Algorithmic Complexity of Well-Quasi-Orders

Sylvain Schmitz

université
PARIS-SACLAY

école
normale
supérieure
paris—saclay

Habilitation à diriger des recherches

Algorithmic Complexity of Well-Quasi-Orders

Habilitation à diriger des recherches
préparée à l'École Normale Supérieure Paris-Saclay
au sein du Laboratoire Spécification & Vérification

Présentée et soutenue à Cachan, le 27 novembre 2017, par

Sylvain Schmitz

Composition du jury :

Mikołaj Bojańczyk Professeur, Universytet Warszawski	Rapporteur
Javier Esparza Professeur, Technische Universität München	Rapporteur
Alain Finkel Professeur, École Normale Supérieure Paris-Saclay	Examinateur
Géraud Sénizergues Professeur, Université de Bordeaux	Rapporteur
Andreas Weiermann Professeur, Universiteit Gent	Examinateur
James B. Worrell Professeur, University of Oxford	Examinateur

Algorithmic Complexity of Well-Quasi-Orders

Habilitation Thesis

Sylvain Schmitz

Acknowledgements [*]

I am indebted to a very large number of people who supported me on institutional, scientific, and personal levels while carrying this work.

My first thanks go to all the people I have worked along with at the *Laboratoire Spécification et Vérification* in Cachan over the nine years I have spent there since 2008: to the administrative and technical staff for their kindness and availability, to my current students Simon Halfon and Anthony Lick and my former intern and visiting students for thought-provoking discussions, and to all my colleagues for creating such a supportive and stimulating environment—it has been an incredibly formative experience, I have learned so much from all of you. Special thanks go to Alain Finkel, Jean Goubault-Larrecq, and Philippe Schnoebelen, who introduced me to the theory of well-quasi-orders when I arrived in Cachan.

Let me thank more generally all my co-authors over the years: it has been a privilege to work by your side. My lasting collaborations with Ranko Lazić and Philippe Schnoebelen have been especially influential: little of the work described in this habilitation would have existed without you.

I also thank the members of the jury for their excellent comments and corrections on the manuscript, and for their patience and goodwill during the entire process.

I have also benefited from the fantastic opportunity to teach at École Normale Supérieure Paris-Saclay all over these nine years; the demands of tutoring and lecturing to such an exacting audience have had a deep influence. I thank my students and everyone I shared a course with, and in particular Paul Gastin for his guidance as head of the computer science department when I started on the job, at what was then called ENS Cachan.

To end on a personal note, I give my heartful thanks to Marie Antczak, Claire David, Laure Daviaud, Mahsa Shirmohammadi, and everyone else who joined in the nudging and encouraged me to complete this habilitation thesis.

Thank you!

[*] I acknowledge the institutional and financial support of ENS Paris-Saclay, CNRS, Inria, ANR, the Leverhulme Trust, the University of Warwick, and the University of Warsaw.

Contents

CHAPTER 1

Introduction

1.1. General Context

The main motivations for this thesis are rooted in formal verification, and in particular in model-checking techniques. This might come as a surprise, because the title seems centred on well-quasi-orders, which are a rather abstract concept, whose invention can be tracked back to the early 1910s and whose theory was developed by mathematicians like Higman [1952], Kruskal [1972], de Jongh and Parikh [1977], and many others. Model-checking on the other hand, while certainly grounded on sound mathematical concepts [e.g. Vardi, 2009], is an eminently practical field of computer science with a widespread adoption in the hardware industry. At its core, it consists of a fully automated, algorithmic check that a model of a system—like a concrete sequential circuit—satisfies its specification—like a *safety* check that no bad configuration can ever be reached in the model.

1.1.1. Verification of Infinite-state Systems. Once the general underpinnings of model-checking for *finite-state* systems were well-understood, the focus of the formal verification community turned in the 1990s to *infinite-state* systems. Indeed, many critical systems worth the time and effort required by formal verification are infinite, due to unbounded variables: integer counters, number of active threads in concurrent settings, real-time clocks, stacks, nonces, communication channels, etc. In some specific instances, these infinite-state systems are 'essentially finite-state' and can be faithfully abstracted into finite ones by *finite partitioning*, i.e. by quotienting them using a bisimulation equivalence of finite index—this method was notably behind the success of timed automata [Alur and Dill, 1994]. But at that time this research area was lacking common methods and general principles to cope with cases where finite partitioning failed.

1.1.2. A Game Changer: Well-Structured Systems. This situation was remedied by Abdulla, Čerāns, Jonsson, and Tsay [2000] and Finkel and Schnoebelen [2001], who popularised the notion of *well-structured systems*. Their approach can be understood as a generalisation of the finite partitioning method, using instead simulation quasi-orders. The finiteness condition on the equivalence's index was accordingly replaced by a much more flexible condition: this quasi-order should be a *well-quasi-order* (a wqo).

The two landmark papers by Abdulla et al. and Finkel and Schnoebelen identified two *generic* algorithmic constructions, backward coverability and finite reachability trees, which terminate in well-structured systems and allow to check properties like safety (through the *coverability* problem), inevitability, termination, boundedness, simulation relations, etc. This provided a unifying explanation of several existing decidability results, and guidelines for discovering similar results in future classes of systems.

Already in 2000, this simple abstract framework had been shown to capture varied concrete formalisms, including extensions of vector addition systems, communicating finite-state machines, process algebra, or timed systems. Twenty years later, this catalogue of applications still keeps growing, with unforeseen far-reaching theoretical applications, like the decidability of logics—including metric temporal logics [Ouaknine and Worrell, 2007], data logics [Figueira, 2012], interval temporal logics [Bresolin et al., 2012], and substructural logics [J1; J4]. Thus the title 'Well-Structured Transition Systems Everywhere!' of Finkel and Schnoebelen's article has turned out to be a self-fulfilling prophecy.

1.1.3. Tools and Practical Applications. Well-structured systems are also at the origin of numerous practical applications, which target among others the automatic verification of parametric systems and of multi-threaded programs. Infinite-state models like vector addition systems and their extensions are indeed necessary in order to analyse systems where correctness should hold for any number of components, or where dynamic creation of processes or execution threads can occur. These tools typically implement highly optimised versions of the backward coverability principle, sometimes alongside forward pruning of the state space, and are able to cope with practical instances extracted from multi-threaded programs, cache coherence protocols, communication protocols, memory managers, etc. [Blondin et al., 2017b; Delzanno et al., 2001; Esparza et al., 2014; Geffroy et al., 2016, 2017; Geeraerts et al., 2005; Kaiser et al., 2014, 2017].

1.1.4. The Complexity Question. Although the theory of well-structured systems provides generic algorithms for numerous verification problems, it might seem rather unclear, what the computational cost of running these generic algorithms might be. Indeed, the underlying well-quasi-orders ensure that the algorithm will eventually terminate, but the usual proofs of this fact are not constructive and do not provide any information as to when that will happen.

What was however known thanks to Schnoebelen [2002] was that the worst-case complexity could be considerable. Indeed, Schnoebelen showed that checking safety in two of the main well-structured formalisms, namely lossy counter systems and lossy channel systems, is not primitive-recursive. Thus the worst-case complexity of checking even small systems exceeds the known estimates for physical quantities like the number of atoms in the observable universe or the number of nanoseconds since the Big Bang, and these problems might be better labelled as 'not undecidable.'

We could stop there on the question of the complexity of algorithms on well-structured systems. Such lower bounds indicate that, as far as practical verification goes, the problems are not that different from undecidable ones. That is, we should not be deterred because those are worst-case bounds that say little about the difficulty of solving practical instances, which presumably do not attempt to implement complex computations of Turing or Minsky machines.

I can easily argue here that the point of complexity theory is very rarely to estimate the running times observed on implementations over realistic inputs.[1] The interest rather lies in finely understanding the computational problem at hand, in

[1]The exception being average-case complexity analysis for a suitable probability distribution on the inputs, which is fiendishly difficult even for simple algorithms, not to mention how hard it is to invent a 'realistic' probability distribution.

having at least a crude notion of optimality for it, in identifying its main sources of complexity, in comparing the complexity of its different algorithms.

Additionally, in the context of well-structured systems where there are many different families of systems using various wqos, computational complexity also serves as a proxy for expressiveness, where algorithmic lower bounds allow to prove that some classes of models cannot be efficiently encoded into others. From this expressiveness perspective, a highly expressive but nevertheless not undecidable formalism also indicates an interesting modelling trade-off.

1.2. Contributions

In this thesis, I present a comprehensive assortment of results answering the 'complexity question' for algorithms that rely on well-quasi-orders for their termination, like the briefly mentioned backward coverability algorithm in well-structured systems. I cover both upper bounds and lower bounds techniques, the definition of suitable complexity classes, and provide applications in verification (prominently for well-structured systems). Because wqos are in such wide use, this topic is of relevance to a broad community with interests in complexity theory and decision procedures for logical theories. As a testimony to this wide applicability, I refer the reader to [J3, Section 6], which contains a surprisingly large catalogue of decision problems shown complete for the complexity classes described herein.

After a brief overview of the algorithmic applications of well-quasi-orders in Chapter 2, I present some of my contributions to this recent research direction in Chapters 3 to 5, which I summarise next.

1.2.1. Techniques for Upper Bounds. As shown in Chapter 2, many algorithms rely on well-quasi-orderings for their proof of termination. Although the classical proofs of Dickson's Lemma, Higman's Lemma, and other wqos, are of non-constructive 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. The resulting combinatorial statements are dubbed *length function theorems*, as they allow to bound the length of so-called 'controlled bad sequences' over the wqo at hand.

1.2.1.1. *Length Function Theorems.* I present in Chapter 3 how one can derive complexity upper bounds for algorithms relying for their termination on ordinals below ε_0 [I3] or Dickson's Lemma over tuples of natural numbers [C19]. The techniques are however quite generic and apply to more complex wqos [Rosa-Velardo, 2017; C18].

Contribution (I3; C18; C19). Length function theorems for several well-quasi-orders: Dickson's Lemma, Higman's Lemma, ordinals below ε_0, and their combinations.

This topic had already been explored in research communities centred on proof theory and rewriting theory [Cichoń and Tahhan Bittar, 1998; Ketonen and Solovay, 1981; McAloon, 1984; Weiermann, 1994], and my results with Figueira, Figueira, and Schnoebelen rely on the same mathematical tools—broadly speaking, maximal order types and subrecursive functions [e.g. Buchholz et al., 1994]—, but provide refined bounds and more general statements on the wqos we consider. Importantly, we showed how to apply these techniques to the algorithms

employed in the literature on well-structured systems, making them accessible to the infinite-state verification community and fostering their adoption.

1.2.1.2. *Monotone Descending Sequences.* Length function theorems on controlled bad sequences are however not able to account for some results. A case in point is the coverability problem in vector addition systems, where length function theorems only provide Ackermannian upper bounds, although coverability witnesses were shown to be of double exponential length by Rackoff [1978]. In joint work with Lazić [W1], we showed how a dual view of the usual backward coverability algorithm, that builds a descending sequence of downwards-closed sets instead of an ascending sequence of upwards-closed sets, exhibits a *monotonicity* invariant yielding the desired double exponential bound. The same approach has since been used to obtain tight bounds in ν-Petri nets [C5] and polynomial automata [Benedikt et al., 2017]. I touch this subject in Section 6.3.2.

Contribution (C5; W1). Length function theorems for monotone descending sequences of downwards-closed sets over tuples of natural numbers and finite multisets of such tuples.

1.2.2. Techniques for Lower Bounds. The complexity upper bounds provided by length function theorems are typically not primitive-recursive, and one is bound to wonder whether they are useful at all, i.e. whether there exist natural decision problems that require Ackermannian resources for their resolution. As shown in pioneering works by Mayr and Meyer [1981], Urquhart [1999], and Schnoebelen [2002], such gigantic complexities are actually tight for problems on counter machines.

1.2.2.1. *Hardy Computations.* I illustrate these lower bound techniques in Section 4.2 in the case of coverability in lossy counter machines. While the general pattern of these lower bound proofs relies on *weak computer* implementations of fast-growing functions and their inverses as in [Chambart and Schnoebelen, 2008; Mayr and Meyer, 1981; Schnoebelen, 2002; 2010a; Urquhart, 1999], my contribution lies in the implementation of *Hardy computations* using robust encodings of ordinals. This allows to considerably simplify the proofs, which quickly become horrendously involved in more expressive formalisms like priority channel systems [J5], lossy channel systems [C14], ordered data nets [C16], nested counter systems [Decker and Thoma, 2016], or unordered data nets [Rosa-Velardo, 2017].

Contribution (Theorem 4.9; J5; C14; C16; L1). Hardy-like computations for proving complexity lower bounds in well-structured systems.

The particular lower bound statement in Theorem 4.9 of this thesis is an original contribution: it slightly improves over the known lower bounds for the coverability problem in lossy counter machines with a fixed number of counters [Schnoebelen, 2010a], and thus in other extensions of vector addition systems where these bounds apply: reset Petri nets, transfer Petri nets, broadcast protocols, ...

1.2.2.2. *Counter Objects.* The approach through weak computers fails to provide tight lower bounds in some cases, including prominently vector addition systems. For the sake of completeness, I briefly discuss in Section 4.3.1 another approach introduced by Lipton [1976] to show the EXPSPACE-hardness of coverability in vector addition systems. This has been instrumental in obtaining lower bounds for several classes of well-structured systems [J4; C5; C9; C21].

1.2.3. Complexity Classes. Complexity classes, along with the associated notions of reductions and completeness, allow us to classify and compare computational problems.

1.2.3.1. *Non-Elementary Problems.* The complexity literature does not provide any intermediate stops where well-known non-elementary problems—like WS1S, the satisfiability problem for the weak monadic theory of one successor [Meyer, 1975]—would fit adequately. Indeed, WS1S is not in ELEMENTARY, but the next class is PRIMITIVE-RECURSIVE, which is far too big: WS1S is not *hard* for PRIMITIVE-RECURSIVE under any reasonable notion of reduction. In other words, we seem to be missing a 'TOWER' complexity class, which ought to sit somewhere between ELEMENTARY and PRIMITIVE-RECURSIVE. Going higher, we find a similar uncharted area between PRIMITIVE-RECURSIVE and RECURSIVE, where all the decision problems considered in this thesis should fit.[2]

1.2.3.2. *Fast-Growing Complexity.* In Section 4.1, I present a proposal for complexity classes above ELEMENTARY, which has already proven suitable for many decision problems [J3, Section 6].

Contribution (J3). The definition of fast-growing complexity classes $(\mathbf{F}_\alpha)_\alpha$ suitable for non-elementary complexity.

The classes $\mathbf{F}_\alpha$ are related to the hierarchy $(\mathscr{E}^k)_k$ of Grzegorczyk [1953] and the extended Grzegorczyk hierarchy $(\mathscr{F}_\alpha)_\alpha$ of Löb and Wainer [1970], which have been used in most older complexity statements involving non-elementary bounds in the literature. The $(\mathscr{F}_\alpha)_\alpha$ classes are well-suited for characterising various classes of functions, for instance computed by forms of 'for' programs [Meyer and Ritchie, 1967] or terminating 'while' programs [Fairtlough and Wainer, 1992], or provably total in fragments of Peano arithmetic [e.g. Fairtlough and Wainer, 1998], and they characterise some important milestones like the classes of elementary or primitive-recursive decision problems. They are however too large to classify decision problems arising from the use of wqos in algorithms: they do not lead to completeness statements—in fact, one can show that there are *no* 'ELEMENTARY-complete' nor 'PRIMITIVE-RECURSIVE-complete' problems.

1.2.4. A Challenge: the Complexity of VASS Reachability. *Vector addition systems* (VASS), or equivalently Petri nets, find a wide range of applications in the modelling of concurrent, chemical, biological, or business processes. Their algorithmics, and in particular the decidability of their *reachability problem*, is a central component to many decidability results spanning from the verification of asynchronous programs [Ganty and Majumdar, 2012] to the decidability of data logics [Bojańczyk et al., 2011; Demri et al., 2016].

Considered as one of the great achievements of theoretical computer science, the original 1981 decidability proof of Mayr is the culmination of more than a decade of research into the topic, and builds notably on an incomplete proof by Sacerdote and Tenney [1977]. The proof was simplified a year later by Kosaraju [1982]. In spite of this success, as put by Lambert in his 1992 article presenting his own simplification of the proof,

[2] Somewhat oddly, the complexities above RECURSIVE are better explored and can rely on the arithmetical and analytical hierarchies.

> 'the complexity of the two proofs (especially in [Mayr, 1981])
> wrapped the result in mystery,'

and no complexity upper bound was known for the general VASS reachability problem, while the best known lower bound is EXPSPACE-hardness [Lipton, 1976]. The only known tight bounds pertain to the case of 2-dimensional VASS, which were recently shown to have a PSPACE-complete reachability problem when using binary encodings [Blondin et al., 2015], and even an NL-complete one with unary encodings [Englert et al., 2016].

1.2.4.1. *Upper Bound.* I present in Chapter 5 the first known complexity upper bounds for the reachability problem, by analysing the algorithm of Mayr, Kosaraju, and Lambert using the length function theorems presented in Chapter 3.

Contribution (Theorem 5.1; C8). The first complexity upper bounds for reachability in vector addition systems.

More precisely, the result in Theorem 5.1 is a small improvement over the 'cubic Ackermann' upper bound obtained with Leroux in [C8]: I show a 'quadratic Ackermann' upper bound, in $\mathbf{F}_{\omega^2}$ in terms of the complexity classes of Chapter 4. This bound is not expected to be tight. However, as observed e.g. by Müller [1985] the algorithms of Mayr, Kosaraju, and Lambert are not primitive-recursive. Thus the complexity gap for these particular algorithms, between $\mathbf{F}_{\omega}$ and $\mathbf{F}_{\omega^2}$, is not that wide.

1.2.4.2. *Ideal Decompositions.* At the heart of these decidability proofs lies a *decomposition technique*, which is dubbed the Kosaraju-Lambert-Mayr-Sacerdote-Tenney (KLMST) decomposition after its inventors. In a nutshell, the KLMST decomposition defines both a structure and a condition for this structure to represent in some way the set of all runs witnessing reachability. The algorithms advanced by Mayr, Kosaraju, and Lambert compute this decomposition by successive refinements of the structure until the condition is fulfilled.

The 'mystery' that shrouds the KLMST decomposition algorithm does not relate to how these structures and conditions are used in the algorithms themselves, nor to the fact that they indeed yield the decidability of VASS reachability. Rather, the issue is to explain how these structures and conditions can be derived in a principled manner. This is what Leroux and I attempt to address in [C8]; I give a quick overview of this result in Section 6.4.2. Using a well-quasi-ordering on VASS runs studied by Jančar [1990] and Leroux [2011], we showed a Decomposition Theorem (Theorem 6.3): the KLMST algorithm computes the *ideal decomposition* of the set of runs, i.e. a decomposition into irreducible downward-closed sets (see Section 6.3.1). The effective representation of those ideals through finite structures turns out to match exactly the structures and conditions expressed by Lambert [1992].

Contribution (C8). Explain the KLMST decomposition algorithm in terms of ideal decompositions of a well-quasi-order.

This provides a full formal framework in which the reachability problem in various VASS extensions might be cast, offering some hope to see progress on those open issues.

1.2.5. Other Contributions. Although it is the main focus of this document, my work does not revolve exclusively around the complexity of wqo algorithms. I would like to mention here a few significant contributions that could not be presented in the remainder of this thesis.

1.2.5.1. *Substructural Logics.* One of the applications of counter systems is to provide algorithmic techniques for other formalisms. For instance, strong connections can be drawn between branching and alternating extensions of vector addition systems on the one hand, and proof systems for substructural logics like linear logic or relevance logic on the other hand. By reducing logic problems to reachability problems in such extensions, I have been able to solve two long-standing open problems.

Contribution (J1). The implicational implicational fragment of relevance logic is 2-EXP-complete.

This fragment had been proven decidable by Kripke in 1959, and no progress had been made since Urquhart's 1990 EXPSPACE lower bound and 1999 ACKERMANN upper bound. The problem is also equivalent to simple type inhabitation in the λI-calculus.

Contribution (J4). Propositional affine linear logic is TOWER-complete.

This logic had been shown decidable by Kopylov in 1995, with no known upper bound. This work with Lazić further provides the best known lower bound for multiplicative exponential linear logic (MELL), whose decidability is a major open problem since the 1990s [Lincoln et al., 1992]—see Section 6.4.

1.2.5.2. *Energy Games.* Alternating vector addition systems also capture a class of 'resource-conscious' games called *multi-dimensional energy parity games*, employed for the synthesis of reactive systems. The motivation here is to generate automatically a correct system against some functional requirements, that furthermore guarantees that several discrete resources (e.g., available funds, items in stock, units of fuel or time, ...) will never get depleted.

Contribution (C1; C7; C9). Multi-dimensional energy parity games are 2-EXP-complete.

This series of articles, first with Courtois, and later in collaborations with Jurdziński, Lazić, and Colcombet, also shows the problem to be in pseudo-polynomial time in fixed dimension. Our results are based on geometric intuitions, which we formalise by introducing *perfect half-space games*. They answer open problems for multi-dimensional energy games [Velner et al., 2015], extended games on vector addition systems [Brázdil et al., 2010], simulation games in basic parallel processes [Lasota, 2009], multi-agent resource logics [Alechina et al., 2016], an affine variant of $(\oplus, !)$-Horn linear logic [Kanovich, 1995], and single-sided μ-calculus model-checking on vector addition systems [Abdulla et al., 2013].

CHAPTER 2

Well-Quasi-Orders & Applications

A *quasi order* (qo) $\langle A, \leq \rangle$ consists of a support set A along with a transitive reflexive relation $\leq \subseteq A \times A$. We call a finite or infinite sequence $x_0, x_1, x_2, \ldots$ over A *good* if there exist two indices $i < j$ such that $x_i \leq x_j$, and *bad* otherwise. A *well-quasi-order* (wqo) is a qo $\langle A, \leq \rangle$ such that any infinite sequence $x_0, x_1, x_2, \ldots$ of elements over A is good. Equivalently, any bad sequence over A is finite.

Thus well-quasi-orders are a means to proving finiteness statements. A typical application is program termination, which can be shown by mapping any execution sequence of a program to a bad sequence of longer or equal length: as bad sequences are finite, so are executions, and this shows that the program terminates (see Section 2.2). A second application is found in *well-structured transition systems* (see Section 2.3), where configurations are ordered with a wqo compatible with the transitions: a large portfolio of algorithms relying on wqos for termination is available on these systems [Abdulla et al., 2000; Finkel and Schnoebelen, 2001], with applications in verification, logic, distributed computing, etc.

The material in this chapter is mostly standard. The presentation is based chiefly on the lecture notes [L1] and the invited papers [I4; I3].

Contents

2.1. Basic Definitions

The *upward-closure* $\uparrow B$ of some $B \subseteq A$ in a qo $\langle A, \leq \rangle$ is defined as $\uparrow B \stackrel{\text{def}}{=} \{x \in A \mid \exists y \in B,\, x \geq y\}$. We write $\uparrow x$ instead of $\uparrow\{x\}$ for singletons. When $B = \uparrow B$, we say that B is *upwards-closed*. The *downward-closure* $\downarrow B$ of B and the notion of *downwards-closed* sets are defined symmetrically.

2.1.1. Well-Quasi-Orders. The definition of a wqo as enforcing finite bad sequences is just one among many equivalent ones [Kruskal, 1972]. Notably, $\langle A, \leq \rangle$ is a wqo if and only if

(1) $\leq$ is *well-founded*, i.e. there does not exist any infinite decreasing sequence $x_0 > x_1 > x_2 > \cdots$ of elements in A, where $< \stackrel{\text{def}}{=} \leq \setminus \geq$, and
(2) $\leq$ satisfies the *finite antichain condition*, i.e. there are no infinite sets of mutually $\leq$-incomparable elements in A.

2.1.1.1. *Ascending Chain Condition.* Another equivalent characterisation is that a qo $\langle A, \leq \rangle$ is a wqo if and only if any increasing sequence $U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots$ of upwards-closed subsets of A eventually stabilises, i.e., $\bigcup_{i \in \mathbb{N}} U_i = U_k = U_{k+1} = U_{k+2} = \cdots$ for some k. Indeed, for the 'only if' direction, in a strictly increasing sequence $U_0 \subsetneq U_1 \subsetneq U_2 \subsetneq \cdots$ of upwards-closed subsets of A, we can extract at each step an element $x_i \in U_{i+1} \setminus U_i$. If $i < j$, then $x_i \leq x_j$ would imply $x_j \in U_{i+1}$ since U_{i+1} is upwards-closed, but this contradicts $x_j \notin U_j$. Hence the sequence of elements $x_0, x_1, x_2, \ldots$ is bad and the strictly increasing sequence $U_0 \subsetneq U_1 \subsetneq U_2 \subsetneq \cdots$ is finite. Note that we can assume each x_i to be a minimal element of U_{i+1} in the previous argument: if every minimal element of U_{i+1} were in U_i, then $U_i \supseteq U_{i+1}$.

2.1.1.2. *Finite Basis Property.* Upwards- and downwards-closed sets of wqos are important algorithmic tools: they are subsets of A that can be finitely represented and handled. The simplest generic representation is by minimal elements. Indeed, the finite antichain condition ensures that, in any subset of A, there are finitely many minimal elements up to the equivalence $\equiv \stackrel{\text{def}}{=} \leq \cap \geq$. Thus any upwards-closed $U \subseteq A$ can be written as the upward-closure of finitely many elements, i.e. $U = \uparrow\{x_1, \ldots, x_n\}$ for some $x_1, \ldots, x_n \in A$. One can see how, using this representation, the comparison of possibly infinite (but upwards-closed) sets can be reduced to finitely many comparisons of elements.

The complement of a downwards-closed set D is upwards-closed. Hence downwards-closed subsets of a wqo can be characterised by so-called *excluded minors*. That is, every downwards-closed D can be associated with a finite set $\{x_1, \ldots, x_n\}$ such that $x \in D$ if and only if $x_1 \not\leq x \wedge \cdots \wedge x_n \not\leq x$. Here the x_is are the excluded minors and D is 'everything that does not have one of them as a minor'. We will see another representation of downwards-closed sets in Section 6.3.

2.1.1.3. *Well-(Partial)-Orders.* A wqo where $\leq$ is antisymmetric is called a *well-partial-order* (wpo). Note that quotienting a wqo by the equivalence $\equiv \stackrel{\text{def}}{=} \leq \cap \geq$, i.e. equating elements x and y whenever $x \leq y$ and $y \leq x$, yields a wpo.

A wpo $\langle A, \leq \rangle$ where $\leq$ is linear (aka total), is a *well-order* (wo). Because a wo has antichains of cardinal at most 1, this coincides with the usual definition as a well-founded linear order. Finally, any *linearisation* of a wpo $\langle A, \leq \rangle$, i.e. any

linear order $\preceq \supseteq \leq$, defines a wo $\langle A, \preceq \rangle$. One can think of the linearisation process as one of 'orienting' pairs of incomparable elements; such a linearisation always exists thanks to the order-extension principle. Note that, over a linear order, a bad sequence $x_0, x_1, \ldots$ is actually a decreasing sequence $x_0 > x_1 > \cdots$.

