.1319598. Cited on page 49.

Rackoff, C., 1978a. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231. doi:10.1016/0304-3975(78)90036-1. Cited on pages 49, 140, 141.

Rackoff, C., 1978b. The covering and boundedness problem for vector addition systems. *Theoretical Computer Science*, 6:223–231. Cited on page 50.

Rado, R., 1954. Partial well-ordering of sets of vectors. *Mathematika*, 1:89–95. Cited on page 9.

Rambow, O., 1994. Multiset-valued linear index grammars: Imposing dominance constraints on derivations. In *32nd Annual Meeting of the Association for Computational Linguistics (ACL'94)*, pages 263–270. ACL Press. Cited on page 50.

Richman, F. and Stolzenberg, G., 1990. Well quasi-ordered sets. Technical report, Northeastern University, Boston, MA and Harvard University, Cambridge, MA. Cited on page 29.

Ritchie, R.W., 1963. Classes of predictably computable functions. *Transactions of the American Mathematical Society*, 106(1):139–173. doi:10.1090/S0002-9947-1963-0158822-2. Cited on page 137.

Robertson, N. and Seymour, P.D., 1995. Graph minors. XIII. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110. Cited on page 13.

Robertson, N. and Seymour, P.D., 2004. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357. Cited on page 12.

Rose, H.E., 1984. *Subrecursion: Functions and Hierarchies*, volume 9 of *Oxford Logic Guides*. Clarendon Press. Cited on pages 79, 119.

Schmitz, S. and Schnoebelen, Ph., 2011. Multiply-recursive upper bounds with Higman's Lemma. In *ICALP 2011*, volume 6756 of *Lecture Notes in Computer Science*, pages 441–452. Springer. doi:10.1007/978-3-642-22012-8_35. Cited on page 79.

Schmitz, S., 2010. On the computational complexity of dominance links in grammatical formalisms. In *48th Annual Meeting of the Association for Computational Linguistics (ACL'10)*, pages 514–524, Uppsala, Sweden. ACL Press. Cited on page 50.

Schmitz, S., 2014. Complexity bounds for ordinal-based termination. In *RP 2014*, volume 8762 of *Lecture Notes in Computer Science*, pages 1–19. Springer. doi:10.1007/978-3-319-11439-2_1. Cited on page 79.

Schmitz, S., 2016a. Implicational relevance logic is 2-ExpTime-complete. *Journal of Symbolic Logic*, 81(2):641–661. Cited on page 51.

Schmitz, S., 2016b. Implicational relevance logic is 2-ExpTime-complete. *Journal of Symbolic Logic*, 81(2):641–661. doi:10.1017/jsl.2015.7. Cited on page 49.

Schmitz, S., 2016c. Complexity hierarchies beyond Elementary. *ACM Transactions on Computation Theory*, 8(1):1–36. doi:10.1145/2858784. Cited on pages 135, 140.

Schnoebelen, Ph., 2002. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261. doi:10.1016/S0020-0190(01)00337-4. Cited on pages 96, 142.

Schnoebelen, Ph., 2010a. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In *MFCS 2010*, volume 6281 of *Lecture Notes in Computer Science*, pages 616–628. Springer. doi:10.1007/978-3-642-15155-2_54. Cited on pages 96, 142.

Schnoebelen, Ph., 2010b. Lossy counter machines decidability cheat sheet. In *RP 2010*, volume 6227 of *Lecture Notes in Computer Science*, pages 51–75. Springer. doi:10.1007/978-3-642-15349-5_4. Cited on page 96.

Seidl, H., 1990. Deciding equivalence of finite tree automata. *SIAM Journal on Computing*, 19(3): 424–437. doi:10.1137/0219027. Cited on page 135.

Stockmeyer, L.J. and Meyer, A.R., 1973. Word problems requiring exponential time. In *STOC '73*, pages 1–9. ACM Press. doi:10.1145/800125.804029. Cited on page 139.

Tan, T., 2010. On pebble automata for data languages with decidable emptiness problem. *Journal of Computer and System Sciences*, 76(8):778–791. doi:10.1016/j.jcss.2010.03.004. Cited on pages 96, 143.

Turing, A., 1949. Checking a large routine. In *Report of a Conference on High Speed Automatic Calculating Machines*. Republished in *The early British computer conferences*, pages 70–72, MIT Press, 1989. Cited on page 49.

Tzevelekos, N. and Grigore, R., 2013. History-register automata. In Pfenning, F., editor, *FoSSaCS 2013*, volume 7794 of *Lecture Notes in Computer Science*, pages 273–288. doi:10.1007/978-3-642-37075-5_2. Cited on page 143.

Urquhart, A., 1984. The undecidability of entailment and relevant implication. *Journal of Symbolic Logic*, 49(4):1059–1073. doi:10.2307/2274261. Cited on pages 49, 142.

Urquhart, A., 1999. The complexity of decision procedures in relevance logic II. *Journal of Symbolic Logic*, 64(4):1774–1802. doi:10.2307/2586811. Cited on pages 49, 96, 142, 143.

Valk, R. and Jantzen, M., 1985. The residue of vector sets with applications to decidability problems in Petri nets. *Acta Informatica*, 21(6):643–674. doi:10.1007/BF00289715. Cited on page 117.

van der Meyden, R., 1997. The complexity of querying indefinite data about linearly ordered domains. *J. Comp. Sys. Sci.*, 54(1):113–135. Cited on pages 27, 28.

Verma, K.N. and Goubault-Larrecq, J., 2005. Karp-Miller trees for a branching extension of VASS. *Discrete Mathematics & Theoretical Computer Science*, 7(1):217–230. http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/VGL-dmtcs05.pdf. Cited on page 50.

Vytiniotis, D., Coquand, T., and Wahlstedt, D., 2012. Stop when you are almost-full, adventures in constructive termination. In Beringer, L. and Felty, A., editors, *Intl. Conf. on Interactive Theorem Proving (ITP'12)*, pages 250–265, Princeton, NJ, USA. Springer Verlag LNCS 7406. Cited on page 43.

Wainer, S.S., 1970. A classification of the ordinal recursive functions. *Archiv für Mathematische Logik und Grundlagenforschung*, 13(3):136–153. doi:10.1007/BF01973619. Cited on pages 137, 139.

Wainer, S.S., 1972. Ordinal recursion, and a refinement of the extended Grzegorczyk hierarchy. *Journal of Symbolic Logic*, 37(2):281–292. doi:10.2307/2272973. Cited on page 139.

Weiermann, A., 1994. Complexity bounds for some finite forms of Kruskal's Theorem. *Journal of Symbolic Computation*, 18(5):463–488. doi:10.1006/jsco.1994.1059. Cited on page 79.

Wies, T., Zufferey, D., and Henzinger, T.A., 2010. Forward analysis of depth-bounded processes. In Ong, L., editor, *Proc. 13th Intl. Conf. Foundations of Software Science and Computational Structures (FoSSaCS'10)*, pages 94–108. Springer Verlag LNCS 6014. Cited on page 51.

Wikipedia, 2017. Ramsey's theorem. https://en.wikipedia.org/wiki/Ramsey's_theorem. Read Aug.

28. Cited on page 3.

Wolk, E., 1967. Partially well ordered sets and partial ordinals. *Fundamenta Mathematicae*, 60(2): 175–186. http://eudml.org/doc/213955. Cited on page 40.

Zetzsche, G., 2015. An approach to computing downward closures. In *ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 440–451. Springer. doi:10.1007/978-3-662-47666-6_35.

INDEX