2.1.2. Examples. For a basic example, consider any finite set Q along with the equality relation, which is a wqo $\langle Q, = \rangle$ by the pigeonhole principle. As explained above, any wo is a wqo, which provides us with another basic example: the set of natural numbers along with its natural ordering $\langle \mathbb{N}, \leq \rangle$.

Many more examples can be constructed using algebraic operations: for instance, if $\langle A, \leq_A \rangle$ and $\langle B, \leq_B \rangle$ are wqos, then so are:

- their disjoint sum $\langle A \sqcup B, \leq_\sqcup \rangle$ with the *sum ordering*: $A \sqcup B \stackrel{\text{def}}{=} \{(x, 0) \mid x \in A\} \cup \{(y, 1) \mid y \in B\}$ and $(z, i) \leq_\sqcup (z', j)$ if and only if $i = j$ and $z \leq_C z'$ where $C \stackrel{\text{def}}{=} A$ if $i = 0$ and $C \stackrel{\text{def}}{=} B$ otherwise,
- their Cartesian product $\langle A \times B, \leq_\times \rangle$ with the *product ordering*: $(x, y) \leq_\times (x', y')$ if and only if $x \leq_A x'$ and $y \leq_B y'$; in the case of $\langle \mathbb{N}^d, \leq_\times \rangle$ this result is also known as Dickson's Lemma [Dickson, 1913],
- $\langle A^*, \leq_* \rangle$, the set of finite sequences over A along with *subword embedding*: $x_1 \cdots x_m \leq_* y_1 \cdots y_n$ if and only if there exists an increasing $f \colon \{1, \ldots, m\} \to \{1, \ldots, n\}$ such that $x_i \leq_A y_{f(i)}$ for all $1 \leq i \leq m$; this result is better known as Higman's Lemma [Higman, 1952],
- the set of finite trees labelled by A with the homeomorphic embedding, aka Kruskal's Tree Theorem [Kruskal, 1960; Nash-Williams, 1963], and
- the set of finite graphs labelled by A with the minor ordering, aka the Graph Minor Theorem [Robertson and Seymour, 2010].

Turning to well orders, an operation that preserves wos is the *lexicographic product* $\langle A \times B, \leq_{\text{lex}} \rangle$ where $(x, y) \leq_{\text{lex}} (x', y')$ if and only if $x <_A x'$, or $x = x'$ and $y \leq_B y'$. This is typically employed in d-tuples of natural numbers ordered lexicographically $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$: observe that this is a linearisation of $\langle \mathbb{N}^d, \leq_\times \rangle$.

Another classical well order employed in termination proofs is the *multiset* order $\langle \mathbb{M}(A), \leq_{\text{m}} \rangle$ of Dershowitz and Manna [1979]. There, $\mathbb{M}(A)$ denotes the set of finite multisets over the wo $\langle A, \leq \rangle$, i.e. of functions $m \colon A \to \mathbb{N}$ with finitely many x in A such that $m(x) > 0$, and $m \leq_{\text{m}} m'$ if and only if for all x in A, if $m(x) > m'(x)$, then there exists $y >_A x$ such that $m(y) < m'(y)$ [see also Jouannaud and Lescanne, 1982].

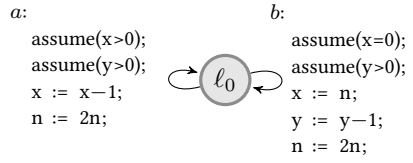
2.2. Application: Program Termination

A first application of well-quasi-orders and well-orders is to prove program termination. The basic principle is to associate a decreasing sequence (over a wo) or a bad sequence (over a wqo) to any sequence of execution steps of the program, thereby showing that the execution must eventually terminate. This is hardly a new idea: using a ranking function into a wo was already put forth by Turing in 1949 (see Section 2.2.1). From there, it is natural to generalise to quasi-ranking functions into wqos [I3], which provide a different viewpoint on termination proofs by disjunctive termination arguments promoted by Podelski and Rybalchenko [2004] (see Section 2.2.2).

We illustrate the main ideas in this section using a very simple program, given in pseudo-code in Figure 2.1a. Formally, we see the operational semantics of a

```
ℓ0: while x >= 0 and y > 0 do
      if x > 0 then
a:      x := x−1; n := 2n;
      else
b:      x := n; y := y−1; n := 2n;
    done;
    return n;
```

(A) A program over integer variables.



(B) The associated control-flow graph.

FIGURE 2.1. A simple terminating program.

program like the one in Figure 2.1a as a transition system $\mathcal{S} = \langle S, \to \rangle$ where S denotes the set of program configurations and $\to \subseteq S \times S$ denotes execution steps. In such a simple non-recursive program, the set of configurations is a variable valuation, including a program counter $\mathtt{pc}$ ranging over the finite set of program locations. For our simple program a single location suffices and we set

$$S = \{\ell_0\} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}\,, \tag{2.1}$$

where the last three components provide the values of $\mathtt{x}$, $\mathtt{y}$, and $\mathtt{n}$, and the first component the value of $\mathtt{pc}$. The corresponding transition relation is the union $\xrightarrow{a} \cup \xrightarrow{b}$ of the two possible updates, and contains for instance

$$(\ell_0, 3, 1, 4) \xrightarrow{a} (\ell_0, 2, 1, 8)$$

using transition a in Figure 2.1b.

2.2.1. Ranking Functions. We say that a transition system $\mathcal{S} = \langle S, \to \rangle$ *terminates* if every execution $s_0 \to s_1 \to \cdots$ is finite. The standard method in order to prove that a program terminates for all inputs is to exhibit a *ranking function* f into some well-order, such that $\to$-related configurations have decreasing rank [Floyd, 1967; Turing, 1949]: $s \to s'$ must imply $f(s) > f(s')$.

2.2.1.1. *Monolithic Ranking Functions.* Observe that any *deterministic* terminating program can be associated to a ranking function into $\mathbb{N}$, which maps each configuration to the number of steps before termination. We leave it as an exercise to the reader to figure out such a ranking function for Figure 2.1—the answer can be found in Remark 3.3. There are at least two motivations for considering other wqos:

- Programs can be nondeterministic, for instance due to abstractions or interactions with an environment. Then the supremum of the number of steps along all the possible paths can be used as the range for a ranking function; this is in general a countable well-order.
- Whether by automated means or by manual means, monolithic ranking functions into $\mathbb{N}$ are often hard to synthesise, and hard to check once found or guessed—note that the canonical 'number of steps' function is not recursive in general. This motivates employing more complex well (quasi-)orders in exchange for simpler ranking functions, as explained in the following.

2.2.1.2. *Lexicographic Ranking Functions.* In order to prove the termination of the program of Figure 2.1 by a ranking function, consider some (possibly infinite) execution

$$(\ell_0, x_0, y_0, n_0) \to (\ell_0, x_1, y_1, n_1) \to (\ell_0, x_2, y_2, n_2) \to \cdots \tag{2.2}$$

over S. Because a negative value for x or y would lead to immediate termination, the associated sequence of pairs

$$(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots \tag{2.3}$$

is actually over $\mathbb{N}^2$, and the successive pairs satisfy $(y_i, x_i) >_{\text{lex}} (y_{i+1}, x_{i+1})$. This shows that $f(\ell_0, x, y, n) \stackrel{\text{def}}{=} (y, x)$ ranging over the wo $\langle \mathbb{N}^2, \leq_{\text{lex}} \rangle$ is a ranking function.

Cook, See, and Zuleger [2013] and Ben-Amram and Genaim [2014] consider for instance the automatic synthesis of such lexicographic affine ranking functions for integer loops like Figure 2.1a. Ranking functions into $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ are described there by d functions $f_1, f_2, \ldots, f_d \colon S \to \mathbb{N}$ such that, whenever $s \to s'$, then $(f_1(s), f_2(s), \ldots, f_d(s)) >_{\text{lex}} (f_1(s'), f_2(s'), \ldots, f_d(s'))$; in our example $f_1(s) \stackrel{\text{def}}{=} y$ and $f_2(s) \stackrel{\text{def}}{=} x$; each function f_i is an affine function of the values of the program variables. This technique is implemented in automated termination provers like FuncTion [Courant and Urban, 2017] or T2 [Brockschmidt et al., 2016].

2.2.1.3. *Ordinal Ranking Functions.* Beyond $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$, we can consider more generally ranking functions into an ordinal [Turing, 1949]. For instance, write $\langle [k], \leq \rangle$ for the initial segment $[k] \stackrel{\text{def}}{=} \{0, \ldots, k-1\}$ of the naturals; this is a finite linear order for each k. We can then replace our previous lexicographic ranking function for Figure 2.1 with a multiset ranking function into $\langle \mathbb{M}([2]), \leq_{\text{m}} \rangle$: the ranking function $f(\ell_0, x, y, m) \stackrel{\text{def}}{=} \{1^y, 0^x\}$ associates a multiset containing y copies of the element '1' and x copies of '0' to the configuration (ℓ_0, x, y, n).

This might seem like a rather artificial example of a multiset ranking function, and indeed more generally $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ and $\langle \mathbb{M}([d]), \leq_{\text{m}} \rangle$ are *order-isomorphic* for every d: indeed, $r(n_1, \ldots, n_d) \stackrel{\text{def}}{=} \{(d-1)^{n_1}, \ldots, 0^{n_d}\}$ is a bijection such that $(n_1, \ldots, n_d) \leq_{\text{lex}} (n'_1, \ldots, n'_d)$ in $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ if and only if $r(n_1, \ldots, n_d) \leq_{\text{m}} r(n'_1, \ldots, n'_d)$ in $\langle \mathbb{M}([d]), \leq_{\text{m}} \rangle$.

In order to pick a unique representative for each isomorphism class of well orders, one can employ their *order types*, presented when they fall below ε_0 as *ordinal terms* in Cantor normal form. Using the order type computations recalled in Appendix A.1.1.2, ω^d is the order type of both $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ and $\langle \mathbb{M}([d]), \leq_{\text{m}} \rangle$, and our ranking function can also be defined as $f(\ell_0, x, y, m) \stackrel{\text{def}}{=} o((y, x), \leq_{\text{lex}}) = \omega \cdot y + x$.

2.2.2. Quasi-Ranking Functions. We introduce now a generalisation of ranking functions to work over wqos. Consider two indices $i < j$ in a putative infinite execution like (2.3) and observe that:

- either b is never fired throughout the execution between steps i and j, and then $y_i = \cdots = y_j$ and $x_i > x_j$,
- or b is fired at least once, and $y_i > y_j$.

In both cases $(x_i, y_i) \not\leq_{\times} (x_j, y_j)$, i.e. the sequence (2.3) is bad for the product ordering. Since $\langle \mathbb{N}^2, \leq_{\times} \rangle$ is a wqo, this sequence is necessarily finite, and so is the original sequence (2.2): the program of Figure 2.1 terminates on all inputs.

This termination argument for our example program easily generalises:

DEFINITION 2.1. Given a transition system $\mathcal{S} = \langle S, \to \rangle$, a *quasi-ranking function* is a map $f : S \to A$ into a wqo $\langle A, \leq \rangle$ such that, whenever $s \to^+ s'$ is a non-empty sequence of transitions of $\mathcal{S}$, $f(s) \not\leq f(s')$.

In our treatment of the program of Figure 2.1 above, we picked $f(\ell_0, x, y, z) \stackrel{\text{def}}{=} (x, y)$ and $\langle A, \leq \rangle \stackrel{\text{def}}{=} \langle \mathbb{N}^2, \leq_\times \rangle$. Note that a ranking function can be seen as a quasi-ranking function into a wo $\langle A, \leq \rangle$. Indeed, if $s \to s'$, then the condition $f(s) \not\leq f(s')$ of Definition 2.1 over a wo is equivalent to requiring $f(s) > f(s')$, and then implies by transitivity $f(s) > f(s')$ whenever $s \to^+ s'$.

2.2.2.1. *Termination by Quasi-Ranking.* The existence of a quasi-ranking function always yields termination:

PROPOSITION 2.2. *Given a transition system $\mathcal{S} = \langle S, \to \rangle$, if there exists a quasi-ranking function for $\mathcal{S}$, then $\mathcal{S}$ terminates.*

PROOF. Let f be a quasi-ranking function of $\mathcal{S}$ into a wqo $\langle A, \leq \rangle$. Any sequence of configurations $s_0 \to s_1 \to \cdots$ of $\mathcal{S}$ is associated by f to a bad sequence $f(s_0), f(s_1), \ldots$ over A and is therefore finite. □

Note that the converse statement also holds, as seen in §2.2.1.1.

2.2.2.2. *Disjunctive Termination Arguments.* In order to prove a program transition relation $\to$ to be well-founded, Geser [1990, Section 3.1] and Podelski and Rybalchenko [2004] show that it suffices to exhibit a finite set of well-founded relations $T_1, \ldots, T_d \subseteq S \times S$ and prove that the transitive closure $\to^+$ is included in the union $T_1 \cup \cdots \cup T_d$. This is also known as a *Ramsey-based termination proof*, as the original arguments relied on Ramsey's Theorem. Such termination arguments are employed in automated termination provers like Terminator [Cook et al., 2006] or CProver [Kroening et al., 2010]. An advantage of the technique over ranking functions is that synthesising the termination arguments for each T_j can be easier than finding a global ranking function over S; a disadvantage is that checking the inclusion $\to^+ \subseteq T_1 \cup \cdots \cup T_d$, which is typically encoded as an assertion passed to a safety checker, can be costly.

In practice, we can assume each of the T_j for $1 \leq j \leq d$ to be proved well-founded through a quasi-ranking function f_j into a wqo $\langle A_j, \leq_j \rangle$. In the case of the program in Figure 2.1, choosing

$$T_1 \stackrel{\text{def}}{=} \{((\ell_0, x, y, n), (\ell_0, x', y', n')) \mid x > 0 \wedge x' < x\} \tag{2.4}$$

$$T_2 \stackrel{\text{def}}{=} \{((\ell_0, x, y, n), (\ell_0, x', y', n')) \mid y > 0 \wedge y' < y\} \tag{2.5}$$

yields such a *disjunctive termination argument*, with $A_1 = A_2 = \mathbb{N}$.

Another way of understanding disjunctive termination arguments is that they define a quasi-ranking function f into the product wqo $\langle A_1 \times \cdots \times A_d, \leq_\times \rangle$, which maps a configuration s to the tuple $\langle f_1(s), \ldots, f_d(s) \rangle$, c.f. [C19, Section 7.1].

2.3. Application: Well-Structured Transition Systems

Well-structured transition systems (WSTS) form a family of computational models where the (usually infinite) set of configurations is equipped with a well-quasi-ordering that is 'compatible' with the computation steps. The existence of this well-quasi-ordering allows for the decidability of some important behavioural properties like termination (from a given initial configuration) or coverability.

Historically, the idea can be traced back to Finkel [1987] who gave a first definition for WSTS abstracting from Petri nets and fifo nets, and who showed the decidability of termination from an initial configuration and of finiteness of the set of reachable configurations (aka boundedness). Then Finkel [1994] applied the WSTS idea to decide termination from a given initial configuration in lossy channel systems, while Abdulla and Jonsson [1996] introduced the backward-chaining algorithm for coverability. One will find a good survey of these early results, and a score of WSTS examples, in [Abdulla et al., 2000; Abdulla, 2010; Finkel and Schnoebelen, 2001; Finkel and Goubault-Larrecq, 2012b]. Many new WSTS models have been introduced since (in distributed computing, software verification, logic, and other fields), using well-quasi-orderings based on trees, sequences of vectors, or graphs, rather than the more traditional vectors of natural numbers or words with the subword embedding.

2.3.1. Formal Definition. A transition system $\mathcal{S} = \langle S, \to \rangle$ is *ordered* when it is further equipped with a quasi-ordering $\leq$ of its configurations. An ordered transition system $\mathcal{S} = \langle S, \to, \leq \rangle$ is *well-structured* if $\langle S, \leq \rangle$ is a wqo and

(2.6)
$$\forall s_1, s_2, t_1 \in S,\ \big(s_1 \to s_2 \text{ and } s_1 \leq t_1\big) \text{ implies } \exists t_2 \in S,\ \big(t_1 \to t_2 \text{ and } s_2 \leq t_2\big)\ .$$

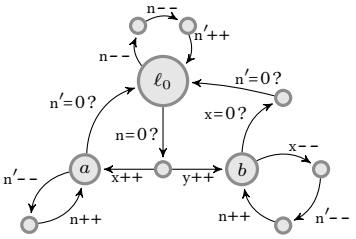
This last property is also called '*compatibility*' (of the ordering with the transitions). Formally, it just means that $\leq$ is a *simulation* relation for $\mathcal{S}$, in precisely the classical sense of Milner [1990]. The point of (2.6) is to ensure that a larger configuration can do at least as much as a smaller configuration.

Remark 2.3. Equation (2.6) comes in many variants. For example, Finkel and Schnoebelen [2001] consider *strict compatibility* (when $<$, the strict ordering underlying $\leq$, is a simulation), *transitive compatibility* (when $\leq$ is a weak simulation), and the definition can further extend to *labelled* transition systems. These are all inessential variations of the main idea.

2.3.2. Example: Lossy Counter Machines. Let us consider *lossy counter machines* (LCM) [Mayr, 2003] as an example of a family of WSTSs. Such counter machines are syntactically identical to Minsky machines $\langle Q, \mathrm{C}, \delta, q_0 \rangle$, where the transitions in $\delta \subseteq Q \times \mathrm{C} \times \{\mathrm{c}{=}0\,?, \mathrm{c}{+}{+}, \mathrm{c}{-}{-}\} \times Q$ operate on a finite set C of counters through *zero-tests* $\mathrm{c}{=}0\,?$, *increments* $\mathrm{c}{+}{+}$ and *decrements* $\mathrm{c}{-}{-}$. The semantics of an LCM differ however from the usual, 'reliable' semantics of a counter machine in that the counter values can decrease in an uncontrolled manner at any point of the execution. These unreliable behaviours make several problems decidable on LCMs, contrasting with the situation with Minsky machines [Schnoebelen, 2010b], the intuition being that, in order to compute anything, LCMs must introduce considerable redundancy and robustness to losses, which makes their analysis possible. These positive decidability results should however be taken with a grain of salt: as we will see in Section 4.2, LCMs and related models like reset Petri nets are the main source of decision problems with provably Ackermannian complexity [Schnoebelen, 2002, 2010a; Urquhart, 1999].

Formally, a configuration $q, \boldsymbol{v}$ associates a control location q in Q with a counter valuation $\boldsymbol{v}$ in $\mathbb{N}^{\mathrm{C}}$, i.e. counter values can never go negative. Observe that the set of configurations $Q \times \mathbb{N}^{\mathrm{C}}$ is well-quasi-ordered by the product ordering: $q, \boldsymbol{v} \leq q', \boldsymbol{v}'$ if and only if $q = q'$ and $\boldsymbol{v}(\mathrm{c}) \leq \boldsymbol{v}'(\mathrm{c})$ for all $\mathrm{c} \in \mathrm{C}$.

```
ℓ0 : while n mod 2 = 0 do
       n := n/2;
       if x = n then
b :      y := y+1; x := 0;
       else
a :      x := x+1;
     done
```



(A) A program inversing Figure 2.1.(B) A counter machine implementing Figure 2.2a.

FIGURE 2.2. The counter machine for Example 2.4.

Let the initial configuration be $q_0, \mathbf{0}$. A transition of the form $(q, \mathrm{c}, \mathrm{op}, q')$ defines a *reliable computation step* $q, \boldsymbol{v} \to q', \boldsymbol{v}'$, where $\boldsymbol{v}(\mathrm{c}') = \boldsymbol{v}'(\mathrm{c}')$ for all $\mathrm{c} \neq \mathrm{c}'$ in C and

- if $\mathrm{op} = =0\,?$, then $\boldsymbol{v}(\mathrm{c}) = \boldsymbol{v}'(\mathrm{c}) = 0$,
- if $\mathrm{op} = \texttt{++}$, then $\boldsymbol{v}(\mathrm{c}) + 1 = \boldsymbol{v}'(\mathrm{c})$, and
- if $\mathrm{op} = \texttt{--}$, then $\boldsymbol{v}(\mathrm{c}) = \boldsymbol{v}'(\mathrm{c}) + 1$.

A *lossy computation step* is then defined by allowing counter values to decrease arbitrarily between reliable steps: $(q, \boldsymbol{v}) \to_\ell (q', \boldsymbol{v}')$ if and only if there exist $\boldsymbol{w} \leq \boldsymbol{v}$ and $\boldsymbol{w}' \geq \boldsymbol{v}'$ such that $(q, \boldsymbol{w}) \to (q', \boldsymbol{w}')$. Now, lossy steps are visibly compatible with $\leq$ according to (2.6), and thus the transition system $\langle Q \times \mathbb{N}^{\mathrm{C}}, \to_\ell, \leq \rangle$ defined by the lossy operational semantics is a WSTS.

EXAMPLE 2.4 (Weak Inverse of Figure 2.1). Figure 2.2b displays a counter machine with counters $\mathrm{C} \stackrel{\text{def}}{=} \{\mathrm{x}, \mathrm{y}, \mathrm{n}, \mathrm{n}'\}$. Figure 2.2a shows the pseudo-code for the computation performed by the machine, where an auxiliary counter n' is used to perform division by two. If started in configuration $(\ell_0, 0, 0, N, 0)$ and assuming reliable semantics, then this machine enumerates, each time it reaches the state ℓ_0 with $\mathbf{n}'$ empty, the values of x, y, n such that the program of Figure 2.1 terminates on input x, y, n and returns N. With lossy semantics, it might also reach smaller values of x, y, n; as will be recalled more formally in Section 4.2, the LCM of Figure 2.2b is a *weak computer* for the function defined by Figure 2.1.

2.3.3. Verifying WSTS. A number of decision problems can be solved in WSTS using generic algorithms relying on the underlying wqo for termination and on variants of compatibility (2.6) for correctness, including termination (from a given initial configuration), inevitability, boundedness, regular simulations, etc.

2.3.3.1. *Coverability.* We focus here on the *coverability problem*.

PROBLEM (Coverability).

instance: An ordered transition system $\langle S, \to, \leq \rangle$ and two configurations s, t in S.

question: Is t *coverable* from s, i.e. is there a run $s = s_0 \to^* s_n \geq t$?

This problem corresponds to the verification of *safety* properties, i.e. that no bad configuration can ever be reached: here t models an error configuration, and we assume that any configuration larger than t is also an error.

In the particular case of a WSTS over configuration space $Q \times A$ for some finite set of control configurations Q and some wqo domain $\langle A, \leq_A \rangle$, *control-state reachability* asks whether some state q can be reached, regardless of the

associated element of A. 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. $(q, x) \leq_\times (q', x')$ if and only if $q = q'$ and $x \leq_A x'$.

2.3.3.2. *Backward Coverability.* The decidability of WSTS Coverability uses a set-saturation method, whose termination relies on the Ascending Chain Condition mentioned in §2.1.1.1. This particular algorithm is called the *backward coverability algorithm*, because it computes

$$\mathrm{Pre}^*_\exists(\uparrow t) \stackrel{\text{def}}{=} \{s' \in S \mid \exists t' \geq t,\ s' \to^* t'\} \tag{2.7}$$

by backward chaining; it only remains to check whether $s \in \mathrm{Pre}^*_\exists(\uparrow t)$ in order to answer the coverability instance. This idea is also at the heart of the algorithms for many other properties defined by fixpoints, see [Bertrand and Schnoebelen, 2013]. The first known published instance of backward coverability seems to be due to Arnold and Latteux [1978] for reset Petri nets, but the algorithm was independently rediscovered by Abdulla and Jonsson [1996] for lossy channel systems and its abstract formulation was popularised in the surveys [Abdulla et al., 2000; Finkel and Schnoebelen, 2001].

For a set of configurations $U \subseteq S$, define its (existential) *predecessor set* as

$$\mathrm{Pre}_\exists(U) \stackrel{\text{def}}{=} \{s \in S \mid \exists s' \in U,\ s \to s'\}\ . \tag{2.8}$$

The backward coverability algorithm computes the limit of the sequence

$$\uparrow t = U_0 \subseteq U_1 \subseteq \cdots \text{ where } U_{n+1} \stackrel{\text{def}}{=} U_n \cup \mathrm{Pre}_\exists(U_n) \tag{2.9}$$

where $U_n = \{s' \in S \mid \exists t' \geq t,\ s' \to^{\leq n} t'\}$ is the set of configurations that cover t in at most n steps.

2.3.3.3. *Termination and Effectiveness of the Algorithm.* There is no reason for the chain in (2.9) to converge in general, but it does in the case of a WSTS.

Lemma 2.5. *If $U \subseteq S$ is an upwards-closed set of configurations, then* $\mathrm{Pre}_\exists(U)$ *is upwards-closed.*

Proof. Assume $s \in \mathrm{Pre}_\exists(U)$. Then $s \to t$ for some $t \in U$. By (2.6), if $s' \geq s$, then $s' \to t'$ for some $t' \geq t$. Thus $t' \in U$ and $s' \in \mathrm{Pre}_\exists(U)$. □

A corollary is that sequence (2.9) stabilises to $\bigcup_{i\in\mathbb{N}} U_i = \mathrm{Pre}^*_\exists(\uparrow t)$ after a finite amount of time thanks to the Ascending Chain Condition.

Moreover, as seen in §2.1.1.2, the Finite Basis Property ensures that all the sets U_i can be finitely represented using their minimal elements, and the union or inclusion of two upwards-closed sets can be computed. The last ingredient is an effectiveness assumption:

- $\langle S, \leq\rangle$ should be effective, meaning that S is recursive and the ordering $\leq$ is decidable, and
- there exists an algorithm accepting any configuration $s \in S$ and returning $\mathrm{pb}(s)$, a finite basis for $\mathrm{Pre}_\exists(\uparrow s)$; this is known as the *effective pred-basis* assumption.

Remark 2.6 (Coverability Pseudo-Witnesses). For future reference, it is worth detailing again the argument for the Ascending Chain Condition in the particular setting of backward coverability: we can extract a sequence of minimal elements $t_0, t_1, \ldots$ from $U_0 \subsetneq U_1 \subsetneq \cdots$ such that $t_0 \stackrel{\text{def}}{=} t$ and $t_{i+1} \in U_{i+1} \setminus U_i$ for all i, which entails that the sequence $t_0, t_1, \ldots$ is bad and therefore finite.

In fact, we can be more precise, and pick t_{i+1} at each step among the minimal elements of $\mathrm{Pre}_\exists(\uparrow t_i)$. Indeed, we can picture the computation in (2.9) as a tree rooted in t, where each node t' has an edge to each minimal element of $\mathrm{Pre}_\exists(\uparrow t')$: for each level i of this tree, the upward closure of the set of nodes at levels at most i is exactly U_i. The tree is finitely branching by the Finite Basis Property, and finite depth since every branch is a bad sequence, hence is finite by Kőnig's Lemma, and its height is exactly the number of steps before termination of the backward coverability algorithm. We call such a bad sequence $t_0, t_1, \ldots, t_n$ with

$$t_0 \stackrel{\text{def}}{=} t \;, \qquad t_{i+1} \in \min \mathrm{Pre}_\exists(\uparrow t_i) \;, \qquad t_n \leq s \tag{2.10}$$

a *pseudo-witness* of the coverability of t from s.

Example 2.7 (Predecessors in Lossy Counter Machines). The previous two effectiveness assumptions hold in all the 'natural' families of WSTS. In the case of LCMs, $\langle Q \times \mathbb{N}^{\mathrm{C}}, \leq \rangle$ is certainly effective, and a finite basis of predecessors of a configuration $q', \boldsymbol{v}'$ can be computed by

$$\mathrm{pb}(q', \boldsymbol{v}') = \min\{q, \mathrm{pre}_{\mathrm{c\,op}}(\boldsymbol{v}') \mid (q, \mathrm{c}, \mathrm{op}, q') \in \delta \text{ and } \boldsymbol{v}'(\mathrm{c}) = 0 \text{ if op} = =0\,?\}$$

where $\mathrm{pre}_{\mathrm{c\,op}}(\boldsymbol{v}')$ is a vector in $\mathbb{N}^{\mathrm{C}}$ with $\mathrm{pre}_{\mathrm{c\,op}}(\boldsymbol{v}')(\mathrm{c}') \stackrel{\text{def}}{=} \boldsymbol{v}'(\mathrm{c}')$ for all $\mathrm{c}' \neq \mathrm{c}$ in C, and

$$\begin{aligned} \mathrm{pre}_{\mathrm{c}=0\,?}(\boldsymbol{v}')(\mathrm{c}) &\stackrel{\text{def}}{=} \boldsymbol{v}'(\mathrm{c}) = 0 \;, \\ \mathrm{pre}_{\mathrm{c}++}(\boldsymbol{v}')(\mathrm{c}) &\stackrel{\text{def}}{=} \max\{0, \boldsymbol{v}'(\mathrm{c}) - 1\} \;, \\ \mathrm{pre}_{\mathrm{c}--}(\boldsymbol{v}')(\mathrm{c}) &\stackrel{\text{def}}{=} \boldsymbol{v}'(\mathrm{c}) + 1 \;. \end{aligned}$$

Example 2.8 (Weak Inverse Computer and Coverability). Let us consider once more Example 2.4 and Figure 2.2b with initial configuration $s \stackrel{\text{def}}{=} (\ell_0, 0, 0, N, 0)$. Because it is a weak computer for inversing the program of Figure 2.1, the configuration $t \stackrel{\text{def}}{=} (\ell_0, x, y, n, 0)$ can be covered if and only if the program of Figure 2.1 would have returned at least N when started with x, y, n as inputs. Now, the backward coverability algorithm with t as target is essentially simulating an execution of the program on input x, y, n, and will require at least as many steps as it took the program to terminate.

CHAPTER 3

Length Function Theorems

This chapter presents some contributions to the complexity analysis of wqo-based termination arguments published jointly with Figueira, Figueira, and Schnoebelen [C19; C18; I3]. We aim to extract complexity upper bounds on the running time of algorithms from their termination proof.

It turns out that, by suitably controlling how 'large' the elements can grow in bad sequences (see Section 3.1), we can derive upper bounds on the time and space required by the algorithms presented in Sections 2.2 and 2.3. I present two *length function theorems*, which are combinatorial statements on the length of controlled bad sequences: one for sequences of ordinals below ε_0 in Section 3.3 and one for tuples of natural numbers in Section 3.4, and how to apply them to the examples from Chapter 2.

A major drawback of all these complexity bounds is that they are *very* high—i.e., non-elementary except in trivial cases—, whereas practitioners are mostly interested in polynomial bounds. This is the price to pay for the generality of the approach: the class of programs terminating thanks to a quasi-ranking function encompasses programs of high complexity.

Example 3.1. For instance, simple integer loops can already be deceivingly simple: recall that the program of Figure 2.1 terminated using a straightforward ranking function into ω^2. Although this is just one notch above a ranking function into ω, we can already witness fairly complex computations. Observe indeed that the following steps can be executed in our program:

$$\begin{aligned}(\ell_0, x, y, 1) &\xrightarrow{a^x b} (\ell_0, 2^x, y-1, 2^{x+1}) \\ &\xrightarrow{a^{2^x} b} (\ell_0, 2^{2^x+x+1}, y-2, 2^{2^x+x+2}) \\ &\xrightarrow{a^{2^{2^x+x+1}} b} (\ell_0, 2^{2^{2^x+x+1}+2^x+x+2}, y-3, 2^{2^{2^x+x+1}+2^x+x+3}) \,.\end{aligned}$$

Continuing this execution, we see that our simple program exhibits executions of length greater than a tower of exponentials in y, i.e. it is non-elementary. As shown by Example 2.8, this non-elementary lower bound also applies to the number of steps required by the backward coverability algorithm to terminate on the LCM of Figure 2.2b with target configuration $(\ell_0, x, y, 1, 0)$.

Such high complexities justify the use of ordinal-indexed *subrecursive functions* in order to denote non-elementary growths. We will recall the definitions of two families of such functions in Section 3.2.

The presentation in this chapter is based chiefly on the lecture notes [L1] and the invited papers [I4; I3].

Contents

3.1. Controlling Bad Sequences

Both in the case of quasi-ranking functions of Section 2.2.2 and in that of the backward coverability algorithm of Section 2.3.3, the running time of the algorithm is essentially bounded by the length of the bad sequences constructed by the termination argument. By definition, bad sequences in a wqo are always finite, but no statement is made regarding *how long* they can be. This is for a very good reason: they can be arbitrarily long.

For instance, over the wo $\langle \mathbb{N}, \leq \rangle$,

$$n, n-1, \ldots, 1, 0 \tag{3.1}$$

is a bad sequence of length $n+1$ for every n. Arguably, this is not so much of an issue, since what we are really interested in is the length *as a function* of the initial configuration—which includes the inputs to the program. Thus (3.1) is the maximal bad sequence over $\langle \mathbb{N}, \leq \rangle$ with initial element of 'size n.'

However, as soon as we move to more complex wqos, we can exhibit arbitrary bad sequence lengths even with fixed initial configurations. For instance, over $\langle \mathbb{N}^2, \leq_{\text{lex}} \rangle$,

$$(1,0), (0,n), (0,n-1), \ldots, (0,1), (0,0) \tag{3.2}$$

is a bad sequence of length $n+2$ for every n starting from the fixed $(1,0)$. Nonetheless, the behaviour of a program exhibiting such a sequence of ranks is

rather unusual: such a sudden 'jump' from $(1,0)$ to an arbitrary $(0,n)$ is not possible in a deterministic program once the user inputs have been provided.

In the following, we will assume that no such arbitrary jump can occur. This comes at the price of some loss of generality in the context of termination analysis, where nondeterministic assignments of arbitrary values are typically employed to model values provided by the environment—for instance interactive user inputs or concurrently running programs—, or because of abstracted operations. Thankfully, in most cases it is easy to *control* how large the program variables can grow during the course of an execution.

3.1.1. Normed wqos. Given a wqo $\langle A, \leq_A \rangle$, we posit a *norm* function $|.|_A$ from A to $\mathbb{N}$ on the elements of A. In order to be able to derive combinatorial statements, we require

$$A_{\leq n} \stackrel{\text{def}}{=} \{x \in A \mid |x|_A \leq n\} \tag{3.3}$$

to be finite for every n. We call the resulting structure $\langle A, \leq_A, |\cdot|_A \rangle$ a *normed wqo* (nqo).

We will use the following norms on some of the wqos defined earlier:

- in a finite Q, all the elements have the same norm 0;
- in $\mathbb{N}$ or $[d]$, n has norm $|n|_{\mathbb{N}} = n$;
- in disjoint sums $A_0 \sqcup A_1$, (x,i) uses the norm $|x|_{A_i}$ of the underlying set;
- in Cartesian or lexicographic products with support $A \times B$, (x,y) has the infinity norm $\max(|x|_A, |y|_B)$; finally,
- in multisets $\mathbb{M}(A)$, m has norm $\max_{x \in A, m(x)>0}(\max(m(x), |x|_A))$.

Regarding ordinals, recall that ε_0 is the smallest fixed point of the equation $\omega^x = x$, and consider some $\gamma < \varepsilon_0$. We define the norm of an ordinal $\alpha < \gamma$ as the maximal coefficient that appears in its associated Cantor normal form: if $\alpha = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_p} \cdot c_p < \gamma$ with $\alpha_1 > \cdots > \alpha_p$ and $c_1, \ldots, c_p > 0$, then

$$N\alpha \stackrel{\text{def}}{=} \max\{c_1, \ldots, c_p, N\alpha_1, \ldots, N\alpha_p\} \; . \tag{3.4}$$

Observe that this definition essentially matches the previously defined norms over multisets and tuples of vectors: in $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$, we have

$$\begin{aligned} No((n_1, \ldots, n_d), \leq_{\text{lex}}) &= N(\omega^{d-1} \cdot n_1 + \cdots + \omega^0 \cdot n_d) \\ &= \max(d-1, |(n_1, \ldots, n_d)|_{\mathbb{N}^d}) \; , \end{aligned}$$

and in $\langle \mathbb{M}(\mathbb{N}^d), \leq_{\text{m}} \rangle$ for multisets of lexicographically-ordered tuples,

$$No(m, \leq_{\text{m}}) = \max(d-1, |m|_{\mathbb{M}(\mathbb{N}^d)}) \; .$$

3.1.2. Controlled Sequences. Let $g \colon \mathbb{N} \to \mathbb{N}$ be a monotone and inflationary function: for all x, x', $x \leq x'$ implies $g(x) \leq g(x')$ and $x \leq g(x)$.

Definition 3.2 (Controlled Sequence; Cichoń and Tahhan Bittar, 1998). We say that a sequence $x_0, x_1, x_2, \ldots$ of elements in A is *amortised controlled by* (g, n_0) for some n_0 in $\mathbb{N}$ if

$$|x_i|_A \leq g^i(n_0) \tag{3.5}$$

for all i, where g^i denotes the ith iterate of g. We say that it is *strongly controlled by* (g, n_0) for some n_0 in $\mathbb{N}$ if

$$|x_0|_A \leq n_0 \quad \text{and} \quad |x_{i+1}|_A \leq g(|x_i|)_A \tag{3.6}$$

for all i.

Observe that by definition a strongly controlled sequence is also amortised controlled: $|x_0|_A \leq g^0(n_0) = n_0$, which prompts the name of *initial norm* for n_0, and amortised steps cannot grow faster than g the *control function*. In the following we simply write '(g, n_0)-controlled' instead of 'amortised controlled by (g, n_0)'.

3.1.2.1. *Example: Controlled Quasi-Rankings.* The notion of control can be lifted to the level of the quasi-ranking functions that generate them. Let us posit a *configuration norm* $\iota\colon S \to \mathbb{N}$. Then a quasi-ranking function $f\colon S \to A$ for a transition system $\mathcal{S} = \langle S, \to\rangle$ and a normed wqo $\langle A, \leq_A, |.|_A\rangle$ is (g, ι)-*controlled* if, for any execution $s_0 \to s_1 \to \cdots$ of $\mathcal{S}$, for all i,

$$|f(s_i)|_A \leq g^i(\iota(s_0)) \ . \tag{3.7}$$

This ensures that any sequence $f(s_0), f(s_1), \ldots$ of ranks associated to the execution is $(g, \iota(s_0))$-controlled.

For instance, our ranking function $f(\ell_0, x, y, n) \stackrel{\text{def}}{=} (y, x)$ for the program of Figure 2.1 into $\langle \mathbb{N}^2, \leq_{\text{lex}}\rangle$ is g-controlled for $g(x) \stackrel{\text{def}}{=} 2x$ and $\iota(\ell_0, x, y, n) \stackrel{\text{def}}{=} \max(x, y, n)$. Observe that having a configuration norm $\iota(s_0)$ rather than simply using the norm of the initial rank $|f(s_0)|_A$ gives us some extra flexibility: in our example, the latter equals $\max(x, y)$, and the sequences of ranks $f(s_0), f(s_1), \ldots$ are in general not $(g, |f(s_0)|_A)$-controlled because the value of n has been lost.

3.1.2.2. *Example: Controlled Minimal Predecessors.* Let us see how this setting applies to the backward coverability algorithm of §2.3.3.2. In order to control the bad sequence of configurations extracted from the Ascending Chain Condition applied to (2.9), observe that the initial norm n_0 can be taken as $|t|_S$. Regarding the control function g, the ith iterate $g^i(n_0)$ of control function should then bound the norm of the minimal elements of U_i.

In the case of lossy counter machines, the computation of minimal predecessors in Example 2.7 shows that a strong control by $H(x) \stackrel{\text{def}}{=} x + 1$ suffices.

3.1.3. Length Functions. The point of controlled sequences is that their length can be bounded. Consider for this the tree one obtains by sharing the common prefixes of all the (g, n_0)-controlled bad sequences over a normed wqo $\langle A, \leq_A, |.|_A\rangle$. This tree has

- finite branching by (3.3) and (3.5), more precisely of branching degree bounded by the cardinal of $A_{\leq g^i(n_0)}$ for a node at depth i, and
- no infinite branches thanks to the wqo property.

By Kőnig's Lemma, this tree of bad sequences is therefore finite, of some height $L_{g,n_0,A}$ representing the length of the maximal (g, n_0)-controlled bad sequence(s) over A. In the following, since we are mostly interested in this length as a function of the initial norm n_0, we will see this as a *length function* $L_{g,A}(n_0)$.

Observe that $L_{g,A}$ bounds the asymptotic execution length in a program endowed with a g-controlled quasi-ranking function into $\langle A, \leq_A, |.|_A\rangle$. It similarly bounds the number of steps required by the backward coverability algorithm for a WSTS over $\langle A, \leq_A, |.|_A\rangle$ with a target configuration of norm $|t|_A \leq n_0$ when minimal elements in U_i are of norm at most $g^i(n_0)$.

Our purpose will thus be to obtain explicit complexity bounds on $L_{g,A}$ depending on g and A. We call such combinatorial statements *length function theorems*. But first we need to make a detour via subrecursive functions, which we employ in order to express those explicit upper bounds.

3.2. Subrecursive Hierarchies

As we saw with the example of Figure 2.1, even simple terminating programs can have a very high complexity. In order to express such high bounds, a convenient tool is found in *subrecursive hierarchies*, which employ induction over ordinal indices to define faster and faster growing functions. We recall the definition of two such hierarchies [see e.g. Cichoń and Tahhan Bittar, 1998; Fairtlough and Wainer, 1998; Schwichtenberg and Wainer, 2012; Wainer, 1972]. We shall strive to employ notations compatible with those of Schwichtenberg and Wainer [2012, Chapter 4], and refer the interested reader to their monograph for proofs and additional material.

3.2.1. Fundamental Sequences and Predecessors. Let us first introduce some notions on ordinal terms. Consider an ordinal term α in Cantor normal form $\omega^{\alpha_1} + \cdots + \omega^{\alpha_p}$. In this representation, $\alpha = 0$ if and only if $p = 0$. An ordinal α of the form $\alpha' + 1$ (i.e. with $p > 0$ and $\alpha_p = 0$) is called a *successor* ordinal, and otherwise if $\alpha > 0$ it is called a *limit* ordinal, and can be written as $\gamma + \omega^\beta$ by setting $\gamma = \omega^{\alpha_1} + \cdots + \omega^{\alpha_{p-1}}$ and $\beta = \alpha_p$. We usually write 'λ' to denote a limit ordinal.

A *fundamental sequence* for a limit ordinal λ is a sequence $(\lambda(x))_{x<\omega}$ of ordinal terms with supremum λ. We use the standard assignment of fundamental sequences to limit ordinals in Cantor normal form, defined inductively by

$$(\gamma + \omega^{\beta+1})(x) \stackrel{\text{def}}{=} \gamma + \omega^\beta \cdot (x+1) \;, \qquad (\gamma + \omega^\lambda)(x) \stackrel{\text{def}}{=} \gamma + \omega^{\lambda(x)} \;. \tag{3.8}$$

This particular assignment satisfies e.g. $0 < \lambda(x) < \lambda(y)$ for all $x < y$. For instance, $\omega(x) = x + 1$, $(\omega^{\omega^4} + \omega^{\omega^3+\omega^2})(x) = \omega^{\omega^4} + \omega^{\omega^3+\omega\cdot(x+1)}$. We also define a fundamental sequence for ε_0 itself, so that we may be able to use it in our hierarchies: $\varepsilon_0(0) \stackrel{\text{def}}{=} \omega$ and $\varepsilon_0(i+1) \stackrel{\text{def}}{=} \omega^{\varepsilon_0(i)}$, so that $\varepsilon_0(i)$ is a tower of ω's of height $i+1$.

The *predecessor* $P_x(\alpha)$ of an ordinal term $\alpha > 0$ at a value x in $\mathbb{N}$ is defined inductively by

$$P_x(\alpha+1) \stackrel{\text{def}}{=} \alpha \;, \qquad P_x(\lambda) \stackrel{\text{def}}{=} P_x(\lambda(x)) \;. \tag{3.9}$$

In essence, the predecessor of an ordinal is obtained by repeatedly taking the xth element in the fundamental sequence of limit ordinals, until we finally reach a successor ordinal and remove 1. For instance, $P_x(\omega^2) = P_x(\omega \cdot (x+1)) = P_x(\omega \cdot x + x + 1) = \omega \cdot x + x$.

3.2.2. Hardy and Cichoń Hierarchies. In the context of controlled sequences, the hierarchies of Hardy and Cichoń turn out to be especially well-suited [Cichoń and Tahhan Bittar, 1998]. Let $h\colon \mathbb{N} \to \mathbb{N}$ be a function. The *Hardy hierarchy* $(h^\alpha)_{\alpha\in\varepsilon_0}$ is defined for all $0 < \alpha < \varepsilon_0$ by[1]

$$h^0(x) \stackrel{\text{def}}{=} x \;, \qquad h^\alpha(x) \stackrel{\text{def}}{=} h^{P_x(\alpha)}(h(x)) \;, \tag{3.10}$$

[1]Note that this is equivalent to defining $h^{\alpha+1}(x) \stackrel{\text{def}}{=} h^\alpha(h(x))$ and $h^\lambda(x) \stackrel{\text{def}}{=} h^{\lambda(x)}(x)$.

and the *Cichoń hierarchy* $(h_\alpha)_{\alpha\in\varepsilon_0}$ is similarly defined for all $0 < \alpha < \varepsilon_0$ by

$$h_0(x) \stackrel{\text{def}}{=} 0 \;, \qquad\qquad h_\alpha(x) \stackrel{\text{def}}{=} 1 + h_{P_x(\alpha)}(h(x)) \;. \tag{3.11}$$

Observe that h^k for some finite k is the kth iterate of h. This intuition carries over: h^α is a transfinite iteration of the function h, using diagonalisation in the fundamental sequences to handle limit ordinals.

We should emphasise that h^α and h_α are defined for any function h; thus later we will see g_α meaning 'the αth Cichoń function based on g.' This provides quite some flexibility, but one standard choice for the initial function is the successor function, noted $H(x) \stackrel{\text{def}}{=} x + 1$. In that case, we see that a first diagonalisation yields $H^\omega(x) = H^x(x+1) = 2x+1$. The next diagonalisation occurs at $H^{\omega\cdot 2}(x) = H^{\omega+x}(x+1) = H^\omega(2x+1) = 4x+3$. Fast-forwarding a bit, we get for instance a function of exponential growth $H^{\omega^2}(x) = 2^{x+1}(x+1) - 1$, and later a non-elementary function $H^{\omega^3}(x)$ akin to a tower of exponentials of height x, an 'Ackermannian' non primitive-recursive function H^{ω^ω}, and a 'hyper-Ackermannian' non multiply recursive-function $H^{\omega^{\omega^\omega}}$.

The Hardy functions also enjoy some nice identities: for all h, α, β, and x

$$h^\alpha \circ h^\beta(x) = h^{\alpha+\beta}(x) \;, \qquad\qquad (h^\alpha)^\beta(x) = h^{\alpha\cdot\beta}(x) \;, \tag{3.12}$$

provided $\alpha + \beta$ (resp. $\alpha \cdot \beta$) satisfies specific conditions like being *tree-structured* [see, e.g., Cichoń and Tahhan Bittar, 1998; Fairtlough and Wainer, 1998]; this will always be the case in our applications of these identities.

Regarding the Cichoń functions, an induction on α shows that $H^\alpha(x) = H_\alpha(x) + x$. More generally, if h is strictly inflationary, i.e. if $h(x) > x$ for all x, then

$$h^\alpha(x) \geq h_\alpha(x) + x \;. \tag{3.13}$$

Remark 3.3 (Measuring Control and Length). On the one hand, Hardy functions are well-suited for expressing large iterates of a control function, and therefore for bounding the norms of elements in a controlled sequence. For instance, the program in Figure 2.1 computes $g^{\omega\cdot y+x}(n)$ for the function $g(x) \stackrel{\text{def}}{=} 2x$ when run on non-negative inputs x, y, n. Note that $g^x(n) = 2^x n$ and $g^{\omega\cdot y}(n)$ is greater than a tower of exponentials of height y.

On the other hand, Cichoń functions are well-suited for expressing the length of controlled sequences. For instance, $g_{\omega\cdot y+x}(n)$ is the length of the execution of the program. This relation is a general one: we can compute how many times we should iterate h in order to compute $h^\alpha(x)$ using the corresponding Cichoń function [Cichoń and Tahhan Bittar, 1998]:

$$h^\alpha(x) = h^{h_\alpha(x)}(x) \;. \tag{3.14}$$

3.3. Length Functions for Ordinals Below ε_0

Our first length function theorem, which was proven in [I3], relies on two main ingredients: a *descent equation* established in [C18] for all normed wqos, and an alternative characterisation of the Cichoń hierarchy in terms of maximisations as observed in [Buchholz et al., 1994; Cichoń, 1993].

3.3.1. Residuals and a Descent Equation. Let $\langle A, \leq, |.|_A \rangle$ be a normed wqo and x be an element of A. We write

$$A/x \stackrel{\text{def}}{=} \{y \in A \mid x \not\leq y\} \tag{3.15}$$

for the *residual* of A in x. Observe that by the wqo property, there cannot be infinite sequences of residuations $A/x_0/x_1/x_2/\cdots$ because $x_i \not\leq x_j$ for all $i < j$.

Consider now a (g, n)-controlled bad sequence $x_0, x_1, x_2, \ldots$ over A. Assuming the sequence is not empty, then because this is a bad sequence we see that for all $i > 0$, $x_0 \not\leq x_i$, i.e. that the suffix $x_1, x_2, \ldots$ is actually a bad sequence over A/x_0. This suffix is now $(g, g(n))$-controlled, and thus of length bounded by $L_{g,A/x_0}(g(n))$. This yields the following *descent equation* when considering all the possible (g, n)-controlled bad sequences [C18]:

$$L_{g,A}(n) = \max_{x \in A_{\leq n}} 1 + L_{g,A/x}(g(n)) \ . \tag{3.16}$$

In the case of a wo $\langle \alpha, \leq, N \rangle$, residuals can be expressed more simply for $\beta \in \alpha$, because

$$\alpha/\beta = \{\gamma \in \alpha \mid \beta > \gamma\} = \beta \ . \tag{3.17}$$

Thus in this case the descent equation simplifies into

$$L_{g,\alpha}(n) = \max_{\beta < \alpha, N\beta \leq n} 1 + L_{g,\beta}(g(n)) \ . \tag{3.18}$$

3.3.2. Upper Bounds. We are now equipped to prove a length function theorem for all ordinals α below ε_0, i.e. an explicit expression for $L_{g,\alpha}$ for the wo $\langle \alpha, \leq, N \rangle$. The reader might have noticed a resemblance between the ordinal descent equation (3.18) and the definition of the Cichoń hierarchy (3.11). It turns out that they are essentially the same functions: indeed, we are going to see in Proposition 3.4 that a fundamental insight of Cichoń [1993] and Buchholz, Cichoń, and Weiermann [1994] applies perfectly to our particular setting: if $N\alpha \leq x$, then choosing $\beta = P_x(\alpha)$ maximises $h_\beta(h(x))$ among those $\beta < \alpha$ with $N\beta \leq x$.

Proposition 3.4 (I3). *Let $\alpha < \varepsilon_0$ and $x \geq N\alpha$. Then*

$$h_\alpha(x) = \max_{\beta < \alpha, N\beta \leq x} 1 + h_\beta(h(x)) \ .$$

Theorem 3.5 (Length Function Theorem for Ordinals). *Let $\alpha < \varepsilon_0$ and $x \geq N\alpha$. Then $L_{g,\alpha}(x) = g_\alpha(x)$.*

Proof. We use the ordinal descent equation (3.18) and Proposition 3.4. □

As an immediate corollary, we can bound the asymptotic complexity of programs proven to terminate through a (g, ι)-controlled ranking function:

Corollary 3.6. *Given a transition system $\mathcal{S} = \langle S, \to \rangle$ with initial configuration s_0, if there exists a (g, ι)-controlled ranking function into $\alpha < \varepsilon_0$, then $\mathcal{S}$ runs in time $O(g_\alpha(\iota(s_0)))$.*

Example 3.7 (Lexicographic Ranking Functions). As an illustration, a program proven to terminate thanks to a (g, ι)-controlled ranking function ranging over $\langle \mathbb{N}^d, \leq_{\text{lex}}, |.|_{\mathbb{N}^d} \rangle$ has therefore an $O(g_{\omega^d}(n))$ bound on its worst-case asymptotic time complexity for $n = \iota(s_0)$.

In the case of the program of Figure 2.1, Corollary 3.6 yields an upper bound on its time complexity in $O(g_{\omega^2}(m)) = O(g_{\omega \cdot m + m}(m))$, where $g(x) \stackrel{\text{def}}{=} 2x$ and

$m = \iota(\ell_0, x, y, n) \stackrel{\text{def}}{=} \max(x, y, n)$. This is very close to its actual complexity, which is exactly $g_{\omega \cdot y + x}(n)$. Regarding its space complexity, by (3.14) it is in $O(\log(g^{\omega^2}(m)))$.

REMARK 3.8 (Abstractions). The bound $g_{\omega^d}(n)$ provided by Corollary 3.6 in the case of lexicographic ranking functions into $\mathbb{N}^d$ is primitive-recursive (in the control function g). This might be taken to imply that non-primitive recursive programs are beyond the reach of the current automated termination methods, which usually rely on the synthesis of affine ranking functions.

This is not the case, as for instance the Ackermann function is correctly analysed as terminating by termination checkers. We can better understand this apparent paradox with the example of *size-change termination* proofs: Lee, Jones, and Ben-Amram [2001] consider as their Example 3 a program computing the two-arguments Ackermann function:

```
a(m, k) = if m = 0 then k + 1 else
          if k = 0 then a(m−1, 1)
                   else a(m−1, a(m, k−1))
```

They construct a size-change graph on two variables to prove its termination. The longest decreasing sequence in such a graph is of length $O(n^2)$; more generally, Colcombet, Daviaud, and Zuleger [2014] showed that the asymptotic worst-case complexity in a size-change graph is $\Theta(n^r)$ for a computable rational r. Here we witness a large gap between the actual program complexity and the complexity derived from its termination argument: the Ackermann function vs. an $O(n^2)$ bound.

The source of this apparent paradox is *abstraction*: the size-change graph for a(m, k) terminates if and only if the original program does, but the time complexity is not preserved by this abstraction. In the example of the Ackermann function, the call stack is abstracted away, whereas we should include it in order for Theorem 3.5 to apply. This is done by Dershowitz and Manna [1979, Example 3], who prove the termination of the Ackermann function by exhibiting an H-controlled ranking function into $\langle \mathbb{M}(\mathbb{N}^2), \leq_{\mathrm{m}} \rangle$, for which Theorem 3.5 yields an $O(H_{\omega^{\omega^2}}(\max(m, k)))$ complexity upper bound—where a more accurate bound would be $O(H_{\omega^m}(k))$. More generally, Ben-Amram [2002] showed that the programs that can be proven to terminate by size change abstraction are multiply recursive in the complexity of the basic operations.

3.4. Length Functions for Dickson's Lemma

Our second length function theorem pertains to the length of bad sequences over $\langle \mathbb{N}^d, \leq_\times \rangle$, which is relevant to both the disjunctive termination arguments from §2.2.2.2 and coverability in LCMs as seen in Example 2.7.

The starting point for the analysis is again the descent equation (3.16). However, we are no longer in the comfortable situation of ordinals, where we could work with the residual α/β directly as the ordinal β. In the case of $\mathbb{N}^d$, it is not immediately clear how to analyse the length $L_{g,\mathbb{N}^d/(n_1,\ldots,n_d)}(g(n))$ of controlled bad sequences in the residual of $\mathbb{N}^d$ by a tuple $(n_1, \ldots, n_d)$.

The key idea introduced in [C19] is to 'over-approximate' $\mathbb{N}^d/(n_1, \ldots, n_d)$ as a disjoint sum of tuples of lower dimension. Thus, instead of working just with

tuples in $\mathbb{N}^d$, we shall consider more generally *polynomial nwqos*, where disjoint sums are also allowed (see Section 3.4.1). Then, the notion of 'over-approximation' of residuals of polynomial nwqos is captured formally by showing the existence of a *normed reflection* into another polynomial nwqo (see Section 3.4.2). The final step lifts this analysis to ordinals through the *maximal order types* of polynomial nwqos, allowing to relate $L_{g,\mathbb{N}^d}(n)$ with functions in the Cichoń hierarchy (see Section 3.4.3). We follow the presentation from [C18], where the arguments were cleaned-up and generalised to obtain a length function theorem for Higman's Lemma.

3.4.1. Polynomial Normed wqos. We shall use the *empty nwqo* $\mathbf{0} \stackrel{\text{def}}{=} \emptyset$, and a *singleton nwqo* $\mathbf{1}$ with equality and constant norm 0. The exact element found in this singleton is actually irrelevant; it could be for instance a letter in an alphabet, or a configuration in a finite configuration set.

Definition 3.9. The set of *polynomial nwqos* is the smallest set of normed wqos containing $\mathbf{0}$, $\mathbf{1}$, and $\mathbb{N}$ and closed under the $\sqcup$ and $\times$ operations.

We shall work up to isomorphism: we write $A \equiv B$ when the two nwqos A and B are *isomorphic* structures. Let us stress that, in particular, norm functions must be preserved by nwqo isomorphisms. For all practical purposes, isomorphic nwqos can be identified; in particular, the length functions $L_{g,A}$ and $L_{g,B}$ are the same for isomorphic nwqos. Observe that the definitions are such that all the expected identities of $\sqcup$ and $\times$ hold up to isomorphism: the class of all normed wqos when considered up to isomorphism forms a *commutative semiring* with $\mathbf{0}$ as zero and $\mathbf{1}$ as one.

We write $A \cdot k$ for $\overbrace{A \sqcup \cdots \sqcup A}^{k \text{ times}}$; then, any finite normed wqo with k elements equipped with equality is isomorphic to $\mathbf{1} \cdot k$; also, $A \cdot 0 \equiv \mathbf{0}$ is the empty normed wqo.

We also write A^d for $\overbrace{A \times \cdots \times A}^{d \text{ times}}$ the d-fold Cartesian product of a normed wqo A with itself; in particular $A^0 \equiv \mathbf{1}$ is a singleton set containing only the empty tuple '()'.

Remark 3.10 (Polynomial Normal Form). Any polynomial normed wqo A can be put in a *polynomial normal form* (PNF)

$$A \equiv \mathbb{N}^{d_1} \sqcup \cdots \sqcup \mathbb{N}^{d_m} \tag{3.19}$$

for $m, d_1, \cdots, d_m \geq 0$. In the following we will deal exclusively with normed wqos in PNF; since $A \equiv A'$ implies $L_{g,A} = L_{g,A'}$ this will be at no loss of generality.

Example 3.11 (LCMs Configurations). The set of configurations $Q \times \mathbb{N}^{\mathrm{C}}$ of an LCM with $|Q| = p$ control states and $|\mathrm{C}| = d$ counters, along with its ordering and infinity norm, is isomorphic to the polynomial nwqo $\mathbb{N}^d \cdot p$ in PNF.

3.4.2. Reflecting Normed wqos. We may now tackle our main problem: computing residuals A/x. The descent equation (3.16), though it offers a way of computing the length function, quickly leads to complex expressions: the nwqos $A/x_0/x_1/\cdots/x_n$ become 'unstructured', i.e. have no nice definition in terms of $\sqcup$ and $\times$. As we are going to see, 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/x over the structure of the polynomial nwqo A.

Definition 3.12. A *nwqo reflection* is a mapping $r\colon A \to B$ between two nwqos that satisfies the two following properties:

$$\forall x, x' \in A : r(x) \leq_B r(x') \text{ implies } x \leq_A x' , \tag{3.20}$$

$$\forall x \in A : |r(x)|_B \leq |x|_A . \tag{3.21}$$

In other words, a nwqo reflection is an order reflection that is not norm-increasing.

We write $r\colon A \hookrightarrow B$ when r is a nwqo reflection and say that B *reflects* A. This induces a quasi-ordering between nwqos, written $A \hookrightarrow B$. Any nwqo reflects its induced substructures since $\mathrm{Id}\colon X \hookrightarrow A$ when X is a substructure of A. Thus $\mathbf{0} \hookrightarrow A$ for any A, and $\mathbf{1} \hookrightarrow A$ for any non-empty A. Remark that reflections are compatible with products and sums:

(3.22)
$$A \hookrightarrow A' \text{ and } B \hookrightarrow B' \text{ imply } A \sqcup B \hookrightarrow A' \sqcup B' \text{ and } A \times B \hookrightarrow A' \times B' .$$

Crucially, reflections preserve controlled bad sequences. Indeed, let $r\colon A \hookrightarrow B$, and consider a sequence $x_0, x_1, \ldots$ over A. Then by (3.20), $r(x_0), r(x_1), \ldots$ is bad when $x_0, x_1, \ldots$ is, and by (3.21), it is (g, n)-controlled when $x_0, x_1, \ldots$ is. Hence

$$A \hookrightarrow B \text{ implies } L_{g,A}(n) \leq L_{g,B}(n) \text{ for all } g, n . \tag{3.23}$$

3.4.2.1. *Inductive Reflection of Residuals.* The base cases of the inductive reflection of A/x over the structure of the polynomial nwqo A are

$$\mathbb{N}^0/() \hookrightarrow \mathbf{0} , \tag{3.24}$$

$$\mathbb{N}/k \hookrightarrow \mathbb{N}^0 \cdot k , \tag{3.25}$$

because $\mathbb{N}/k = [k]$ by the ordinal descent equation (3.18), and then $[k] \hookrightarrow \mathbb{N}^0 \cdot k$.

Regarding disjoint sums $A \sqcup B$, it is plain that

$$(A \sqcup B)/(x, A) = (A/x) \sqcup B , \qquad (A \sqcup B)/(y, B) = A \sqcup (B/y) , \tag{3.26}$$

and reflections are not required.

Turning to Cartesian products, the key insight is that any element (x', y') in $(A \times B)/(x, y)$ is by definition (3.15) such that $(x, y) \not\leq_\times (x', y')$, hence $x \not\leq_A x'$ or $y \not\leq_B y'$:

$$(A \times B)/(x, y) \hookrightarrow \big((A/x) \times B\big) \sqcup \big(A \times (B/y)\big) . \tag{3.27}$$

3.4.2.2. *Derivation Relation.* We finally combine the inductive residuation and reflection operations into a *derivation relation* ∂_n: intuitively, the relation $A\ \partial_n\ A'$ is included in the relation '$A/x \hookrightarrow A'$ for some $x \in A_{\leq n}$' (see Lemma 3.13 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 3.10), $\partial_n A$ is a *finite* set of

polynomial nwqos also in PNF, defined inductively by

$$\partial_n \mathbf{0} \stackrel{\text{def}}{=} \emptyset \,, \tag{3.28}$$

$$\partial_n \mathbb{N}^0 \stackrel{\text{def}}{=} \{\mathbf{0}\} \,, \tag{3.29}$$

$$\partial_n \mathbb{N}^d \stackrel{\text{def}}{=} \{\mathbb{N}^{d-1} \cdot nd\} \,, \tag{3.30}$$

$$\partial_n (A \sqcup B) \stackrel{\text{def}}{=} \big((\partial_n A) \sqcup B\big) \cup \big(A \sqcup (\partial_n B)\big) \,, \tag{3.31}$$

for $d > 0$ and A, B in PNF; in these definitions the $\sqcup$ operations are lifted to act upon nwqo sets S by $A \sqcup S \stackrel{\text{def}}{=} \{A \sqcup A' \mid A' \in S\}$ and symmetrically.

Lemma 3.13 (C18). *Let A be a polynomial nwqo in PNF and $x \in A_{\leq n}$ for some n. Then there exists A' in $\partial_n A$ such that $A/x \hookrightarrow A'$.*

3.4.3. Maximal Order Types. As it is more convenient to reason with ordinal arithmetic rather than with polynomial nwqos, we map polynomial nwqos $\langle A, \leq, |.|_A \rangle$ to ordinals in ω^ω using the *maximal order type* $o(A)$ of the underlying wqo $\langle A, \leq \rangle$ [de Jongh and Parikh, 1977]; see Appendix A.1.2 for details. The map o is a bijection (but not an order isomorphism) between polynomial (n)wqos and ω^ω:

$$o(\mathbf{0}) = 0 \,, \qquad o(\mathbf{1}) = 1 \,, \qquad o(\mathbb{N}) = \omega \,,$$
$$o(A \sqcup B) = o(A) \oplus o(B) \,, \qquad o(A \times B) = o(A) \otimes o(B) \,,$$

where '$\oplus$' and '$\otimes$' denote the natural sum and natural product on ordinals (see Appendix A.1.1.1). Thus, given a polynomial nwqo in PNF $A \equiv \bigsqcup_{i=1}^{m} \mathbb{N}^{d_i}$, its associated maximal order type is $o(A) = \bigoplus_{i=1}^{m} \omega^{d_i}$.

Let us accordingly lift the definition of ∂_n to ordinals in ω^ω, so that $o(A') \in \partial_n o(A)$ if and only if $A' \in \partial_n A$. We restate equations (3.29) to (3.31) using maximal order types: for all $\alpha > 0$ in ω^ω and all d, n in $\mathbb{N}$

$$\partial_n \alpha \stackrel{\text{def}}{=} \big\{\gamma \oplus \partial_n \omega^d \mid \alpha = \gamma \oplus \omega^d\big\} \,, \qquad \partial_n \omega^d \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } d = 0, \\ \omega^{d-1} \cdot (nd) & \text{otherwise.} \end{cases}$$

Observe that $\alpha' \in \partial_n \alpha$ implies $\alpha' < \alpha$, thus $\bigcup_n \partial_n$ is a well-founded relation. This leads to the definition of an over-approximation of the length function $L_{g,A}(n)$:

$$M_{g,\alpha}(n) \stackrel{\text{def}}{=} \max_{\alpha' \in \partial_n \alpha} \big\{1 + M_{g,\alpha'}(g(n))\big\} \,. \tag{3.32}$$

Proposition 3.14 (C18). *For any polynomial nwqo A, any control function g, and any initial control n,*

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

Proof. Either $A_{\leq n}$ is empty and then $L_{g,A}(n) = 0 \leq M_{g,o(A)}(n)$, or there exists some $x \in A_{\leq n}$ that maximises $L_{g,A/x}(g(n))$ in the descent equation (3.16):

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

By Lemma 3.13 there exists $A' \in \partial_n A$ such that $A/x \hookrightarrow A'$, thus by (3.23)

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

By well-founded induction on $A' \in \partial_n A$,

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

Thus by definition of M,

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

3.4.4. Upper Bounds. As $M_{g,\alpha}(n)$ is a subrecursive function, it only remains to compare it with more 'standard' functions like the Cichoń functions. We refer the reader to [L1] for the details of these finishing touches. To summarise those, we use the resemblance between $\partial_n\alpha$ and $P_{dn}(\alpha)$ to show that, for $\alpha < \omega^{d+1}$ [L1, Corollary 2.33]:

$$M_{g,\alpha}(n) \leq g_\alpha(n) \quad \text{assuming } (\gamma + \omega^{\beta+1})(x) \stackrel{\text{def}}{=} \gamma + \omega^\beta \cdot (dx+1)\ . \tag{3.33}$$

This extra twist of using an assignment of fundamental sequences different from the standard one of (3.8) can be avoided: (3.33) can be converted to the standard assignment of fundamental sequences at the price of some contortions, yielding [L1, Theorem 2.34]:

THEOREM 3.15 (Length Function Theorem for Polynomial NWQOs). *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 A is a polynomial nwqo with $o(A) < \omega^{d+1}$, then $L_{g,A}(n) \leq h_{o(A)}(nd)$.*

Setting $h(x) \stackrel{\text{def}}{=} g(x)d$ always satisfies the conditions of the theorem. There are however examples where setting $h \stackrel{\text{def}}{=} g$ suffices: e.g., $g(x) \stackrel{\text{def}}{=} 2x$, $g(x) \stackrel{\text{def}}{=} x^2$, or $g(x) \stackrel{\text{def}}{=} 2^x$; more generally, Theorem 3.15 can use $h \stackrel{\text{def}}{=} g$ whenever g is *super-homogeneous*, i.e. satisfies $g(dx) \geq g(x)d$ for all $d, x \geq 1$—then $L_{g,A}(n) \leq g_{o(A)}(nd)$.

EXAMPLE 3.16 (Coverability in LCMs). Consider an instance of the coverability problem for a LCM with $|Q| = p$ control states, $|\mathrm{C}| = d$ counters, and a target configuration $q, \boldsymbol{v}$ with $\max_{\mathrm{c}\in\mathrm{C}} \boldsymbol{v}(\mathrm{c}) = n$. As seen in Example 3.11, $o(Q \times \mathbb{N}^{\mathrm{C}}) = \omega^d \cdot p$.

Recall from §3.1.2.2 that we can use n as initial norm and $H(x) \stackrel{\text{def}}{=} x + 1$ as control function on the bad sequences extracted from the Ascending Chain Condition on Equation (2.9). Setting $h(x) \stackrel{\text{def}}{=} x + d = H^d(x)$ satisfies the conditions of Theorem 3.15. Hence the backward coverability computation on this instance converges after at most $h_{\omega^d \cdot p}(nd)$ steps. We will analyse the complexity of the algorithm itself in Section 4.1.3 in the next chapter.

EXAMPLE 3.17 (Disjunctive Termination Arguments into $\mathbb{N}^d$). Let us consider disjunctive termination arguments as in §2.2.2.2, where each of the d relations T_j has a ranking function f_j into $\mathbb{N}$, i.e. the argument defines a quasi-ranking function $f(s) \stackrel{\text{def}}{=} (f_1(s), \ldots, f_d(s))$ into $\langle \mathbb{N}^d, \leq_\times \rangle$. Assume f to be (g, ι)-controlled and let $h(x) \stackrel{\text{def}}{=} g(x) \cdot d$. By Theorem 3.15, the program will terminate in time $h_{\omega^d}(\iota(s_0)d)$ from configuration s_0.

For the program of Figure 2.1, the invariants in equations (2.4) and (2.5) are shown well-founded by $f_1(\ell_0, x, y, n) \stackrel{\text{def}}{=} x$ and $f_2(\ell_0, x, y, n) \stackrel{\text{def}}{=} y$ respectively. Then $g(x) \stackrel{\text{def}}{=} 2x$ and $\iota(\ell_0, x, y, n) \stackrel{\text{def}}{=} \max(x, y, n)$ can be used to control the sequence of ranks

$$(f_1(s_0), f_2(s_0)), (f_1(s_1), f_2(s_1)), \ldots$$

As g is super-homogeneous, the program terminates in time $g_{\omega^2}(d \cdot \max(x, y, n))$. Note that this is nearly the same bound as in Example 3.7, which used lexicographic ranking functions instead.

Remark 3.18 (Comparison with the Lexicographic Ordering). Examples 3.7 and 3.17 show that the bounds we obtain by applying Theorem 3.5 to lexicographic ranking functions into $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ and Theorem 3.15 to disjunctive termination arguments into $\langle \mathbb{N}^d, \leq_\times \rangle$ are quite similar. Furthermore, as noted by Blass and Gurevich [2008], attempting to differentiate the two approaches through their maximal order types is inconclusive, since ω^d is both the order type of $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ and the maximal order type of $\langle \mathbb{N}^d, \leq_\times \rangle$. As Theorem 3.5 gives an exact measure of the length function but Theorem 3.15 only provides an upper bound, one might wonder whether there is a difference between the two length functions.

It turns out that the two length functions are different: since $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle \hookrightarrow \langle \mathbb{N}^d, \leq_\times \rangle$, by (3.23) controlled bad sequences are at most as long in $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ as in $\langle \mathbb{N}^d, \leq_\times \rangle$. The following example taken from [C19, Remark 6.2] shows that they can be strictly shorter: the following sequence is a $(g, 1)$-controlled bad sequence over $\langle \mathbb{N}^2, \leq_\times \rangle$, which is good for $\langle \mathbb{N}^2, \leq_{\text{lex}} \rangle$, where $g(x) \stackrel{\text{def}}{=} x + 2$:

$$(1,1), (3,0), (2,0), (1,0), (0,9), (0,8), \ldots, (0,1), (0,0) \tag{3.34}$$

This sequence has length 14 whereas the maximal $(g, 1)$-controlled bad sequence for $\langle \mathbb{N}^2, \leq_{\text{lex}} \rangle$ is of length $g_{\omega^2}(1) = 8$:

$$(1,1), (1,0), (0,5), (0,4), \ldots, (0,1), (0,0)\ . \tag{3.35}$$

3.5. Related Work & Perspectives

The length function theorems we proved in this chapter are instances of a more general concern: whenever we prove the termination of a procedure using a wqo, we might also expect to gain some information about its complexity.

Put more abstractly, we seek to answer Kreisel's question 'What more than its truth do we know if we have a proof of a theorem in a given formal system?' in the particular setting of termination proofs and complexity analysis. This question can be understood in a strict sense, as requiring the termination proof to be carried in a proof system. For instance, Buchholz [1995] derives complexity bounds for term rewriting systems terminating using the multiset and lexicographic path orderings by showing that such termination proofs can be carried into 'small' fragments of Peano arithmetic, for which complexity bounds have long been known to exist [Kreisel, 1952]; Steila [2016] similarly analyses the Ramsey-based proof of termination by disjunctive termination arguments in several systems of arithmetic. More generally, the same question can be asked without necessarily referring to a precise proof system, e.g. for termination orderings [Buchholz, 1995; Hofbauer, 1992; Lepper, 2001; Weiermann, 1994, 1995], polynomial interpretations [Bonfante et al., 2001], dependency pairs [Hirokawa and Moser, 2008], size-change abstractions [Ben-Amram, 2002], abstract interpretation [Gulwani, 2009], disjunctive termination invariants [Steila, 2016], or ranking functions [Alias et al., 2010] to cite a few.

3.5.1. Length Functions for Ordinals. When focusing on ordinals below ε_0, Theorem 3.5 will probably not surprise anyone familiar with ordinal recursive functions. In particular, Proposition 3.4 is a particular case of a connection established more generally by Cichoń [1993] and Buchholz et al. [1994] between the definition of Hardy functions through fundamental sequences on the one hand, and through maximisation over ordinals of bounded norm on the other hand.

Nevertheless, the fact that we obtain an *exact* equality—not up to a primitive-recursive function—is worth mentioning, especially since the applications to program termination and coverability in LCMs require such a fine-grained statement if we wish to distinguish e.g. the d-dimensional case from the $d+1$-dimensional one.

Two specific cases of Theorem 3.5 are also considered in the literature: the lexicographic ordering $\langle \mathbb{N}^d, \leq_{\text{lex}} \rangle$ is analysed in [C19], and an upper bound for the multiset ordering over the lexicographic ordering $\langle \mathbb{M}(\mathbb{N}^d), \leq_{\text{m}} \rangle$ is shown by Abriola, Figueira, and Senno [2015].

3.5.2. Length Functions for Well-Quasi-Orders. The first upper bounds for bad sequences over $\langle \mathbb{N}^d, \leq_\times \rangle$ were proven by McAloon [1984] using an arguably complex combinatorial argument on *large intervals*. Clote [1986] later published a simpler proof based on an analysis by Ketonen and Solovay [1981], but with coarser bounds than McAloon's and no parametricity in the control function.[2] In the context of commutative algebra, Moreno Socías [1992] shows similar bounds for chains of polynomial ideals. Friedman [2001, Theorem 6.2] also shows that the length functions for fixed d are primitive-recursive. Since the publication of [C19], Abriola et al. [2015] have obtained bounds similar to those of Theorem 3.15 by relating directly the product ordering with the lexicographic ordering.

Regarding other wqos, Cichoń and Tahhan Bittar [1998] proved upper bounds for Higman's Lemma and arbitrary control functions. The presentation in Section 3.4 follows the framework of [C18], where tight bounds for Higman's Lemma are proven. The same framework has also been employed by Rosa-Velardo [2017] to derive a length function theorem for finite multisets of tuples in $\langle \mathbb{N}^d, \leq_\times \rangle$, where multisets are ordered by *multiset embedding*.

Weiermann [1994] proved a length function theorem for Kruskal's Tree Theorem when using affine control functions. Weiermann's result encompasses many wqos, and it might be interesting to develop specialised versions for simpler wqos and to allow arbitrary control functions. As Kruskal's Tree Theorem is seldom needed in its full generality, to my knowledge this very general result has never been applied in the WSTS literature: indeed, this literature rather focuses on bounded-depth trees [e.g., D'Osualdo et al., 2017; Genest et al., 2008; Meyer, 2008; Wies et al., 2010], which are within the reach of the techniques of this chapter using nested applications of Higman's Lemma [C18], with $\mathbf{F}_{\varepsilon_0}$ upper bounds [J5].

3.5.3. Further Applications. As wqos are pervasive in termination proofs, the upper bounds offered by length function theorems find applications in many fields. Besides the verification of infinite-state systems, the bounds in Sections 3.3 and 3.4 have been applied for instance to obtain small (inverse Ackermann) certificates in distributed decision systems [Fraigniaud et al., 2016], and sharp bounds for instance for the reverse mathematics of disjunctive termination arguments [Steila, 2016], for proof search in sequent calculi for substructural logics like relevance logic or contractive linear logic [Urquhart, 1999; J4], or for algorithms

[2] In terms of the extended Grzegorczyk hierarchy [Löb and Wainer, 1970; see also Section 4.1.1], Clote places his bound at level $\mathscr{F}_{d+6}$ and McAloon's at level $\mathscr{F}_{d+1}$ for successor-controlled bad sequences in fixed dimension d; Theorem 3.15 provides an upper bound $H^{\omega^d \cdot d}(dn)$ in $\mathscr{F}_d$ in this case.

relying on Hilbert's Basis Theorem in algebraic geometry [Benedikt et al., 2017; León Sánchez and Ovchinnikov, 2016]. This is only a small sample, as length function theorems have a surprisingly large number of applications, and rather than attempting to list them here, we refer the reader to the catalog of problems in [J3, Section 6].

3.5.4. Perspectives. A natural direction to extend the results presented in this chapter is to attempt to prove length function theorems for other wqos and to refine the existing bounds when possible.

Trees and Graphs. Among the little-known consequences of Weiermann's 1994 result, it might for instance be worthwhile re-visiting the case of binary trees (with ε_0 as maximal order type [Schmidt, 1979]) and of series-parallel partial orders (with the Schütte-Feferman ordinal Γ_0 as maximal order type [Pouzet and Sobrani, 2003]).

A contrario, one could attempt to go beyond Kruskal's Tree Theorem and to investigate more general wqos, like graph minors; here, computing the maximal order type is already an issue [Van der Meeren, 2015]. Note however that the WSTS literature seems to consider the graph minor ordering as too permissive [e.g. König and Stückrath, 2017], and rather considers the subgraph or induced subgraph orderings on restricted classes of graphs [Delzanno et al., 2010; König and Stückrath, 2017], for instance with bounded path lengths [Ding, 1992] or bounded shrub-depth [Ganian et al., 2012]—which can again be analysed using nested applications of Higman's Lemma.

Genericity. A systematic way of relating maximal order types with the length of controlled bad sequences is also missing. The work of Abriola et al. [2015] on $\mathbb{N}^d$, which analyses the combinatorics of controlled *reifications*—one of the main techniques for proving upper bounds on maximal order types—might be a first step in that direction.

Refinements. Finally, the length function theorems in this chapter all deal with amortised controlled sequences. It would be interesting to obtain tighter bounds for *strongly* controlled sequences. Recall for instance that the sequences arising in LCMs were in fact strongly controlled (see §3.1.2.2), and this could be a way of closing the complexity gap for their coverability problem in fixed dimension (see the next chapter).

CHAPTER 4

Fast-Growing Complexity

The previous chapter has shown how to derive complexity upper bounds for algorithms relying on wqos for their termination. For instance, for the backward coverability algorithm for LCMs, Example 3.16 extracted primitive-recursive upper bounds in terms of Cichoń functions when the number of counters was fixed, but only an upper bound of Ackermannian growth in general. We also saw with Example 3.1 that these bounds were rather tight.

However, when considering decision problems like Coverability rather than algorithms like Backward Coverability, the analysed algorithm itself might not be optimal. The standard tools to tackle such questions are complexity classes, along with the associated notions of reductions and completeness.

This chapter first summarises [J3] and proposes in Section 4.1 a suitable definition for the non-elementary complexities often encountered when working with wqos. As an application, I show in Section 4.1.3 how the length function theorem of the previous chapter entail that the Backward Coverability algorithm for LCMs yields an upper bound in 'ACKERMANN', a complexity class for decision problems of Ackermannian complexity.

The point of these fast-growing complexity classes is however to be able to prove completeness statements. I therefore present in Section 4.2 a proof that LCM Coverability is ACKERMANN-hard, meaning that the Backward Coverability algorithm is essentially optimal for this model.

This hardness proof follows essentially the proof using weak computations due to Schnoebelen [2010a]. The presentation in this chapter uses a clearer, tail recursive formulation using Hardy functions, which I introduced in joint publications with Haase, Haddad, and Schnoebelen [C16; L1; I4; J5]. This formulation is easier to adapt to more complex settings, and has been successfully employed for proving lower bounds for Coverability in several families of WSTS (c.f. Table 4.1 on page 49). Furthermore, I introduce here a variant of the Hardy functions, which allows to derive slightly tighter lower bounds in fixed dimension than the original hardness proofs of Urquhart [1999] and Schnoebelen [2002, 2010a].

Contents

4.1. Complexity Classes Beyond ELEMENTARY

We define in this section an ordinal-indexed hierarchy $(\mathbf{F}_\alpha)_{\alpha<\varepsilon_0}$ of complexity classes. We rely for this on the Hardy functions $(H^{\omega^\alpha})_{\alpha<\varepsilon_0}$ from Section 3.2.2 as a standard against which we can measure high complexities.

In logic and recursion theory, the functions $(H^{\omega^\alpha})_{\alpha<\varepsilon_0}$ are used to generate the extended Grzegorczyk hierarchy $(\mathscr{F}_\alpha)_{\alpha<\varepsilon_0}$ [Löb and Wainer, 1970], when closed under substitution and limited primitive recursion. We shall start by recalling the definition of this hierarchy and some of it main properties in Section 4.1.1. As we shall argue, the classes in the extended Grzegorczyk hierarchy are however not suitable for our complexity classification objectives.

We define instead in Section 4.1.2 another hierarchy $(\mathbf{F}_\alpha)_{\alpha<\varepsilon_0}$, where each $\mathbf{F}_\alpha$ class is the class of problems decidable within time bounded by a single application of H^{ω^α} composed with any function p already defined in the lower levels $\mathscr{F}_\beta$ for $\beta < \alpha$. We succinctly describe the main properties of these classes here, but the main argument in their favour is the catalogue of complete problems for several levels of the hierarchy, arising from the literature, and compiled in Section 6 of [J3].

4.1.1. The Extended Grzegorczyk Hierarchy is an ordinal-indexed infinite hierarchy of classes $(\mathscr{F}_\alpha)_{\alpha<\varepsilon_0}$ of functions with argument(s) and images in $\mathbb{N}$ and was defined by Löb and Wainer [1970]. It has multiple natural characterisations: for instance via `loop` programs for $\alpha < \omega$ [Meyer and Ritchie, 1967], via ordinal-recursive functions with bounded growth [Wainer, 1970], via functions computable with restricted resources as we will see in Equation (4.2), via functions that can be proven total in fragments of Peano arithmetic [Fairtlough and Wainer, 1998], etc.

4.1.1.1. *Computational Characterisation.* The extended Grzegorczyk hierarchy itself is defined by means of recursion schemes with the $(H^{\omega^\alpha})_\alpha$ as generators. Nevertheless, for $\alpha \geq 2$, each of its levels $\mathscr{F}_\alpha$ is also characterised as a class of functions computable with bounded resources [Wainer, 1970].

Formally, it is arguably more natural to first define a slight variant $(\mathscr{F}_{<\alpha})_\alpha$ of the extended Grzegorczyk hierarchy. Each class $\mathscr{F}_{<\alpha}$ for $\alpha > 2$ is the class

of functions computable by deterministic Turing machines in time bounded by $O(H^\gamma(n))$ for some ordinal $\gamma < \omega^\alpha$, when given an input of size n. Using standard notations for complexity classes (where 'F' denotes as usual a class of functions, 'D' that we use deterministic Turing machines, and 'TIME' that we consider time-bounded computations), this translates as

$$\mathscr{F}_{<\alpha} \stackrel{\text{def}}{=} \bigcup_{\gamma<\omega^\alpha} \mathsf{FDTIME}\left(H^\gamma(n)\right). \tag{4.1}$$

Note that the choice between deterministic and nondeterministic, or between time- and space-bounded computations in (4.1) is irrelevant, because $\alpha > 2$ and H^{ω^2} is already a function of exponential growth.

We find among others in this hierarchy $\mathscr{F}_{<3} = \mathsf{FELEMENTARY}$ the set of Kalmar-elementary functions, $\mathscr{F}_{<\omega} = \mathsf{FPRIMITIVE\text{-}RECURSIVE}$ the set of primitive-recursive functions, and $\mathscr{F}_{<\omega^\omega} = \mathsf{FMULTIPLY\text{-}RECURSIVE}$ the set of multiply-recursive functions.

4.1.1.2. *Main Properties.* The classes $\mathscr{F}_\alpha$ are then simply defined for $\alpha \geq 2$ as

$$\mathscr{F}_\alpha \stackrel{\text{def}}{=} \mathscr{F}_{<\alpha+1} = \bigcup_{\gamma<\omega^{\alpha+1}} \mathsf{FDTIME}\left(H^\gamma(n)\right). \tag{4.2}$$

This is equivalent to the computational characterisation of Wainer [1970]

$$\mathscr{F}_\alpha = \bigcup_{c<\omega} \mathsf{FDTIME}\left((H^{\omega^\alpha})^c(n)\right). \tag{4.3}$$

Indeed, by Equation (3.12), $H^{\omega^\alpha \cdot c}$ is the same as the cth iterate $(H^{\omega^\alpha})^c$ of H^{ω^α}, and $\omega^\alpha \cdot c < \omega^{\alpha+1}$. Conversely we can rely on the fact that, for each $\gamma < \omega^{\alpha+1}$, there exists $c < \omega$ such that $\gamma \leq \omega^\alpha \cdot c$, and then for all large enough n, $H^\gamma(n) \leq H^{\omega^\alpha \cdot c}(n)$ (c.f. Lemma A.1). Note that this also yields

$$\mathscr{F}_{<\alpha} = \bigcup_{\beta<\alpha} \mathscr{F}_\beta\ . \tag{4.4}$$

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

$$\mathscr{F}_\alpha = \bigcup_{f\in\mathscr{F}_\alpha} \mathsf{FDTIME}\left(f(n)\right)\ . \tag{4.5}$$

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

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

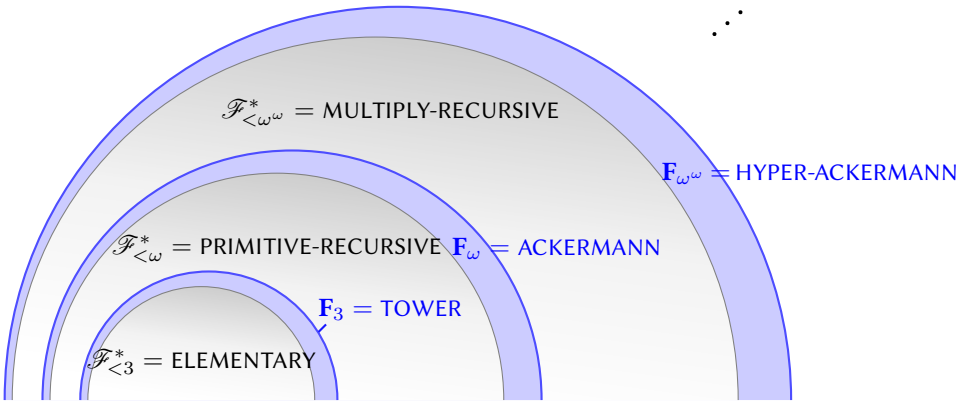


FIGURE 4.1. Some complexity classes beyond ELEMENTARY.

4.1.1.3. *Decision Problems.* We have been dealing this far with classes of functions, but writing $\mathscr{F}^*_\alpha$ for the restriction of $\mathscr{F}_\alpha$ to $\{0,1\}$-valued functions, i.e.

$$\mathscr{F}^*_\alpha = \bigcup_{\gamma<\omega^{\alpha+1}} \mathsf{DTIME}\left(H^\gamma(n)\right), \qquad \mathscr{F}^*_{<\alpha} \stackrel{\text{def}}{=} \bigcup_{\gamma<\omega^\alpha} \mathsf{DTIME}\left(H^\gamma(n)\right), \tag{4.6}$$

we obtain the corresponding classes of decision problems $\mathscr{F}^*_{<3}$ = ELEMENTARY, $\mathscr{F}^*_{<\omega}$ = PRIMITIVE-RECURSIVE, and $\mathscr{F}^*_{<\omega^\omega}$ = MULTIPLY-RECURSIVE.

4.1.2. The Fast-Growing Complexity Hierarchy. Unfortunately, the $\mathscr{F}^*_\alpha$ classes in the extended Grzegorczyk hierarchy are not suitable for many interesting decision problems, which are non elementary (or non primitive-recursive, or non multiply-recursive, etc.), but only *barely* so. The issue is that complexity classes like e.g. $\mathscr{F}^*_3$, which is the first class to contain non elementary problems, are very large: $\mathscr{F}^*_3$ contains for instance problems that require space $H^{\omega^3\cdot 100}(n)$, more than a hundred-fold compositions of towers of exponentials. As a result, hardness for $\mathscr{F}^*_3$ cannot be obtained for many classical examples of non elementary problems. In fact, as we shall see at the end of §4.1.2.2, there are *no* complete problems for $\mathscr{F}^*_\alpha$ under reasonable notions of reduction.

We therefore introduce *smaller* classes of decision problems for $3 \leq \alpha < \varepsilon_0$:

$$\mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{p\in\mathscr{F}_{<\alpha}} \mathsf{DTIME}\left(H^{\omega^\alpha}(p(n))\right) . \tag{4.7}$$

In contrast with $\mathscr{F}^*_\alpha$ in (4.6), only a single application of H^{ω^α} is possible, composed with some 'lower' *reduction* function p from $\mathscr{F}_{<\alpha}$. This definition is intended to be used with reductions from $\mathscr{F}_{<\alpha}$ in hardness or completeness statements.

As previously, because we assume $\alpha \geq 3$ and $\mathscr{F}_{<\alpha}$ contains all the elementary functions, we can derive the robustness of the $\mathbf{F}_\alpha$ classes under changes in the model of computation—e.g. RAM vs. Turing machines vs. Minsky machines, deterministic or nondeterministic or alternating—or the type of resources under consideration—time or space; e.g.

$$\mathbf{F}_\alpha = \bigcup_{p\in\mathscr{F}_{<\alpha}} \mathsf{NTIME}\left(H^{\omega^\alpha}(p(n))\right) = \bigcup_{p\in\mathscr{F}_{<\alpha}} \mathsf{SPACE}\left(H^{\omega^\alpha}(p(n))\right). \tag{4.8}$$

4.1.2.1. *Milestones.* The definition of the $(\mathbf{F}_\alpha)_\alpha$ complexity classes yields for instance a class

$$\mathsf{TOWER} \stackrel{\text{def}}{=} \mathbf{F}_3 = \bigcup_{p\in\mathsf{FELEMENTARY}} \mathsf{DTIME}\left(H^{\omega^3}(p(n))\right)$$

closed under elementary reductions (i.e., reductions from $\mathscr{F}_2$), a class

$$\mathsf{ACKERMANN} \stackrel{\text{def}}{=} \mathbf{F}_\omega = \bigcup_{p\in\mathsf{FPRIMITVE\text{-}RECURSIVE}} \mathsf{DTIME}\left(H^{\omega^\omega}(p(n))\right)$$

of Ackermannian problems closed under primitive-recursive reductions (i.e., reductions from $\mathscr{F}_{<\omega}$), and a class

$$\mathsf{HYPER\text{-}ACKERMANN} \stackrel{\text{def}}{=} \mathbf{F}_{\omega^\omega} = \bigcup_{p\in\mathsf{FMULTIPLY\text{-}RECURSIVE}} \mathsf{DTIME}\left(H^{\omega^{\omega^\omega}}(p(n))\right)$$

of hyper-Ackermannian problems closed under multiply-recursive reductions (i.e., reductions from $\mathscr{F}_{<\omega^\omega}$), etc. In each case, we can think of $\mathbf{F}_\alpha$ as the class of problems not solvable with resources in $\mathscr{F}_{<\alpha}$, but barely so: non elementary problems for $\mathbf{F}_3$, non primitive-recursive ones for $\mathbf{F}_\omega$, non multiply-recursive ones for $\mathbf{F}_{\omega^\omega}$, and so on. As a consequence of Corollary 4.2 below, the classes $(\mathbf{F}_\alpha)_\alpha$ also yield another definition of the primitive-recursive and multiply-recursive problems:

$$\mathsf{PRIMITIVE\text{-}RECURSIVE} = \mathscr{F}^*_{<\omega} = \bigcup_k \mathbf{F}_k \ ,$$
$$\mathsf{MULTIPLY\text{-}RECURSIVE} = \mathscr{F}^*_{<\omega^\omega} = \bigcup_k \mathbf{F}_{\omega^k} \ .$$

See Figure 4.1 for the first main stops of the hierarchy.

4.1.2.2. *Strictness.* For an ordinal $\alpha > 2$ and a finite k, let us define the class of decision problems

$$k\text{-}\mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{\gamma<\omega^\alpha\cdot(k+1)} \mathsf{DTIME}\big(H^\gamma(n)\big) \ . \tag{4.9}$$

By arguments similar to those sketched for (4.3), we see that

$$\mathbf{F}_\alpha = 1\text{-}\mathbf{F}_\alpha \ , \tag{4.10}$$

and the hierarchy $(k\text{-}\mathbf{F}_\alpha)_{k,\alpha}$ is thus a generalisation of the $(\mathbf{F}_\alpha)_\alpha$ one. The hierarchy $(k\text{-}\mathbf{F}_\alpha)_{k,\alpha}$ captures the decision problems in the extended Grzegorczyk hierarchy: by definition of $\mathscr{F}^*_\alpha$ and $\mathscr{F}^*_{<\alpha}$ in (4.6),

$$\mathscr{F}^*_\alpha = \bigcup_k k\text{-}\mathbf{F}_\alpha \ , \qquad\qquad \mathscr{F}^*_{<\alpha} = 0\text{-}\mathbf{F}_\alpha \ . \tag{4.11}$$

The $(k\text{-}\mathbf{F}_\alpha)_{k,\alpha}$ hierarchy is proven to be strict in [J3, Theorem 5.3] by a standard diagonalisation argument.

THEOREM 4.1 (Strictness). *For all k and $2 < \beta < \alpha$,*

$$k\text{-}\mathbf{F}_\beta \subsetneq (k+1)\text{-}\mathbf{F}_\beta \subsetneq \mathscr{F}^*_\beta \subsetneq \mathbf{F}_\alpha \ .$$

A consequence of Theorem 4.1 is that $(\mathbf{F}_\alpha)_\alpha$ 'catches up' with $(\mathscr{F}^*_\alpha)_\alpha$ at every limit ordinal.

Corollary 4.2. *Let λ be a limit ordinal, then*

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

Proof. The equality $\mathscr{F}^*_{<\lambda} = \bigcup_{\beta<\lambda} \mathbf{F}_\beta$ and the inclusion $\mathscr{F}^*_{<\lambda} \subseteq \mathbf{F}_\lambda$ can be checked by considering a problem in some $\mathscr{F}^*_\beta$ for $\beta < \lambda$: it is in k-$\mathbf{F}_\beta$ for some $k > 0$ by Equation (4.11), hence in $\mathbf{F}_{\beta+1}$ by Theorem 4.1, where $\beta + 1 < \lambda$ since λ is a limit ordinal and therefore the problem is in $\mathbf{F}_\lambda$, again by Theorem 4.1. The strictness of the inclusion follows from (4.11) and Theorem 4.1 with $k = 0$. □

Note that strictness implies that there are no '$\mathscr{F}^*_\alpha$-complete' problems under $\mathscr{F}_{<\alpha}$ reductions, since by Equation (4.11) such a problem would necessarily belong to some k-$\mathbf{F}_\alpha$ level, which would in turn entail the collapse of the $(k\text{-}\mathbf{F}_\alpha)_k$ hierarchy at the k-$\mathbf{F}_\alpha$ level and contradict Theorem 4.1.

Similarly, fix a limit ordinal λ and some reduction class $\mathscr{F}_\alpha$ for some $\alpha < \lambda$: there cannot be any meaningful '$\mathscr{F}^*_{<\lambda}$-complete' problem under $\mathscr{F}_\alpha$ reductions, since such a problem would be in $\mathscr{F}^*_\beta$ for some $\alpha < \beta < \lambda$, hence would contradict the strictness of the $(\mathscr{F}^*_\beta)_{\beta<\alpha}$ hierarchy. Just like the Time Hierarchy Theorem entails that there are no 'ELEMENTARY-complete' problems, there are no 'PRIMITIVE-RECURSIVE-complete' nor 'MULTIPLY-RECURSIVE-complete' problems either.

4.1.2.3. *Reduction Classes.* In order to be used together with reductions in $\mathscr{F}_{<\alpha}$, the classes $\mathbf{F}_\alpha$ need to be closed under such functions. Formally, for a function $f\colon \mathbb{N} \to \mathbb{N}$ and two languages A and B, we say that A many-one reduces to B in time $f(n)$, written $A \leq^f_m B$, if there exists a Turing transducer T working in deterministic time $f(n)$ such that, for all x, x is in A if and only if $T(x)$ is in B. For a class of functions $\mathcal{C}$, we write $A \leq^{\mathcal{C}}_m B$ if there exists f in $\mathcal{C}$ such that $A \leq^f_m B$. We write similarly that $A \leq^f_T B$ if there exists a Turing machine for A working in deterministic time $f(n)$ with oracle calls to B, and $A \leq^{\mathcal{C}}_T B$ if there exists f in $\mathcal{C}$ such that $A \leq^f_T B$.

As could be expected given the definitions, each class $\mathbf{F}_\alpha$ is closed both under many-one and Turing $\mathscr{F}_{<\alpha}$ reductions [J3, Theorems 4.7 and 4.8].

Theorem 4.3. *Let A and B be two languages with $B \in \mathbf{F}_\alpha$. If $A \leq^{\mathscr{F}_{<\alpha}}_m B$ or $A \leq^{\mathscr{F}_{<\alpha}}_T B$, then $A \in \mathbf{F}_\alpha$.*

Of course, we could replace in (4.7) the class of reductions $\mathscr{F}_{<\alpha}$ by a more traditional one, like logarithmic space (FL) or polynomial time (FP) functions. We feel however that our definition in (4.7) better captures the intuition we have of a problem being 'complete for H^{ω^α}.' Moreover, using at least $\mathscr{F}_2$ as our class of reductions allows to efficiently compute H^{ω^α} itself within $\mathbf{F}_\alpha$ bounds, leading to interesting combinatorial algorithms (see §4.1.3.2 for an example).

We always assume many-one $\mathscr{F}_{<\alpha}$ reductions when discussing hardness for $\mathbf{F}_\alpha$ in the remainder of this manuscript.

4.1.2.4. *Basic Complete Problem.* By (4.7), $\mathbf{F}_\alpha$-hardness proofs for many-one $\mathscr{F}_{<\alpha}$ reductions can reduce from the acceptance problem of some input string x by some deterministic Turing machine M working in time $H^{\omega^\alpha}(p(n))$ for some p in $\mathscr{F}_{<\alpha}$. This can be simplified to a machine M' working in time $H^{\omega^\alpha}(n)$ by a basic padding argument. Indeed, because p in $\mathscr{F}_{<\alpha}$ is honest, $p(n)$ can be computed in

$\mathscr{F}_{<\alpha}$. Thus the acceptance of x by M can be reduced to the acceptance problem of a #-padded input string $x' \stackrel{\text{def}}{=} x\#^{p(|x|)-|x|}$ of length $p(|x|)$ by a machine M' that simulates M, and treats # as a blank symbol—now M' works in time $H^{\omega^\alpha}(n)$.

To sum up, we have by definition of the $(\mathbf{F}_\alpha)_\alpha$ classes the following basic $\mathbf{F}_\alpha$-complete problem.

PROBLEM (Acceptance of H^{ω^α}-Bounded Turing Machines).

instance: A deterministic Turing machine M working in time H^{ω^α} and an input x.

question: Does M accept x?

This problem or slight variations of it has been used in most of the master reductions in the literature in order to prove non primitive-recursiveness, non multiple-recursiveness, and other hardness results [Chambart and Schnoebelen, 2008; Jančar, 2001; Lazić et al., 2016; Rosa-Velardo, 2017; Schnoebelen, 2010a; Urquhart, 1999; C16; J5]. Section 6 in [J3] presents a catalogue of 'natural' complete problems at various levels of the $(\mathbf{F}_\alpha)_\alpha$ hierarchy, which should be easier to employ in reductions.

4.1.2.5. *Robustness.* In the applications of fast-growing classes, one often relies on their 'robustness' to minor changes in their definition. More precisely, we can employ alternative generative functions; there are indeed many variants for the definition of the Hardy functions $(H^{\omega^\alpha})_\alpha$, but they are all known to generate essentially the same hierarchy $(\mathscr{F}_\alpha)_\alpha$.[1] The same holds for the hierarchy $(\mathbf{F}_\alpha)_\alpha$, as shown in Section 4 of [J3]; we shall see an example of this behaviour in §4.1.2.6.

The main rationale for this robustness is that each fast growing function class $\mathbf{F}_\alpha$ features enough 'wiggle room' to accommodate small computational overheads, where 'small' means 'in $\mathscr{F}_{<\alpha}$'. A possible formal translation of this intuition is proven in [J3, Lemma 4.6].

LEMMA 4.4. *Let f and f' be two functions in $\mathscr{F}_{<\alpha}$. Then there exists p in $\mathscr{F}_{<\alpha}$ such that $f \circ H^{\omega^\alpha} \circ f' \leq H^{\omega^\alpha} \circ p$.*

Lemma 4.4 justifies for instance Equation (4.8), since the overhead of working with different models of computation is elementary (hence in $\mathscr{F}_{<3}$) and α is always assumed to be at least 3.

Another application of Lemma 4.4 pertains to the computational resources required to compute H^{ω^α} itself. The functions H^{ω^α} are known to be *honest*, i.e. to be computable in time $\mathscr{F}_\alpha$ [Fairtlough and Wainer, 1998; Wainer, 1970]. This is however not tight enough to allow us to compute H^{ω^α} as part of an algorithm and retain an upper bound in $\mathbf{F}_\alpha$. We use the following refinement instead: let us call a function f *elementarily constructible* if there exists an elementary function e in $\mathsf{FELEMENTARY} = \mathscr{F}_{<3}$ such that $f(n)$ can be computed in time $e(f(n))$ for all n.

THEOREM 4.5 (J3). *Let $h\colon \mathbb{N} \to \mathbb{N}$ be an elementarily constructible strictly increasing function and α be an ordinal, then h^α is also elementarily constructible.*

Together with Lemma 4.4, Theorem 4.5 shows that algorithms can afford to compute H^γ for $\gamma < \omega^\alpha \cdot 2$ and still end with a complexity in $\mathbf{F}_\alpha$.

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

4.1.2.6. *Relativised Hierarchies.* One means of defining a variant of the fast-growing complexity classes is to pick a base function h different from $H(x) \stackrel{\text{def}}{=} x+1$ for the Hardy hierarchy. The corresponding relativised complexity classes are then defined by

$$\mathbf{F}_{h,\alpha} \stackrel{\text{def}}{=} \bigcup_{\gamma<\omega^\alpha\cdot 2} \mathsf{DTIME}\left(h^\gamma(n)\right). \tag{4.12}$$

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

The converse inclusion does not hold, since for instance h^ω is non elementary for $h(x) = 2^x$. Observe however that, in this instance, $h \leq H^{\omega^2}$ and $(H^{\omega^2})^\omega = H^{\omega^3}$ by Equation (3.12). This entails that $\mathbf{F}_{h,1} \subseteq \mathbf{F}_3$ when $h(x) = 2^x$. Thus, when working with relativised classes, one should somehow 'offset' the ordinal index by an appropriate amount. This idea is formalised in [J3, Theorem 4.2] and allows to show:

THEOREM 4.6. *Let $h\colon \mathbb{N} \to \mathbb{N}$ be a strictly increasing function and α, β be two ordinals.*

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

The statement of Theorem 4.6 is somewhat technical, but easy to apply to concrete situations; for instance:

COROLLARY 4.7. *Let $h\colon \mathbb{N} \to \mathbb{N}$ be a strictly increasing primitive recursive function and $\alpha \geq \omega$. Then $\mathbf{F}_{h,\alpha} = \mathbf{F}_\alpha$.*

PROOF. The function h is in $\mathscr{F}_k$ for some $k < \omega$, thus $\mathbf{F}_{h,\alpha} \subseteq \mathbf{F}_{k+1+\alpha} = \mathbf{F}_\alpha$ by Theorem 4.6. Conversely, since h is strictly increasing, $\mathbf{F}_\alpha \subseteq \mathbf{F}_{h,\alpha}$. □

4.1.3. Example: LCM Coverability is in ACKERMANN. As an application of the fast growing complexity classes, let us consider again the Coverability problem for lossy counter machines (c.f. Example 2.7). We have seen in Example 3.16 that the backward coverability algorithm terminates after at most $L \stackrel{\text{def}}{=} h_{\omega^d\cdot p}(nd)$ steps for an LCM with d counters and p states and a target configuration $q, \boldsymbol{v}$ with $\max_{\mathrm{c}\in\mathrm{C}} \boldsymbol{v}(\mathrm{c}) = n$, where $h(x) \stackrel{\text{def}}{=} x + d = H^d(x)$.

4.1.3.1. *Analysis of Backward Coverability.* At this point, we can analyse more closely the computational cost of each step of the backward algorithm. By Equation (3.14), each of the vectors in the minimal bases for U_i has norm at most

$$N \stackrel{\text{def}}{=} h^{\omega^d\cdot p}(nd) \leq H^{\omega^d\cdot dp}(nd) \leq H^{\omega^{d+1}}(ndp)\,, \tag{4.13}$$

and U_i contains therefore at most $p\cdot(N+1)^d$ minimal configurations. As seen in Example 2.7, the computation of the minimal elements of U_{i+1} can therefore be performed in time $e(N)$ for some elementary function $e \in \mathscr{F}_{<3}$. Hence the total time of the computation is bounded by $e(N)\cdot L$, which is an elementary function of N since $L \leq N$ by Equation (3.13). By Lemma 4.4, this is in $\mathbf{F}_{d+1}$ when the dimension $d = |\mathrm{C}|$ is fixed, and in $\mathbf{F}_\omega$ when d is part of the input.

THEOREM 4.8 (C19, Sec. VII-B). *LCM Coverability is in $\mathbf{F}_\omega$, and in $\mathbf{F}_{|\mathrm{C}|+1}$ if the number of counters is fixed.*

4.1.3.2. *A Combinatorial Algorithm.* Another way of obtaining the bound of Theorem 4.8 is to rely on the existence of pseudo-witnesses pointed out in Remark 2.6. Indeed, instead of the backward coverability algorithm, we can leverage pseudo-witnesses to obtain a non-deterministic forward algorithm. The argument is that a pseudo-witness must have length bounded by the same N defined just before in Equation (4.13). The algorithm therefore

(1) computes N in a first phase: as seen with Theorem 4.5, this can be performed in time (and space) $H^{\omega^{d+1}}(e(ndp))$ for some elementary function e,
(2) then nondeterministically explores the reachable configurations, starting from the initial configuration $q_0, \mathbf{0}$ and attempting to reach the target configuration $q, \boldsymbol{v}$—but aborts if the upper bound on the length is reached. This second phase uses at most N steps, and each step can be performed in space polynomial in the size of the current configuration and of an N-bounded counter, thus in $O(d \log(N+1) + \log p)$. The whole phase can thus be performed using space polynomial in N, which is bounded by $H^{\omega^{d+1}}(f(ndp))$ for some f in $\mathscr{F}_d$ by Lemma 4.4.

This algorithm thus works in space $H^{\omega^{d+1}}(f(ndp))$ for some f in $\mathscr{F}_d$, which by Equation (4.8) yields again Theorem 4.8.

4.2. Lower Bounds Through Hardy Computations

The main purpose of the fast-growing complexity classes $(\mathbf{F}_\alpha)_\alpha$ is to provide the means to prove completeness statements. As an illustration, first of the fact that such high complexities can arise naturally in computer science, and second of how such lower bounds can be proven, we show here that Coverability is ACKERMANN-hard in lossy counter machines, thus matching the upper bound of Theorem 4.8.

THEOREM 4.9. *LCM Coverability is $\mathbf{F}_\omega$-hard, and $\mathbf{F}_{|\mathrm{C}|}$-hard if the number of counters $|\mathrm{C}| \geq 3$ is fixed.*

This result was first shown by Urquhart [1999] for a related model of expansive alternating counter machines, and independently by Schnoebelen [2002] for lossy counter machines. The proof in this section follows the same general principle of implementing weak computers for fast-growing functions and their inverses (see Section 4.2.2), but works in the arguably more elegant setting of *Hardy computations* developed in [C16; L1; I4; J5]. Furthermore, we employ a variant of the Hardy functions $(H^\alpha)_\alpha$ defined Section 4.2.1, which allows to slightly improve over the $\mathbf{F}_{|\mathrm{C}|-2}$ lower bound proven by Schnoebelen [2010a] when the number of counters is fixed.

REMARK 4.10 (Reset VASS). As explained by Schnoebelen [2010a], the bounds of Theorem 4.9 apply directly to *reset vector addition systems* with states, which are Minsky machines where the zero test operation is replaced by a reset operation that assigns zero to the counter; this is true even with a fixed number of counters. The $\mathbf{F}_\omega$ lower bound holds more generally for *transfer* vector addition systems [Schnoebelen, 2010a] and broadcast protocols [I4].

4.2.1. Hardy-like Computations. We start with a variant of the Hardy hierarchy, that differs from the definition of $(H^\alpha)_\alpha$ in (3.10) on several counts:

- in successor cases, the argument is incremented by two instead of one, and
- in limit cases, the argument is reset to 2 and the index descends slightly faster through the fundamental sequence, with an update of ω to $x-1$ instead of $x+1$.

We define the a^α function for $\alpha < \omega^\omega$ and $x > 0$ by

$$a^0(x) \stackrel{\text{def}}{=} x\ , \qquad a^{\alpha+1}(x) \stackrel{\text{def}}{=} a^\alpha(x+2)\ , \qquad a^\lambda(x) \stackrel{\text{def}}{=} a^{\lambda(x-2)}(2)\ . \tag{4.14}$$

The last two cases of (4.14) read left-to-right define a rewriting system over pairs (α, n) where $0 \leq \alpha < \omega^\omega$ is an ordinal and $n > 0$ is a natural number:

$$(\alpha+1, n) \to (\alpha, n+2) \tag{4.15}$$

$$(\lambda, n) \to (\lambda(n-2), 2) \tag{4.16}$$

We call a rewriting sequence $(\alpha_0, n_0) \to (\alpha_1, n_1) \to \cdots \to (\alpha_\ell, n_\ell)$ a *Hardy-like computation*. It enforces the invariant $a^{\alpha_i}(n_i) = a^{\alpha_0}(n_0)$ for all i. Observe that $\alpha_0 > \alpha_1 > \cdots > \alpha_\ell$, thus such a computation must terminate, and that if $\alpha_\ell = 0$ (in which case we say that the computation is *complete*), then $n_\ell = a^{\alpha_0}(n_0)$.

4.2.1.1. *Basic Properties.* The point of using superscripts for the ordinal indices in the $(a^\alpha)_\alpha$ hierarchy is that these functions satisfy the same identity for composition as the Hardy hierarchy (c.f. Equation (3.12)).

Lemma 4.11 (Composition). *Let $\alpha + \beta < \omega^\omega$ be an ordinal term in Cantor normal form and $x > 0$. Then $a^\alpha\big(a^\beta(x)\big) = a^{\alpha+\beta}(x)$.*

Proof. By transfinite induction over β. For the zero case, $a^\alpha(a^0(x)) = a^\alpha(x)$. For the successor case, $a^\alpha\big(a^{\beta+1}(x)\big) = a^\alpha\big(a^\beta(x+2)\big) = a^{\alpha+\beta}(x+2) = a^{\alpha+\beta+1}(x)$. For the limit case, $a^\alpha\big(a^\lambda(x)\big) = a^\alpha\big(a^{\lambda(x-2)}(2)\big) = a^{\alpha+\lambda(x-2)}(2) = a^{\alpha+\lambda}(x)$. □

They also enjoy similar monotonicity properties.

Lemma 4.12 (Inflationary). *For all $0 < x$ and $0 < \alpha < \omega^\omega$, $x < a^\alpha(x)$.*

Proof. By transfinite induction over α. For the base case, $a^1(x) = x+2 > x$. For the successor case, $a^{\alpha+1}(x) = a^\alpha(x+2) > x+2 \geq x$ by induction hypothesis. For the limit case $a^\lambda(x) = a^{\lambda(x-2)}(x) > x$ by induction hypothesis. □

Lemma 4.13 (Monotone). *For all $0 < x < y$ and $0 \leq \alpha < \omega^\omega$, $a^\alpha(x) < a^\alpha(y)$.*

Proof. By transfinite induction over α. For the base case, $a^0(y) = y > x = a^0(x)$. For the successor case, $a^{\alpha+1}(y) = a^\alpha(y+2) > a^\alpha(x+2) = a^{\alpha+1}(x)$ by induction hypothesis. For the limit case, define $z \stackrel{\text{def}}{=} y - x > 0$ and write the limit ordinal λ as $\gamma + \omega^{d+1}$ for some d and $\gamma \geq \omega^{d+1}$; then $a^\lambda(y) = a^{\lambda(y-2)}(2) = a^{\gamma+\omega^d\cdot(y-1)}(2) = a^{\gamma+\omega^d\cdot(x-1)+\omega^d\cdot z}(2) = a^{\gamma+\omega^d\cdot(x-1)}\big(a^{\omega^d\cdot z}(2)\big) > a^{\gamma+\omega^d\cdot(x-1)}(2) = a^\lambda(x)$, where we applied Lemma 4.11, followed by Lemma 4.12 to show that $a^{\omega^d\cdot z}$ is strictly inflationary and the induction hypothesis to show that $a^{\gamma+\omega^d\cdot(x-1)}$ is strictly monotone. □

4.2.1.2. *Connection with the Ackermann Hierarchy.* The hierarchy $(a^\alpha)_\alpha$ yields in the case of the ordinal indices $(\omega^d)_{d>0}$ a hierarchy of 'hyper-operations' that appears quite often in the literature, with multiplication by two appearing as the base case for $d = 1$ and the higher levels defined through iteration.

LEMMA 4.14 (Ackermann Hierarchy). *For all $x > 0$, $a^\omega(x) = 2x$, and for all $x, d > 0$, $a^{\omega^{d+1}}(x) = (a^{\omega^d})^x(1)$.*

PROOF. Indeed, regarding the first equation,

$$a^\omega(x) = a^{x-1}(2) = 2 + 2(x-1) = 2x \ .$$

Let us also note that, for all $d > 0$, $a^{\omega^d}(1) = 2$; indeed by definition (4.14),

$$a^{\omega^{d+1}}(1) = a^{\omega^d \cdot 0}(2) = a^0(2) = 2 \ .$$

Thus, regarding the second equation, and using Lemma 4.11

$$a^{\omega^{d+1}}(x) = a^{\omega^d \cdot (x-1)}(2) = a^{\omega^d \cdot (x-1)}\big(a^{\omega^d}(1)\big) = (a^{\omega^d})^x(1) \ . \qquad \square$$

The definition in (4.14) thus results for instance in $a^{\omega^2}(x) = 2^x$ and $a^{\omega^3}(x) = \text{tower}(x)$, and is typically used in lower bound proofs. Nevertheless, the definition in (4.14) offers a *tail-recursive* viewpoint instead of an iterative viewpoint, which makes it easier to handle for our proofs.

We can define a hierarchy of classes of decision problems generated by the $(a^{\omega^d})_{d>2}$ functions by analogy with (4.7):

$$\text{(4.17)} \quad \mathbf{A}_d \stackrel{\text{def}}{=} \bigcup_{p \in \mathscr{F}_{<d}} \mathsf{DTIME}\left(a^{\omega^d}(p(n))\right), \quad \mathbf{A}_\omega \stackrel{\text{def}}{=} \bigcup_{p \in \mathscr{F}_{<\omega}} \mathsf{DTIME}\left(a^{\omega^{p(n)+1}}(p(n))\right).$$

It turns out that this defines the same class of problems as $\mathbf{F}_d$, and thus that the halting problem of a^{ω^d}-bounded Minsky machines suffices to prove $\mathbf{F}_d$-hardness.

THEOREM 4.15 (J3, Theorem 4.1). *For all $2 < \alpha \leq \omega$, $\mathbf{A}_\alpha = \mathbf{F}_\alpha$.*

Therefore, using a padding argument as in §4.1.2.4, a basic $\mathbf{F}_d$-complete problem is the halting problem for Minsky machines where the sum of the counter values is bounded by $a^{\omega^d}(n)$ along the execution. Two counters suffice here, as the halting problem of 2-counters Minsky machines with counters bounded by $2^{2^{s(n)}}$ is hard for $\mathsf{SPACE}(s(n))$ [Fischer et al., 1968], and the double exponential can be dealt with padding as we always assume $d \geq 3$.

PROBLEM (Halting of a^{ω^d}-Bounded Minsky Machines).
instance: A deterministic 2-counters Minsky machine M where the sum of counters is bounded by $a^{\omega^d}(|M|)$.
question: Does M halt?

Finally, the version of the problem with $d = |M| + 1$ is $\mathbf{F}_\omega$-complete.

4.2.2. Weak Computation in Lossy Counter Machines. Let us fix some $d > 0$. We are going to implement Hardy-like computations and their inverses in Minsky machines, using an encoding of ordinals below ω^d in a subset of their counters. The implementations in these Minsky machines will be perfect, in that they will compute a^{ω^d} and its inverse exactly. When the same machines are run as lossy counter machines, we will see that the lossy behaviour can only yield smaller or equal results: the implementations are *weak computers* for the functions.

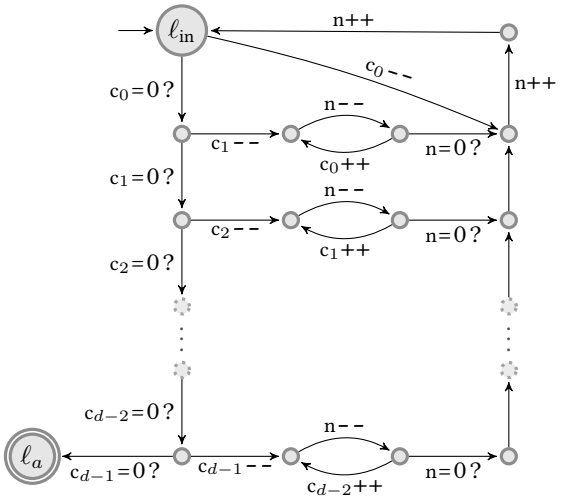


FIGURE 4.2. A Minsky machine M_a implementing direct Hardy-like computations as in equations (4.19) and (4.20).

Codes. We construct Minsky machines with counters $\mathrm{C}_d \stackrel{\text{def}}{=} \{\mathrm{c}_{d-1}, \ldots, \mathrm{c}_0, \mathrm{n}\}$. A valuation $\boldsymbol{v} = (c_{d-1}, \ldots, c_0, n)$ in $\mathbb{N}^{\mathrm{C}_d}$ encodes a pair (α, n) from $\omega^d \times \omega$ with

$$\alpha(\boldsymbol{v}) \stackrel{\text{def}}{=} \omega^{d-1} \cdot c_{d-1} + \cdots + \omega^0 \cdot c_0 \ . \tag{4.18}$$

Direct Computations. We translate the rewriting rules of (4.15) in terms of valuations $(c_{d-1}, \ldots, c_0, n)$ in $\mathbb{N}^{\mathrm{C}_d}$, and unwrap the definition of the assignment of fundamental sequences from Equation (3.8):

$$(c_{d-1}, \ldots, c_0 + 1, n) \to_a (c_{d-1}, \ldots, c_0, n+2) \tag{4.19}$$

$$(c_{d-1}, \ldots, c_j + 1, 0, 0, \ldots, 0, n) \to_a (c_{d-1}, \ldots, c_j, n-1, 0, \ldots, 0, 2) \tag{4.20}$$

for $0 < j \leq d-1$. These rules are straightforward to implement in a Minsky machine M_a shown in Figure 4.2. Starting in ℓ_{in} with valuation $\boldsymbol{v} = (c_{d-1}, \ldots, c_0, n)$ with $n > 0$, M_a reaches ℓ_a exactly when it has performed the complete Hardy-like computation starting from the pair $(\alpha(\boldsymbol{v}), n)$, and thus with a final resulting valuation $(0, \ldots, 0, a^{\alpha(\boldsymbol{v})}(n))$.

Inverse Computations. We also need to translate the inverse rewriting rules of (4.15) in terms of valuations $(c_{d-1}, \ldots, c_0, n)$ in $\mathbb{N}^{\mathrm{C}_d}$. This now yields the rules

$$(c_{d-1}, \ldots, c_0, n+2) \to_{a^{-1}} (c_{d-1}, \ldots, c_0 + 1, n) \tag{4.21}$$

$$(c_{d-1}, \ldots, c_j, n, 0, \ldots, 0, 2) \to_{a^{-1}} (c_{d-1}, \ldots, c_j + 1, 0, 0, \ldots, 0, n+1) \tag{4.22}$$

for $0 < j \leq d-1$. Note that the inverse rules are not deterministic, as for a given N there exist many pairs (α, n) such that $a^\alpha(n) = N$. However, for each fixed α, a^α is injective since it is strictly monotone by Lemma 4.13.

The rules are again implemented in a Minsky machine $M_{a^{-1}}$ depicted in Figure 4.3. Starting in $\ell_{a^{-1}}$ with valuation $\boldsymbol{v} = (c_{d-1}, \ldots, c_0, n)$ with $n > 0$, $M_{a^{-1}}$ performs inverse Hardy-like computation steps, and when reaching ℓ_{out} must have found a valuation $(c'_{d-1}, 0, \ldots, 0, n')$ such that $a^{\omega^{d-1} \cdot c'_{d-1}}(n') = a^{\alpha(\boldsymbol{v})}(n)$.

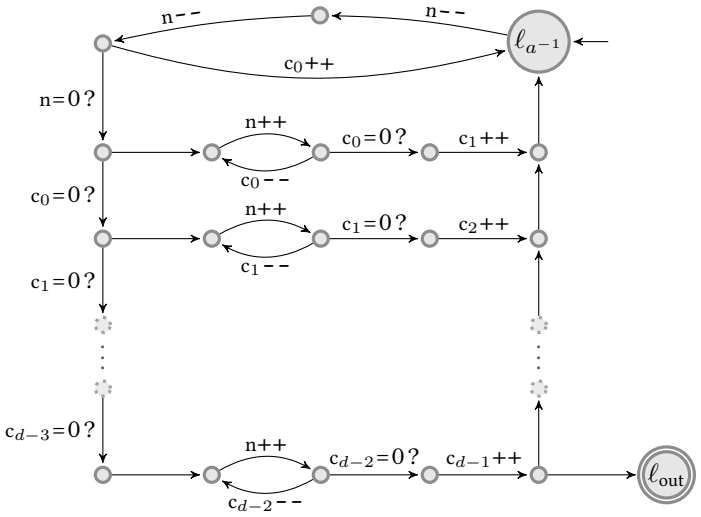


FIGURE 4.3. A Minsky machine $M_{a^{-1}}$ implementing inverse Hardy-like computations as in equations (4.21) and (4.22).

Robustness. When running the two machines M_a and $M_{a^{-1}}$ of Figures 4.2 and 4.3 with lossy semantics, we need to ensure that, if two valuations $\boldsymbol{v}$ and $\boldsymbol{v}'$ from $\mathbb{N}^{\mathsf{C}_d}$ are such that $\boldsymbol{v} \leq \boldsymbol{v}'$ (for the product ordering), then

$$a^{\alpha(\boldsymbol{v})}(\boldsymbol{v}(\mathrm{n})) \leq a^{\alpha(\boldsymbol{v}')}(\boldsymbol{v}'(\mathrm{n}))\ . \tag{4.23}$$

By Lemmas 4.11 to 4.13, (4.23) holds. We say that our codes are *robust*; they ensure that losses can only yield results smaller than the perfect results we would have obtained with reliable semantics.

CLAIM 4.16 (Weak Hardy-like Computations). In an LCM, the systems M_a and $M_{a^{-1}}$ are such that, for all valuations $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4$ in $\mathbb{N}^{\mathsf{C}_d}$,

(1) if $(\ell_{\text{in}}, \boldsymbol{v}_1) \to_\ell^* (\ell_a, \boldsymbol{v}_2)$, then $a^{\alpha(\boldsymbol{v}_1)}(\boldsymbol{v}_1(\mathrm{n})) \geq \boldsymbol{v}_2(\mathrm{n})$ and $\boldsymbol{v}_2(c_j) = 0$ for all $0 \leq j < d$;
(2) if $a^{\alpha(\boldsymbol{v}_1)}(\boldsymbol{v}_1(\mathrm{n})) = \boldsymbol{v}_2(\mathrm{n})$ with $\boldsymbol{v}_2(c_j) = 0$ for all $0 \leq j < d$, then an execution $(\ell_{\text{in}}, \boldsymbol{v}_1) \to_\ell^* (\ell_a, \boldsymbol{v}_2)$ is possible;
(3) if $(\ell_{a^{-1}}, \boldsymbol{v}_3) \to_\ell^* (\ell_{\text{out}}, \boldsymbol{v}_4)$, then $\boldsymbol{v}_4(c_j) = 0$ for all $0 \leq j < d-1$ and $a^{\alpha(\boldsymbol{v}_3)}(\boldsymbol{v}_3(\mathrm{n})) \geq a^{\alpha(\boldsymbol{v}_4)}(\boldsymbol{v}_4(\mathrm{n}))$;
(4) if $(\alpha(\boldsymbol{v}_4), \boldsymbol{v}_4(\mathrm{n})) \to^* (\alpha(\boldsymbol{v}_3), \boldsymbol{v}_3(\mathrm{n}))$ is a Hardy-like computation with $\boldsymbol{v}_4(\mathrm{c}_j) = 0$ for all $0 \leq j < d-1$, then an execution $(\ell_{a^{-1}}, \boldsymbol{v}_3) \to_\ell^* (\ell_{\text{out}}, \boldsymbol{v}_4)$ is possible.

4.2.3. Lower Bound. We prove now Theorem 4.9 for a number of counters $d = |\mathsf{C}| > 2$. We start from an instance $M = \langle Q, \{\mathrm{c}_0, \mathrm{c}_1\}, \delta, \ell_0 \rangle$ of the halting problem for a^{ω^d}-bounded Minsky machines, where we further assume the counters are initially valued to 0, and are zero-tested right before reaching the halting state ℓ_h.

Counter Machine on a Budget. We modify M by adding a new counter n and by splitting transitions (introducing new intermediary states) so that any increment c_i++ for $i \in \{0, 1\}$ is preceded by a decrement n−−, and any decrement c_i−− for $i \in \{0, 1\}$ is followed by an increment n++. The resulting machine M' is thus

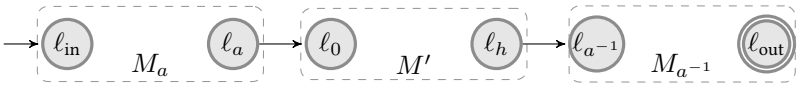


FIGURE 4.4. The LCM M_d constructed in the proof of Theorem 4.9.

put *on a budget*; its sum of counters is bounded by the initial value of n, and any losses along a halting execution will result in a smaller valuation for n upon reaching q_h.

CLAIM 4.17 (Minsky Machine on a Budget). For all N, N', c_0, c_1 in $\mathbb{N}$,

(1) if $(\ell_0, 0, 0, N) \to_\ell^* (\ell_h, c_0, c_1, N')$ in M', then $c_0 = c_1 = 0$ and $N' \leq N$;
(2) if $(\ell_0, 0, 0) \to^* (\ell_h, 0, 0)$ in M and $N \geq a^{\omega^d}(|M|)$, then $(\ell_0, 0, 0, N) \to_\ell^* (\ell_h, 0, 0, N)$ in M';
(3) if $(\ell_0, 0, 0, N) \to_\ell^* (\ell_h, 0, 0, N)$ in M', then $(\ell_0, 0, 0) \to^* (\ell_h, 0, 0)$ in M.

Reduction. We finally plug M' with M_a as an initialisation phase and $M_{a^{-1}}$ as a finalisation phase, resulting in the machine M_d depicted in Figure 4.4. This machine uses $d + 1$ counters, but we will see later that we can remove one of its counters.

We define the source configuration for our instance of the Coverability problem to be $(\ell_{\text{in}}, \boldsymbol{v})$ where $\boldsymbol{v}(\mathrm{c}_{d-1}) \stackrel{\text{def}}{=} |M| - 1$, $\boldsymbol{v}(\mathrm{c}_j) \stackrel{\text{def}}{=} 0$ for $0 \leq j < d - 1$, and $\boldsymbol{v}(\mathrm{n}) \stackrel{\text{def}}{=} 2$. This satisfies $\alpha(\boldsymbol{v}) = \omega^{d-1} \cdot (|M| - 1)$, hence $a^{\omega^d}(|M|) = a^{\alpha(\boldsymbol{v})}(\boldsymbol{v}(\mathrm{n}))$. We define the target configuration for our instance to be $(\ell_{\text{out}}, \boldsymbol{v})$. The LCM for our instance is M_d.

Let us first check the correctness of our reduction. First assume that M halts, i.e. that it reaches $(\ell_h, 0, 0)$ from $(\ell_0, 0, 0)$ with sum of counters bounded by $a^{\omega^d}(|M|)$. Then by Claim 4.16.(2), there is a (reliable) execution in M_d from the source $(\ell_{\text{in}}, |M| - 1, 0, \ldots, 0, 2)$ to $(\ell_a, 0, 0, \ldots, 0, a^{\omega^d}(|M|))$. Since this budget is sufficient by hypothesis on M, by Claim 4.17.(2) there is a (reliable) execution of M' that reaches the configuration $(\ell_{a^{-1}}, 0, 0, \ldots, 0, a^{\omega^d}(|M|))$. By Claim 4.16.(4), there is a (reliable) execution in $M_{a^{-1}}$ to the target $(\ell_{\text{out}}, |M| - 1, 0, \ldots, 0, 2)$, which is therefore covered.

Conversely, assume the target is covered from the source in M_d, i.e. that $(\ell_{\text{in}}, \boldsymbol{v}) \to_\ell^* (\ell_{\text{out}}, \boldsymbol{v}''')$ with $\boldsymbol{v}''' \geq \boldsymbol{v}$. By Claim 4.16.(3) and the robustness of our encoding (4.23), this means that we entered $M_{a^{-1}}$ in a configuration $(\ell_{a^{-1}}, \boldsymbol{v}'')$ such that $a^{\alpha(\boldsymbol{v}'')}(\boldsymbol{v}''(\mathrm{n})) \geq a^{\omega^d}(|M|)$. By Claim 4.17.(1), this implies that $\boldsymbol{v}''(\mathrm{c}_j) = 0$ for all $0 \leq j < d$ and that we entered M' with a configuration $(\ell_a, \boldsymbol{v}')$ with $a^{\alpha(\boldsymbol{v}')}(\boldsymbol{v}'(\mathrm{n})) \geq a^{\alpha(\boldsymbol{v}'')}(\boldsymbol{v}''(\mathrm{n}))$. By Claim 4.16.(1), this entails that $\boldsymbol{v}'(\mathrm{c}_j) = 0$ for all $0 \leq j < d$ and that

$$a^{\omega^d}(|M|) \geq a^{\alpha(\boldsymbol{v}')}(\boldsymbol{v}'(\mathrm{n})) \geq a^{\alpha(\boldsymbol{v}'')}(\boldsymbol{v}''(\mathrm{n})) \geq a^{\omega^d}(|M|)\,. \tag{4.24}$$

Thus M' had a budget of $\boldsymbol{v}'(\mathrm{n}) = \boldsymbol{v}''(\mathrm{n}) = a^{\omega^d}(|M|)$ at both the start and the end of its execution, which therefore had to be reliable: by Claim 4.17.(3), $(\ell_0, 0, 0)$ can reach $(\ell_h, 0, 0)$ in M.

The construction of M_d can eschew the use of c_{d-1} by maintaining its value as part of its state space. Indeed, in any reliable execution that covers the target, this counter value must remain less than $|M|$. This shows the $\mathbf{F}_d$-hardness of LCM Coverability for $|\mathrm{C}| = d > 2$.

TABLE 4.1. The complexity of Coverability in a few families of WSTS.

WSTS	Complexity	Reference for lower bound
Lossy Counter Machines	$\mathbf{F}_\omega$-complete	[Schnoebelen, 2002, 2010a; Urquhart, 1999; Th. 4.9]
Lossy Channel Systems	$\mathbf{F}_{\omega^\omega}$-complete	[Chambart and Schnoebelen, 2008; C14]
Unordered Data nets	$\mathbf{F}_{\omega^\omega}$-complete	[Rosa-Velardo, 2017]
Ordered Data nets	$\mathbf{F}_{\omega^{\omega^\omega}}$-complete	[C16]
Timed-arc Petri nets	$\mathbf{F}_{\omega^{\omega^\omega}}$-complete	[C16]
Nested Counter Systems	$\mathbf{F}_{\varepsilon_0}$-complete	[Decker and Thoma, 2016]
Priority Channel Systems	$\mathbf{F}_{\varepsilon_0}$-complete	[J5]

Finally, when the number of counters is not fixed, we can reduce from the halting problem for $a^{\omega^{|M|+1}}$-bounded Minsky machines; it suffices to set $d \stackrel{\text{def}}{=} |M|+1$ in the reduction above. This shows the $\mathbf{F}_\omega$-hardness of coverability and concludes the proof of Theorem 4.9.

4.3. Related Work & Perspectives

Together with the upper bounds provided by length function theorems such as the ones presented in Chapter 3, we have with the complexity classes $(\mathbf{F}_\alpha)_\alpha$ from Section 4.1 and the template for proving lower bounds using Hardy computations from Section 4.2 a complete toolbox for pinpointing the exact complexity of many decision problems. Table 4.1 presents the results obtained using this toolbox on the coverability problem for several classes of WSTS.

I would however like to emphasise that many more problems have now been successfully classified with the $(\mathbf{F}_\alpha)_\alpha$ classes, which can be found in the catalogue of problems in [J3, Section 6]. While these problems all rely at some point on a wqo for their decidability, many do not explicitly involve WSTS nor coverability at all. In most cases, the lower bounds were obtained through reductions from coverability problems rather than going through the pains of a direct reduction from a halting problem as in Section 4.2.

4.3.1. Lower Bounds Through Counter Objects. The hardness proof of Section 4.2 using weak implementations of Hardy computations and their inverses is not the only means to prove hardness statements for WSTS Coverability. In several classes of WSTS, weak computers for inverse Hardy computations are not known to exist—and there might even be evidence that they do not exist [Leroux and Schnoebelen, 2014].

Hardnesss proofs for coverability can rely instead on the implementation of *counter objects*. These should really be understood as an 'object-oriented' style of implementing bounded counters in the system at hand. Given a natural number $n \in \mathbb{N}$, a *counter object* allows to manipulate an internal, private representation of a counter in the range $[0, n-1]$ through an interface allowing to

init: construct an instance with initial value 0,
min: check whether the internal state represents 0,
max: check whether the internal state represents $n-1$,

incr: increment the represented counter by 1,
decr: decrement the represented counter by 1.

These operations deadlock if the check fails or the counter value would go outside the range $[0, n-1]$. When an implementation of counter objects for n is available, reducing from the halting problem for n-bounded multicounter Minsky machines is straightforward.

Given a counter object for n, one is sometimes able to implement a counter object for $f(n)$ for some specific function $f\colon \mathbb{N} \to \mathbb{N}$, using internally references to instances of the counter object for n. This mechanism allows therefore compositions of functions, or even iterations of functions.

This technique was introduced in by Lipton [1976] to prove the EXPSPACE-hardness of Coverability in *vector addition systems*, where one can start with a counter object for 2^{2^0} and compose it d times with the squaring function $f(n) \stackrel{\text{def}}{=} n^2$ to implement a counter object for 2^{2^d} for any d. The more recent uses of counter objects in several hardness proofs are due notably to Lazić [Lazić et al., 2016; Lazić and Totzke, 2017; J4; C5]. The complexity lower bounds in Table 5.2 on page 65 and in Table 6.1 on page 72 have all been obtained through applications of this idea.

The main difficulty, with both weak Hardy computations and counter objects, lies with the implementation in systems of restricted capabilities, which can quickly become very hard to manage [e.g. C16; C5]. The usage of proof assistants to ensure the correctness of the implementations would sometimes seem warranted.

4.3.2. Provability in Theories of Arithmetic. We have heavily relied in this chapter and the previous one on definitions and results—including the subrecursive functions like the Hardy functions $(H^\alpha)_\alpha$, the subrecursive hierarchies like the extended Grzegorczyk hierarchy $(\mathscr{F}_\alpha)_\alpha$, and more generally ordinal notation systems—that have been studied in mathematical logic and proof theory to analyse the strength of logical theories and find 'natural' mathematical statements independent from a theory [e.g. Fairtlough and Wainer, 1998; Ketonen and Solovay, 1981; Schwichtenberg and Wainer, 2012].

The 'proof-theoretic ordinal' of a logical theory is, very informally, the least ordinal whose notation is not provable in the theory [Rathjen, 2006]. For instance, Gentzen's proof of the consistency of Peano arithmetic PA entails that it has proof-theoretic ordinal ε_0; similarly, the fragments $\mathsf{I}\Sigma_i$ of PA, which are restricted to use the induction scheme on Σ^0_i formulæ, have proof-theoretic ordinal $\varepsilon_0(i)$, i.e. a tower of ω's of height $i+1$. It turns out that the proof-theoretic ordinal of a theory may also inform us on which qos can be proven to be wqo [Simpson, 1988], and which functions of the Hardy hierarchy $(H^\alpha)_\alpha$ can be proven total in the theory: for a well-studied instance, $\mathsf{I}\Sigma_i$ can prove the totality of H^α if and only if $\alpha < \varepsilon_0(i)$ [e.g. Schwichtenberg and Wainer, 2012, Theorem 4.3.4].

Thus $\mathbf{F}_\alpha$-complete problems like the coverability problems in Table 4.1 are a likely source of natural 'decidability statements' independent from various theories: I mean here the statement that there is (an encoding of) a Turing machine that decides the problem, i.e. terminates on all inputs with the correct acceptance or rejection verdict. I conjecture that the decidability of $\mathbf{F}_\omega$-complete coverability problems like LCM Coverability is independent from $\mathsf{I}\Sigma_1$, that the decidability

of $\mathbf{F}_{\omega^\omega}$-complete problems like Coverability in Lossy Channel Machines or Unordered Data Nets is independent from $\mathsf{I}\Sigma_2$, ..., and that the decidability of Coverability in Priority Channel Systems is independent from PA. The consequence would be that, while these problems are decidable, their decidability proofs require strong logical theories. The same conjecture can be made for fragments of second-order arithmetic like RCA_0 (with proof-theoretic ordinal ω^ω) and ACA_0 (with proof-theoretic ordinal ε_0).

CHAPTER 5

Reachability in Vector Addition Systems

Vector addition systems with states (VASS) are essentially multi-counter Minsky machines without zero tests. These systems are equivalent to Petri nets, and find a wide range of applications in the modelling of concurrent, chemical, biological, or business processes.

A Cornerstone Problem. The decidability of their *reachability problem* is considered as one of the greatest achievements of theoretical computer science. Its 1981 decidability proof for the reachability problem by Mayr is the culmination of more than a decade of research into the topic, and the algorithm has since been revisited and simplified by Kosaraju [1982] and Lambert [1992].

The centrality of the problem was recognised long before it was shown decidable; what is remarkable however is the regularity with which decision problems—in seemingly unrelated areas, in logic [Bojańczyk et al., 2011; Demri et al., 2016; Kanovich, 1995], automata [Crespi-Reghizzi and Mandrioli, 1977; Gischer, 1981; Render and Kambites, 2009], verification [Esparza et al., 2017; Ganty and Majumdar, 2012; German and Sistla, 1992], etc.—turn out to be interreducible with the reachability problem. In fact, given its importance in many fields, it would be no exaggeration to define a complexity class REACHABILITY for the class of problems inter-reducible with VASS reachability; a representative sample can be found in [I2, Section 5].

Complexity. In spite of its importance, fairly little is known about the computational complexity of the reachability problem. Regarding the general case, the inclusive surveys on the complexity of decision problems on VASS by Esparza and Nielsen [1994; 1998] could only point to the EXPSPACE lower bound of Lipton [1976] and to the fact that the running time of the known algorithms is not primitive recursive: *no* complexity upper bound was known, not even a coarse one, besides decidability—Bouziane claimed to have found an algorithm with primitive-recursive complexity in 1998, but it was shown incorrect by Jančar in 2008.

In [C8], I have shown with Leroux that reachability has a 'cubic Ackermann' upper bound, i.e. is in $\mathbf{F}_{\omega^3}$, by analysing the complexity of the classical algorithm developed and refined by Mayr, Kosaraju, and Lambert. Recall from Chapter 4 that $\mathbf{F}_{\omega^3}$ is a non primitive-recursive complexity class, but among the lower multiply-recursive ones; see Figure 5.1. The main ingredients for the analysis performed in [C8] are the fast-growing complexity bounds for termination proofs by well-quasi-orders and ordinal ranking functions from [C19; I3] and presented in Chapter 3.

In this chapter, I present a slightly improved version of the result from [C8], with a 'quadratic Ackermann' upper bound.

THEOREM 5.1 (Upper Bound Theorem). *VASS Reachability is in* $\mathbf{F}_{\omega^2}$*, and in* $\mathbf{F}_{\omega\cdot(d+1)}$ *if the dimension* d *is fixed.*

I give here a high-level presentation based chiefly on the invited column [I2], with pointers to [C8] for the detailed arguments. The algorithm used by Mayr, Kosaraju, and Lambert for the reachability problem is explained semi-formally on an example in Section 5.2, before we can work out how to apply the length function theorems from Chapter 3 and derive an $\mathbf{F}_{\omega^2}$ upper bound. Let us first recall the basic definitions in the upcoming section.

Contents

5.1. Basic Definitions

5.1.1. Vector Addition Systems with States. Formally, a VASS [Hopcroft and Pansiot, 1979] is a tuple $\mathcal{V} = \langle Q, d, T\rangle$ where Q is a finite set of 'control' states, d in $\mathbb{N}$ is a non-negative dimension, and $T \subseteq Q \times \mathbb{Z}^d \times Q$ is a finite set of transitions.

The operational semantics of such a system is captured by an infinite transition system $\mathcal{S}_\mathcal{V}$ over the set of *configurations* $Confs_\mathcal{V} \stackrel{\text{def}}{=} Q \times \mathbb{N}^d$, with a *step* $(q, \boldsymbol{u}) \xrightarrow{t}_\mathcal{V} (q', \boldsymbol{u}+\boldsymbol{a})$ defined whenever $t = (q, \boldsymbol{a}, q')$ belongs to T; note that $\boldsymbol{u}+\boldsymbol{a}$ must belong to $\mathbb{N}^d$ for such a step to be possible. A *run* from a configuration c_0 to

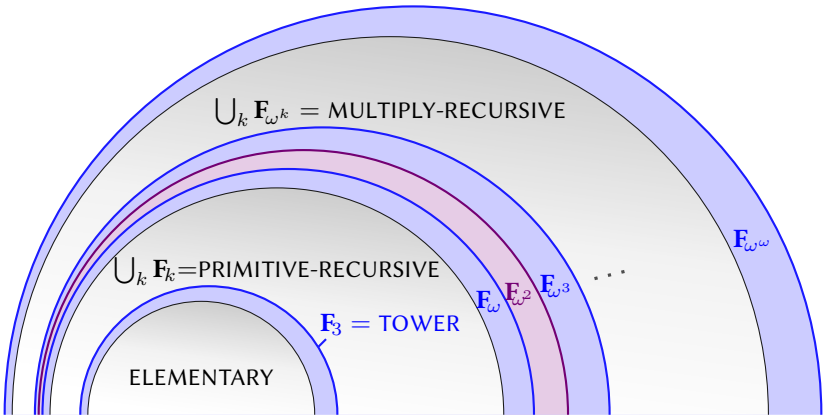


FIGURE 5.1. Pinpointing $\mathbf{F}_{\omega^2}$ among the complexity classes beyond ELEMENTARY.

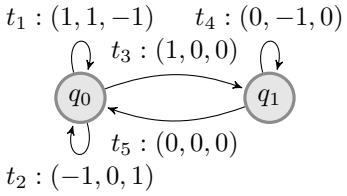


FIGURE 5.2. A 3-dimensional VASS.

a configuration c_ℓ is a finite sequence of steps $c_0 \xrightarrow{t_1}_{\mathcal{V}} c_1 \xrightarrow{t_2}_{\mathcal{V}} c_2 \cdots c_{\ell-1} \xrightarrow{t_\ell}_{\mathcal{V}} c_\ell$, which can also be written $c_0 \xrightarrow{t_1 \cdots t_\ell}_{\mathcal{V}} c_\ell$. Finally, let us write $c_0 \to^*_{\mathcal{V}} c_\ell$ if there exists a finite sequence of transitions $\sigma \in T^*$ such that $c_0 \xrightarrow{\sigma}_{\mathcal{V}} c_\ell$.

Note that $\mathcal{S}_{\mathcal{V}} = \langle \mathit{Confs}_{\mathcal{V}}, \to_{\mathcal{V}}, \leq \rangle$, where $(q, \boldsymbol{u}) \leq (q', \boldsymbol{u}')$ if and only if $q = q'$ and $\boldsymbol{u}(i) = \boldsymbol{u}'(i)$ for all $1 \leq i \leq d$, is a WSTS as defined in Section 2.3, and VASS Coverability is decidable. We focus here on a more difficult problem: exact reachability in the infinite system $\mathcal{S}_{\mathcal{V}}$.

PROBLEM (VASS Reachability).

instance: A VASS $\mathcal{V}$ and two configurations c and c' in $\mathit{Confs}_{\mathcal{V}}$,
question: Can c reach c', i.e. does $c \to^*_{\mathcal{V}} c'$?

EXAMPLE 5.2. Consider for instance the 3-dimensional VASS of Figure 5.2 with $Q \stackrel{\text{def}}{=} \{q_0, q_1\}$ and $T \stackrel{\text{def}}{=} \{t_1, t_2, t_3, t_4, t_5\}$. One can check that $(q_0, 1, 0, 1)$ reaches $(q_1, 2, 2, 1)$, for instance by the run

$$(q_0,1,0,1) \xrightarrow{t_1} (q_0,2,1,0) \xrightarrow{t_2} (q_0,1,1,1) \xrightarrow{t_1} (q_0,2,2,0) \xrightarrow{t_2} (q_0,1,2,1) \xrightarrow{t_3} (q_1,2,2,1)\,.$$

This is just one example of a run witnessing reachability; observe that any sequence of transitions in $\{t_1t_2, t_2t_1\}^{n+2}t_3t_4^n$ for $n \geq 0$ would similarly do.

REMARK 5.3 (Binary Encodings). Regarding complexity, one typically assumes a binary encoding of the integers of a VASS and of the source and target configurations. If we let $\|\boldsymbol{a}\| \stackrel{\text{def}}{=} \max_{1 \leq i \leq d} |\boldsymbol{a}(i)|$ denote the infinity norm of a vector $\boldsymbol{a}$ in $\mathbb{Z}^d$, then the maximal effect $\|T\| \stackrel{\text{def}}{=} \max_{(q,\boldsymbol{a},q') \in T} \|\boldsymbol{a}\|$ can be exponential in the

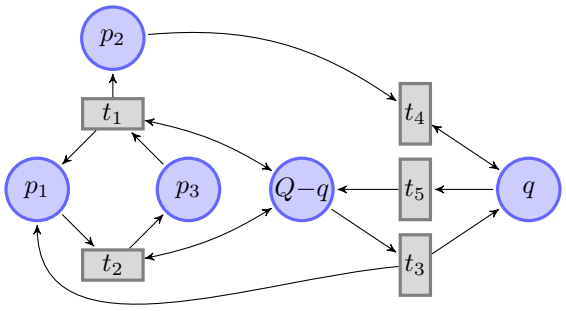


FIGURE 5.3. A Petri net equivalent to the VASS of Figure 5.2.

size of a VASS $\mathcal{V} = \langle Q, d, T\rangle$. The choice of a binary rather than a unary encoding has no impact in the general case—because there is a LOGSPACE reduction to the case where $T \subseteq Q \times \{-1, 0, 1\}^d \times Q$ (at the expense of increasing the dimension) and $c = (q, \mathbf{0})$ and $c' = (q', \mathbf{0})$ for some states q, q'—, but is important in fixed dimension, see §5.4.2.2.

5.1.2. Closely Related Models. Historically, VASS do not seem to have been studied before the works of Greibach [1978] and Hopcroft and Pansiot [1979]. Nevertheless, equivalent models had been investigated before, in particular *Petri nets* [Petri, 1962] and *vector addition systems* (VAS) [Karp and Miller, 1969]. The absence of explicit control states makes these two classes of models rather convenient for the modelling of concurrent or distributed systems.

5.1.2.1. *Petri Nets.* A Petri net is a tuple $\mathcal{N} = \langle P, T, W\rangle$ where P is a finite set of *places*, T is a finite set of *transitions*, and $W\colon (P \times T) \cup (T \times P) \to \mathbb{N}$ is a (weighted) *flow* function. It defines a transition system with configurations in $\mathbb{N}^P$—i.e. multisets of places, also called *markings*—and steps $m \xrightarrow{t} m'$ whenever $m(p) \geq W(p,t)$ and $m'(p) = m(p) - W(p,t) + W(t,p)$ for all p in P. A Petri net can be encoded as an equivalent $|P|$-dimensional VASS with $|T| + 1$ states, and conversely a d-dimensional VASS can be encoded as an equivalent Petri net with $d + 2$ places (see Figure 5.3 for the result of this construction on the VASS of Figure 5.2, where places are depicted as circles, transitions as rectangles, and flows as arrows)—'equivalence' here should be understood as far as the decision problems like reachability are concerned.

5.1.2.2. *Vector Addition Systems.* A VAS is a pair $\langle d, \boldsymbol{A}\rangle$ where $\boldsymbol{A}$ is a finite subset of *actions* in $\mathbb{Z}^d$ [Karp and Miller, 1969]. It defines a transition system with configurations $\boldsymbol{u}$ in $\mathbb{N}^d$ and steps $\boldsymbol{u} \to \boldsymbol{u} + \boldsymbol{a}$ for $\boldsymbol{a}$ in $\boldsymbol{A}$, again implicitly checking that $\boldsymbol{u} + \boldsymbol{a} \geq \mathbf{0}$. Put differently, a VAS can be seen as a VASS with a singleton state set. Conversely, the finite control of a d-dimensional VASS can be encoded in an equivalent VAS by increasing the system's dimension to $d+3$ [Hopcroft and Pansiot, 1979, Lemma 2.1].

5.2. The Decomposition Algorithm

The reachability problem was famously shown to be decidable by Mayr [1981], building notably on an incomplete proof by Sacerdote and Tenney [1977].

THEOREM 5.4 (Decidability Theorem; Mayr, 1981). *VASS Reachability is decidable.*

This proof has since been simplified twice: one year later by Kosaraju [1982], and another ten years later by Lambert [1992]. At the heart of these proofs lies a *decomposition technique*, which is called the 'Kosaraju-Lambert-Mayr-Sacerdote-Tenney' (KLMST) decomposition. In a nutshell, the KLMST decomposition defines both

- a structure (resp. *regular constraint graphs* for Mayr, *generalised VASS* for Kosaraju, and *marked graph-transition sequences* for Lambert) and
- a condition for this structure to represent in some way the set of all runs witnessing reachability (resp. *consistent marking*, the *θ condition*, and the *perfect condition*).

The algorithms proposed by Mayr, Kosaraju, and Lambert compute this decomposition by successive refinements of the structure until the condition is fulfilled, by which time the existence of a run becomes trivial.

The reader is referred to the original articles of Kosaraju [1982] and Lambert [1992], and to the excellent accounts by Müller [1985], Reutenauer [1990], and Leroux [2010] for examples and details on the KLMST decomposition. Here we shall keep the description at an informal level, and see how the decomposition algorithm works in the case of Example 5.2 without entering its details.

5.2.1. Marked Witness Graph Sequences. We fix the VASS $\mathcal{V} = \langle Q, d, T\rangle$ under consideration. Let us first complete $\mathbb{N}$ with a top element ω and write $\mathbb{N}_\omega \stackrel{\text{def}}{=} \mathbb{N} \uplus \{\omega\}$ for the result; also let $\omega + z = z + \omega = \omega$ for all z in $\mathbb{Z}$.

A *witness graph* is a finite strongly connected directed graph $G = (S, E)$ with vertices $S \subseteq Q \times \mathbb{N}_\omega^d$, and labelled edges $E \subseteq S \times T \times S$, such that the edge labels from T are consistent with the vertices from S. This means that, if (s, t, s') is an edge in E with transition $t = (q, \boldsymbol{a}, q')$ from T as label, then $s = (q, \boldsymbol{u})$ for some $\boldsymbol{u}$ in $\mathbb{N}_\omega^d$ and $s' = (q', \boldsymbol{u} + \boldsymbol{a})$. Note that these conditions together imply that all the vertices in the graph share the same set $I \subseteq \{1, \ldots, d\}$ of ω-components.

A *marked witness graph* $M = (G, c^{\text{in}}, s^{\text{in}}, c^{\text{out}}, s^{\text{out}})$ is further endowed with distinguished input and output vertices s^{in} and s^{out} from S, along with input and output constraints c^{in} and c^{out} taken from $Q \times \mathbb{N}_\omega^d$, such that for all $1 \leq i \leq d$, $s^{\text{in}}(i) \neq \omega$ implies $c^{\text{in}}(i) = s^{\text{in}}(i)$, and similarly for the output vertex and constraint. In other words, s^{in} and c^{in} agree on their finite components. This entails that I^{in} the set of ω-components of c^{in} is a subset of I the set of ω-components of s^{in}, and similarly $I^{\text{out}} \subseteq I$.

Finally, a *marked witness graph sequence* ξ is a sequence

$$\xi = M_0, t_1, M_1, \ldots, t_k, M_k \tag{5.1}$$

alternating marked witness graphs $M_0, \ldots, M_k$ and transitions $t_1, \ldots, t_k$ taken from T. Let us write $M_j = (G_j, c_j^{\text{in}}, s_j^{\text{in}}, c_j^{\text{out}}, s_j^{\text{out}})$ and $t_j = (q_j, \boldsymbol{a}_j, q'_j)$ for all j. It is also required that, for all $1 \leq j \leq k$, $c_{j-1}^{\text{out}} = (q_j, \boldsymbol{u}_j)$ for some $\boldsymbol{u}_j$ and $c_j^{\text{in}} = (q'_j, \boldsymbol{u}'_j)$ for some $\boldsymbol{u}'_j$. In such a sequence, c_0^{in} is the *source* and c_k^{out} is the *target*.

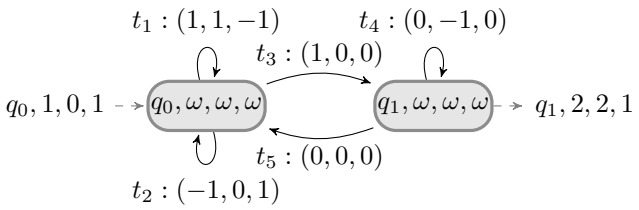


FIGURE 5.4. The initial marked witness graph sequence $\xi_0 = M_0$.

Figure 5.4 displays a marked witness graph for the VASS of Example 5.2, with input constraint $(q_0, 1, 0, 1)$ on the input vertex $(q_0, \omega, \omega, \omega)$ and output constraint $(q_1, 2, 2, 1)$ on the output vertex $(q_1, \omega, \omega, \omega)$; this is the initial marked witness graph for the decomposition algorithm when attempting to decide whether $(q_1, 2, 2, 1)$ is reachable from $(q_0, 1, 0, 1)$.

5.2.2. An Example of a KLMST Decomposition. The KLMST decomposition algorithm builds a sequence $\Xi_0, \Xi_1, \Xi_2, \ldots$ of finite sets of marked witness graph sequences. At step n, it checks whether all the sequences ξ in Ξ_n are *perfect* (in the sense of Lambert [1992], or equivalently fulfil the θ-condition of Kosaraju [1982]) and stops if this is the case; then either Ξ_n is empty and the algorithm answers 'not reachable', or Ξ_n is not empty and the algorithm answers 'reachable'.

If however some sequence ξ from Ξ_n is not perfect, then it is *decomposed* into a finite set of marked witness graph sequences $\mathrm{dec}(\xi)$—which is possibly empty. Then we let

$$\Xi_{n+1} \stackrel{\text{def}}{=} (\Xi \setminus \{\xi\}) \cup \mathrm{dec}(\xi) \tag{5.2}$$

and the algorithm proceeds to the next step.

The perfectness condition comprises two sub-conditions, along with the corresponding ways of decomposing marked witness graph sequences when the sub-conditions are violated. We are going to illustrate these two sub-conditions in the upcoming §5.2.2.1 and §5.2.2.2, in the case of Example 5.2, and starting from $\Xi_0 = \{\xi_0\}$.

5.2.2.1. *Flow Constraints.* Consider a path from the source to the target in the graph of Figure 5.4: denoting by z_j the number of times transition t_j is used along this path for $j \in \{1, \ldots, 5\}$, we can see that

$$z_3 = z_5 + 1 \ . \tag{5.3}$$

Consider now a run in the VASS of Figure 5.2, which follows a path in the marked witness graph of Figure 5.4 from $(q_0, 1, 0, 1)$ to $(q_1, 2, 2, 1)$, i.e. with overall effect $(1, 2, 0)$. Then, considering the effect of these transitions for each coordinate i in $\{1, 2, 3\}$,

$$\begin{aligned} z_1 + z_3 &= z_2 + 1 \ , \\ z_1 &= z_4 + 2 \ , \\ z_1 &= z_2 \ . \end{aligned} \tag{5.4}$$

The system of equations (5.3–5.4) requires $z_3 = 1$ and $z_5 = 0$; z_1, z_2 and z_4 are on the other hand unbounded.

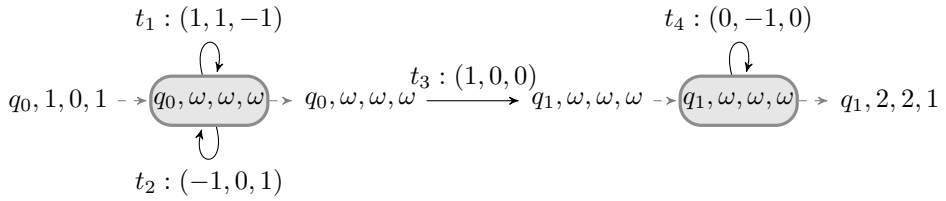


FIGURE 5.5. The next marked witness graph sequence $\xi_1 = M'_0, t_3, M'_1$.

This shows that the marked witness graph sequence ξ_0 of Figure 5.4 is too permissive, allowing to follow paths that do not bring the source of the sequence to its target. We therefore decompose it, using the fact that t_3 must be employed exactly once and that t_5 is never employed: $\mathrm{dec}(\xi_0) = \{\xi_1\}$ where the new sequence ξ_1 is depicted in Figure 5.5; it contains two marked witness graphs M'_0 and M'_1 connected by a single occurrence of transition t_3.

In general, flow constraints are expressed using two systems of equations built from the sequence ξ. One system ensures that there exists a solution to Kirchhoff's Laws when considering the finite values in c_j^{in} and c_j^{out}; if not, then $\mathrm{dec}(\xi) = \emptyset$ because no run enforcing these constraints can exist. The second system ignores these finite values and allows to check that there exist solutions with

- unbounded number of occurrences of each transition in $\bigcup_j E_j$—if not the offending marked graph is unfolded finitely many times up to the bound, as in the example—and
- unbounded values can instantiate the ω's in the input and output constraints c_j^{in} and c_j^{out}—if not we replace these ω's by finite values below the bound.

The checks can be carried in polynomial time in the size of ξ, while the extracted bounds in case of a decomposition are at most exponential in this size. See for instance [Leroux, 2010, Section 3] for a detailed exposition.

5.2.2.2. *Pumpability.* A marked witness graph M is *forward pumpable* if there exist runs in the VASS following paths of M and starting from the input vertex which, when applied to the input constraint, allow to 'pump' arbitrarily high values in the (necessarily common) components labelled ω in the vertices of the graph.

On the one hand, M'_0 in Figure 5.5 is not forward pumpable: any run of the VASS of Figure 5.2 starting from $(q_0, 1, 0, 1)$ and using only t_1 and t_2 can indeed reach arbitrarily high values on the second component, but the first and third components are bounded.

On the other hand, M'_1 is forward pumpable, but not *backward pumpable*: starting from $(q_1, 2, 2, 1)$ and applying t_4 in reverse allows to reach arbitrarily high values on the second component, but the first and third components are again bounded.

Again, the decomposition algorithm will observe that the current marked witness graph sequence over-approximates the possible behaviours of the VASS, and refine M'_0 and M'_1 using their bounded components. Propagating the flow constraints, we obtain the final marked witness sequence depicted in Figure 5.6. This

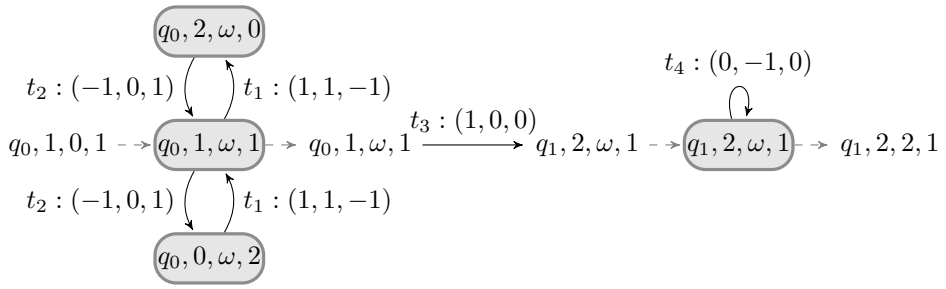


FIGURE 5.6. The final marked witness graph sequence $\xi_2 = M_0'', t_3, M_1''$.

sequence is perfect, and captures in some sense[1] all the runs from $(q_0, 1, 0, 1)$ to $(q_1, 2, 2, 1)$ in the VASS of Example 5.2.

In general, forward and backward pumpability reduce to instances of the *place boundedness* problem on the marked witness graph under consideration, which can be solved in EXPSPACE [e.g. C17]. If pumpability fails, the bounded values can be computed in practice using the coverability tree construction of Karp and Miller [1969], with Ackermannian upper bounds [C19, Section VII-C]. See again [Leroux, 2010, Section 3] for a detailed exposition.

5.2.3. Termination. The termination of the KLMST decomposition algorithm relies on a *ranking function* r mapping marked witness graph sequences to elements of a well-order, and ensuring $r(\xi) > r(\xi')$ whenever ξ' belongs to $\mathrm{dec}(\xi)$ [Kosaraju, 1982]. More precisely, the ranking function r associates to ξ a multiset of triples of natural numbers, one triple for each marked witness graph in the sequence. These triples consist of

(1) $|I|$, the number of ω-components of the marked witness graph,
(2) $|E|$, the number of transitions of the marked witness graph, and
(3) $|I^{\mathrm{in}}| + |I^{\mathrm{out}}|$, the number of ω-components in the input and output constraints.

This results for the sequences ξ_0, ξ_1, and ξ_2 of our example in the multisets

$$\begin{aligned} r(\xi_0) &= \{(3, 5, 0)\} \,, \\ r(\xi_1) &= \{(3, 2, 3), (3, 1, 3)\} \,, \\ r(\xi_2) &= \{(1, 4, 1), (1, 1, 1)\} \,. \end{aligned} \tag{5.5}$$

Let us consider the lexicographic ordering over $\mathbb{N}^3$; finite multisets of triples in $\mathbb{N}^3$ are then well-ordered using the ordering of Dershowitz and Manna [1979].

We can see the KLMST algorithm as building in general a forest of marked witness graph sequences, with the elements of Ξ_0 as its finitely many roots, and where each imperfect marked witness graph sequence ξ is the parent of the sequences in $\mathrm{dec}(\xi)$. The ranking function r then shows that the trees in this forest are of finite height; since $\mathrm{dec}(\xi)$ is finite for all ξ, they are also of finite branching degree, hence the trees are finite by Kőnig's Lemma and the algorithm terminates.

[1] It represents exactly the *downward closure* of the set of runs from $(q_0, 1, 0, 1)$ to $(q_1, 2, 2, 1)$; see the Decomposition Theorem of [C8] and Section 6.4.2, which might also help the reader build an intuition about marked witness graph sequences and the KLMST decomposition algorithm.

5.3. Complexity Upper Bound

Hopefully, the reader has now some intuition about the KLMST decomposition algorithm. The key point for complexity considerations is the termination argument by a ranking function explained in Section 5.2.3: we know that any sequence $\xi_0, \xi_1, \xi_2, \dots$ of marked witness graph sequences with $\xi_{n+1} \in \mathrm{dec}(\xi_n)$ is finite since

$$r(\xi_0) > r(\xi_1) > r(\xi_2) > \cdots \tag{5.6}$$

is a decreasing sequence in the well-order of multisets of triples of naturals. In order to bound the complexity of the KLMST decomposition algorithm, we are going to bound the length L of such sequences. We shall relate this length with the order type of the ranking function.

5.3.1. An Ordinal Ranking Function. The order type of multisets of triples of natural numbers is ω^{ω^3} [Dershowitz and Manna, 1979], and we can equivalently see the ranking function r of Section 5.2.3 as ranging over ordinals below ω^{ω^3}: the ranks in (5.5) then become

$$\begin{aligned} r(\xi_0) &= \omega^{\omega^2\cdot 3+\omega\cdot 5}, \\ r(\xi_1) &= \omega^{\omega^2\cdot 3+\omega\cdot 2+3} + \omega^{\omega^2\cdot 3+\omega+3}, \\ r(\xi_2) &= \omega^{\omega^2+\omega\cdot 4+1} + \omega^{\omega^2+\omega+1}. \end{aligned} \tag{5.7}$$

This is where we can refine the results from [C8]. In the ranking function defined in Section 5.2.3, the first component of the triples is bounded by d, and the third by $2d$. Thus the order type of the ranking function is actually $\omega^{\omega\cdot(d+1)}$. To see this, define the rank of a single marked witness graph M with edge set E, ω-components I, and input and output ω-components I^{in} and I^{out} by

$$r(M) \stackrel{\text{def}}{=} \omega \cdot |I| + (2d+1) \cdot |E| + |I^{\mathrm{in}}| + |I^{\mathrm{out}}| \tag{5.8}$$

and the rank of a marked witness graph sequence by

$$r(M_0, t_1, M_1, \dots, t_k, M_k) \stackrel{\text{def}}{=} \bigoplus_{0\le j\le k} \omega^{r(M_j)} \tag{5.9}$$

where '$\oplus$' denotes the natural sum over ordinals (see Appendix A.1.1.1). This results in the ordinal ranks

$$\begin{aligned} r(\xi_0) &= \omega^{\omega\cdot 3+7\cdot 5} = \omega^{\omega\cdot 3+35}, \\ r(\xi_1) &= \omega^{\omega\cdot 3+7\cdot 2+3} + \omega^{\omega\cdot 3+7+3} = \omega^{\omega\cdot 3+17} + \omega^{\omega\cdot 3+10}, \\ r(\xi_2) &= \omega^{\omega+12} + \omega^{\omega+8} \end{aligned} \tag{5.10}$$

for the multisets of (5.5) instead of the ordinal ranks of (5.7).

5.3.2. Applying the Length Function Theorems. If we provide an initial norm $n \ge d+1$ and a control function g for sequences like (5.5), we will obtain by Theorem 3.5 on page 25 an

$$L \stackrel{\text{def}}{=} g_{\omega^{\omega\cdot(d+1)}}(n) \tag{5.11}$$

bound on their lengths. Regarding the initial norm, the maximum of the sizes of the initial sequences in Ξ_0 will do, and is bounded by the size of the reachability instance, which is certainly at least $d+1$. For the control function, let us first

note that $Nr(\xi) \leq \|\xi\|$ for any reasonable definition of the size $\|\xi\|$ of a marked witness graph sequence ξ. Thus we will bound the size of the marked witness graph sequences ξ' in $\mathrm{dec}(\xi)$ compared to that of ξ. This is done formally in [C8, Section IX-C], where a control function

$$g(x) \stackrel{\text{def}}{=} H^{\omega^{d+1}}(e(x)) \tag{5.12}$$

for an elementary function e is shown. Let us examine how this control function is obtained.

5.3.2.1. *Controlling Decompositions for Flow Constraints.* In the case of the flow conditions of §5.2.2.1, $\|\xi'\| \leq 2^{p(\|\xi\|)}$ for some polynomial p, as it is constructed based on the solution to a polynomial-sized system of equations derived from ξ.

5.3.2.2. *Controlling Decompositions for Pumpability.* In the case of the pumping conditions of §5.2.2.2, the blow-up can be bounded using Theorem 3.15 from page 30. Indeed, the size of ξ' can be bounded by $\|\xi\| \cdot B^d$ where B bounds the maximal finite values in a coverability tree construction à la Karp and Miller [1969] applied to the marked witness graph $M = ((S, E), c^{\text{in}}, s^{\text{in}}, c^{\text{out}}, s^{\text{out}})$ under consideration.

As shown in [C19, Section VII-C], these values can be analysed by considering the length of bad sequences over the nwqo $(S \times \mathbb{N}^d) \cdot d$ with maximal order type $\omega^d \cdot |S|d$. The sequences are controlled with initial norm $c \stackrel{\text{def}}{=} \|c^{\text{in}}\|$ or $c \stackrel{\text{def}}{=} \|c^{\text{out}}\|$ depending on whether we are checking forward or backward pumpability, and by the function $f(x) \stackrel{\text{def}}{=} x + \|T\|$ representing how fast the values can grow in the VASS. Choosing $h(x) \stackrel{\text{def}}{=} x + \|T\|d = H^{\|T\|d}(x)$ fits the conditions of Theorem 3.15, and Equation (3.13) then provides a bound

$$B \leq h^{\omega^d \cdot |S|d}(cd) = (H^{\|T\|d})^{\omega^d \cdot |S|d}(cd) \leq H^{\omega^d \cdot \|T\||S|d^2}(cd) \tag{5.13}$$

by Equation (3.12). Letting $p(\|\xi\|) \stackrel{\text{def}}{=} \max(\|T\||S|d^2, cd)$,

$$B \leq H^{\omega^{d+1}}(p(\|\xi\|)) \ . \tag{5.14}$$

By Lemma 4.4 and assuming $d \geq 2$, $\|\xi'\| \leq \|\xi\| \cdot B^d \leq H^{\omega^{d+1}}(e(\|\xi\|))$ for some elementary function e as advertised in (5.12).

5.3.2.3. *Fast-Growing Complexity Bounds.* The bound in (5.11) also provides a bound $g^L(n)$, i.e. of L iterations of the function g, on the size of the marked witness graph sequences constructed by the KLMST decomposition algorithm. Thus we have a bound on the space complexity of a non-deterministic version of the algorithm, which explores the decomposition tree until it finds a perfect marked witness graph. Note that we neglect here the various algorithmic overheads of testing for perfectness—this is in EXPSPACE—and of actually building the decompositions, because Lemma 4.4 allows us to hide all these in the elementary function e of (5.12).

By Equation (3.13) and because $n \geq d + 1$, this yields a bound of

$$g^L(n) = g^{\omega^{\omega \cdot (d+1)}}(n) \leq g^{\omega^{\omega^2}}(n) \tag{5.15}$$

using the Hardy hierarchy on the space complexity of this algorithm. By Theorem 4.6 on page 42, and because g in (5.12) is in $\mathscr{F}_{d+1}$, or $\mathscr{F}_\omega$ when d is part of the input, we obtain an $\mathbf{F}_{\omega \cdot (d+1)}$ upper bound when d is fixed, and a 'quadratic Ackermann' $\mathbf{F}_{\omega^2}$ upper bound in general, as announced in Theorem 5.1.

Remark 5.5. Recall from Chapter 4 that the following complexity classes are different: 'quadratic Ackermann' complexity $\mathbf{F}_{\omega^2}$, which as we just saw contains VASS reachability, 'double Ackermann' complexity $\mathbf{F}_{\omega\cdot 2}$, which is for instance the complexity of coverability in ν-Petri nets [C5], 'Ackermann plus one' complexity $\mathbf{F}_{\omega+1}$, and 'composed Ackermann' complexity 2-$\mathbf{F}_{\omega}$, which corresponds to problems solvable in time bounded by the Ackermann function composed with itself once (see §4.1.2.2). The underlying functions of the Hardy hierarchy that define these complexity classes are respectively

- $H^{\omega^{\omega^2}}$, which eventually dominates $H^{\omega^{\omega\cdot k}}$ for every k,
- $H^{\omega^{\omega\cdot 2}}$, which eventually dominates $H^{\omega^{\omega+k}}$ for every k,
- $H^{\omega^{\omega+1}}$, which eventually dominates $H^{\omega^{\omega}\cdot k}$ for every k, and
- $H^{\omega^{\omega}\cdot 2} = H^{\omega^{\omega}} \circ H^{\omega^{\omega}}$, which is the composition of the Ackermann function with itself.

There is a considerable complexity jump between each one of these classes, and they should not be confused with each other.

5.4. Related Work & Perspectives

There is a considerable literature surrounding the computational complexity of the VASS Reachability Problem and of several related problems, drawing a complex landscape of results. I give here a somewhat partial account of the current literature, summarised in Table 5.1. The main question, about how tight the $\mathbf{F}_{\omega^2}$ upper bound of Theorem 5.1 might be is discussed in Section 5.4.1. Given the lack of a definitive answer on the complexity of VASS Reachability, it is also natural to consider simpler problems, which I discuss in Section 5.4.2. The few known results and many open problems related to Reachability in VASS extensions are then quickly mentioned in Section 5.4.3; some of those questions will be further discussed in the next chapter.

5.4.1. Tightness. Facing a galactic upper bound like $\mathbf{F}_{\omega^2}$, it is natural to ask how tight it is. It seems very likely that it is not tight; to the best of our knowledge, the problem might very well be EXPSPACE-complete: the best complexity lower bound is indeed the following 1976 result of Lipton already mentioned in Section 4.3.1 (see also the presentation given by Esparza [1998]).

Theorem 5.6 (Lower Bound Theorem; Lipton, 1976). *VASS Reachability is EXPSPACE-hard.*

This leaves a gigantic gap between EXPSPACE and $\mathbf{F}_{\omega^2}$. Nevertheless, the upper bound is obtained with a specific algorithm, the KLMST decomposition algorithm, which—due to its reliance on coverability trees—is known to require at least an Ackermannian time in the worst case [Müller, 1985], i.e. there is an $\mathbf{F}_{\omega}$ lower bound for that particular algorithm. Hence, besides the main open question about the exact complexity of the reachability problem, there is a possibly easier open question about the complexity of the KLMST decomposition algorithm, with a smaller complexity gap between $\mathbf{F}_{\omega}$ and $\mathbf{F}_{\omega^2}$.

5.4.1.1. *Language Inclusion Problems.* In fact, the KLMST decomposition algorithm solves much more than just reachability, and we can justify the $\mathbf{F}_{\omega}$ lower

bound. Indeed, the final computed decomposition Ξ_n contains a lot of information about the 'set of executions' from c to c',[2] allowing in particular to reason about the set of sequences $\sigma \in T^*$ such that $c \xrightarrow{\sigma}_{\mathcal{V}} c'$.

An appropriate context here is the one of *labelled* VASS, i.e. when the VASS $\mathcal{V} = \langle Q, d, T, \ell, \Sigma \rangle$ is further equipped with a labelling function $\ell \colon T \to \Sigma^*$ into some finite alphabet Σ—which is lifted to a homomorphism from T^* to Σ^*—, and one can define its *reachability language*

$$L(\mathcal{V}, c, c') \stackrel{\text{def}}{=} \{\ell(\sigma) \in \Sigma^* \mid c \xrightarrow{\sigma}_{\mathcal{V}} c'\} \ . \tag{5.16}$$

The *language emptiness problem* then asks, given $\mathcal{V}$ and two configurations c, c', if $L(\mathcal{V}, c, c') = \emptyset$; this is the complement of the reachability problem and it is decidable. By contrast, the *language inclusion problem* asks, given two VASS $\mathcal{V}_1, \mathcal{V}_2$ and four configurations c_1, c_1', c_2, c_2', whether $L(\mathcal{V}_1, c_1, c_1') \subseteq L(\mathcal{V}_2, c_2, c_2')$. Like most behavioural quasi-orders on VASS, it is an undecidable problem [Jančar, 2001], and this was one of the first VASS problems to be shown undecidable [Hack, 1975].

Habermehl, Meyer, and Wimmel [2010] have shown that, once the KLMST decomposition algorithm has terminated, one can compute in polynomial time from the resulting decomposition Ξ_n a regular expression E_{Ξ_n} denoting the *downward-closure* of $L(\mathcal{V}, c, c')$ with respect to subword embedding $\leq_*$ (c.f. Section 2.1.2):

$$L(E_{\Xi_n}) = {\downarrow}L(\mathcal{V}, c, c') \ . \tag{5.17}$$

Thus the following variant of the language inclusion problem is decidable:

Problem (Downward VASS Language Inclusion).

instance: Two VASS $\mathcal{V}_1, \mathcal{V}_2$ and four configurations $c_1, c_1' \in \mathit{Confs}_{\mathcal{V}_1}$ and $c_2, c_2' \in \mathit{Confs}_{\mathcal{V}_2}$.

question: Is ${\downarrow}L(\mathcal{V}_1, c_1, c_1') \subseteq {\downarrow}L(\mathcal{V}_2, c_2, c_2')$?

While decidable, this problem was proven ACKERMANN-hard by Zetzsche [2016]; the conclusion is that any algorithm for VASS Reachability that computes a similar decomposition will require at least $\mathbf{F}_\omega$ time. One should therefore look for new algorithms; note that reachability only needs to find *some* execution witnessing reachability, while downward language inclusion needs to account for *all* the possible such executions.

5.4.1.2. *Inductive Presburger Invariants.* Leroux has invented in 2010 a new, very simple algorithm for VASS Reachability based on inductive Presburger invariants. The initial proof relied on the KLMST decomposition, but Leroux [2011] then re-proved the correctness of this new algorithm *without* referring to it, yielding an compact, and self-contained decidability proof for VASS Reachability.

While the new algorithm is considerably simpler than the KLMST decomposition algorithm, it is also more difficult to analyse, as it consists of two semi-decision procedures and its 2011 proof relies on non-constructive arguments. It should however be possible to extract complexity upper bounds from the 2010 proof based on the KLMST decomposition, but as mentioned above this decomposition has at least Ackermannian worst-case size.

5.4.2. Restrictions. Due to the considerable difficulty of both the decidability proofs themselves and of proving any upper bounds, it is natural to consider 'simpler' variants of the VASS Reachability Problem.

[2] See the Decomposition Theorem in [C8] and Section 6.4.2 for a formal viewpoint.

TABLE 5.1. The complexity of a few VASS problems.

Problem	Lower bound	Upper bound
Reachability	EXPSPACE [Lipton, 1976]	$\mathbf{F}_{\omega^2}$ [Th. 5.1]
Reversible Reachability	EXPSPACE [Lipton, 1976]	EXPSPACE [Leroux, 2013]
Coverability	EXPSPACE [Lipton, 1976]	EXPSPACE [Rackoff, 1978]
Dim-2 Reachability, binary encoding	PSPACE [Fearnley and Jurdziński, 2015]	PSPACE [Blondin et al., 2015]
Dim-1 Reachability, binary encoding	NP [Subset Sum]	NP [Haase et al., 2009]
Dim-2 Reachability, unary encoding	NL [STCON]	NL [Englert et al., 2016]
Language Inclusion	undecidable [Hack, 1975; Jančar, 2001]	
Downward Language Inclusion	ACKERMANN [Zetzsche, 2016]	$\mathbf{F}_{\omega^2}$ [Th. 5.1 + Habermehl et al., 2010]

TABLE 5.2. Reachability in VASS extensions; all the lower bounds already hold for the (top-down) coverability problem.

Problem	Lower bound	Upper bound
VASS with one zero test	EXPSPACE [Lipton, 1976]	decidable [Bonnet, 2013; Reinhardt, 2008]
Branching VASS	TOWER [J4]	open
Pushdown VASS	TOWER [Lazić and Totzke, 2017]	open
Unordered Data Petri Nets	ACKERMANN [Lazić and Totzke, 2017]	open

5.4.2.1. *Related Problems.* The best known variant is arguably the Coverability Problem (c.f. Section 2.3.3), where one asks instead for the existence of a configuration c'' such that $c \to_{\mathcal{V}} c''$ and $c'' \geq c'$ for the product ordering over configurations. The proof of Lipton [1976] for the EXPSPACE-hardness of reachability in Theorem 5.6 already applies to coverability, and the problem was shown to be in EXPSPACE by Rackoff [1978]. Coverability suffices for many algorithmic applications, and unlike the reachability, there are several implementations, with increasing success at solving it on large practical instances [e.g. Blondin et al., 2017b; Esparza et al., 2014; Geffroy et al., 2016; Kaiser et al., 2014].

Many more decision problems on VASS have been shown EXPSPACE-complete based on the techniques of Rackoff [e.g. Atig and Habermehl, 2011; Demri, 2013; Leroux et al., 2013; Rosier and Yen, 1986; C17]. Notably, the following *reversible reachability problem* is also EXPSPACE-complete [Leroux, 2013].

PROBLEM (Reversible VASS Reachability).

instance: A VASS $\mathcal{V}$ and two configurations c and c' in $\mathit{Confs}_{\mathcal{V}}$,
question: Can both $c \to^*_{\mathcal{V}} c'$ and $c' \to^*_{\mathcal{V}} c$?

The proof for the EXPSPACE upper bound by Leroux [2013] combines insights from both Rackoff [1978] and the KLMST decomposition algorithm. Adapting these ideas to prove tighter upper bounds for the KLMST decomposition algorithm seems promising, but has not been successful so far.

5.4.2.2. *Fixed Dimension.* A natural way of restricting the reachability problem is to fix the dimension. The $\mathbf{F}_{\omega\cdot(d+1)}$ upper bound of Theorem 5.1 in dimension d is rather unsatisfactory, as it assumes $d \geq 2$, and is hopelessly high: for instance, in dimension one with a binary encoding of weights, VASS Reachability is NP-complete [Haase et al., 2009]—this is a subcase of reachability in one-counter automata.

In the 2-dimensional case, Hopcroft and Pansiot [1979] were the first to show decidability by proving that the reachability set from a given initial configuration c was effectively *semilinear*, i.e. that one could compute for each state q a representation of the set of vectors $\boldsymbol{u}$ such that $c \to^*_{\mathcal{V}} (q, \boldsymbol{u})$ as a finite union of *linear* sets

$$L(\boldsymbol{b}, \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}) \stackrel{\text{def}}{=} \{\boldsymbol{b} + \lambda_1\boldsymbol{p}_1 + \cdots + \lambda_n\boldsymbol{p}_n \mid \lambda_1, \ldots, \lambda_n \in \mathbb{N}\} \tag{5.18}$$

defined by a *base* $\boldsymbol{b}$ in $\mathbb{Z}^d$ and a finite set of periods $\boldsymbol{p}_j$ in $\mathbb{Z}^d$—equivalently, these sets are definable in Presburger arithmetic $\mathrm{FO}(\mathbb{Z}, +, \leq)$. Howell, Rosier, Huynh, and Yen [1986] then showed that this construction was in 2-NEXP and improved it to obtain a 2-EXP algorithm. This left a gap with the NP-hardness proven by Rosier and Yen [1986] the same year. The lower bound was improved to PSPACE-hardness by Fearnley and Jurdziński [2015], and finally Blondin, Finkel, Göller, Haase, and McKenzie [2015] closed the complexity gap with a PSPACE upper bound.

These bounds hold with a binary encoding of the weights of the input VASS. As mentioned in Remark 5.3, the choice of a binary or unary encoding has an influence in fixed dimension, and Englert, Lazić, and Totzke [2016] further refined the analysis of Blondin et al. [2015] to show that reachability is NL-complete in dimension 2 when assuming a unary encoding.

The upper bounds in dimension 2 in [Blondin et al., 2015; Englert et al., 2016] are technical feats relying on fine geometric analyses of the effect of *semilinear*

path schemes [Leroux and Sutre, 2004] witnessing the semilinearity of reachability, i.e. regular expressions of the form

$$v_0 w_1^* v_1 \cdots w_k^* v_k \tag{5.19}$$

for some finite sequences $v_0, w_1, v_1, \ldots, w_k, v_k$ of transitions in T. However, Hopcroft and Pansiot [1979, Lemma 2.8] showed that there exists a 3-dimensional VASS with a non-semilinear reachability set, so this approach does not readily generalise to higher dimensions.

5.4.3. Extensions. The decidability of the reachability problem for VASS is an intricate result, and undecidability is never very far. This section is an opportunity to see how far the Decidability Theorem can be pushed, but also to advertise for a few open problems.

5.4.3.1. *Zero Tests.* A *zero test* is a special type of transition $t = (q, \mathrm{zero}_i, q')$ where i ranges over $\{1, \ldots, d\}$, allowing a step $(q, \boldsymbol{u}) \xrightarrow{t} (q', \boldsymbol{u})$ if $\boldsymbol{u}(i) = 0$. Allowing unrestricted zero tests yields a Minsky machine, with an undecidable reachability problem already in dimension two—even coverability is undecidable on such systems.

However, reachability in VASS extended with the ability to test a *single* component for zero—for instance always the first component—remains decidable. It still remains decidable if several components can be tested with a *hierarchical* policy: component i_1 can be freely tested for zero, but i_2 can only be tested for zero in configurations where the i_1th component is zero, and i_3 only in configurations where both the i_1th and i_2th components are zero, etc. [Bonnet, 2013; Reinhardt, 2008]. The complexity of reachability in these models is widely open; the best known lower bound is still Lipton's EXPSPACE-hardness, and no upper bound is known.

5.4.3.2. *Recursion and Nesting.* Motivated by the need to model distributed systems with some recursive behaviour, there is a variety of VASS extensions that include recursion in some manner, and differing on the (sometimes subtle) way in which they allow interactions between recursion and the integer components. For instance, *nested counter systems* [Decker et al., 2014; Decker and Thoma, 2016; Lomazova and Schnoebelen, 2000] act on finite multisets of finite multisets of ... of finite multisets of states as configurations, allowing to model hierarchical computations, but have an undecidable reachability problem. On the other hand, the *process rewrite systems* of Mayr [2000] perform prefix rewrites on terms of a process algebra, and generalise in a sense both VASS and pushdown systems, but still enjoy a decidable reachability problem—with unknown complexity.

Pushdown VASS. One natural way to extend VASS to handle recursion is to add new push and pop operations acting on a pushdown stack with a finite stack alphabet. Note that this generalises VASS with a single zero test, since the particular component tested for zero could be implemented as a stack with a distinguished bottom-of-stack symbol. The decidability of reachability is currently open, but here at least we have better lower bounds: Lazić and Totzke [2017] showed indeed the problem to be TOWER-hard.

Alternating Branching VASS. A different way of adding a pushdown stack to a VASS is to let it store vectors from $\mathbb{N}^d$ on its stack. The system can then add a vector $\boldsymbol{a}$ from $\mathbb{Z}^d$ to the vector $\boldsymbol{u}$ currently on top of the stack. The key question

is which semantics to employ when popping $\boldsymbol{u}$ and pushing multiple vectors, say $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, to the top of the stack. For instance, if we allow to duplicate the vector $\boldsymbol{u}$ so that $\boldsymbol{u} = \boldsymbol{u}_1 = \boldsymbol{u}_2$, then we obtain the model of *alternating VASS* [e.g. C9], which have an undecidable reachability problem.

A very interesting case occurs when we split $\boldsymbol{u}$ nondeterministically into $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ such that $\boldsymbol{u} = \boldsymbol{u}_1 + \boldsymbol{u}_2$. This model of *branching VASS* was introduced by Rambow [1994] in computational linguistics, and independently rediscovered on several occasions since [see the survey in C21]. The decidability of the reachability problem for branching VASS is a major open problem, that we further discuss in Section 6.4. As with pushdown VASS, although we do not know whether reachability is decidable, we have again a TOWER lower bound [J4].

5.4.3.3. *Data.* The model of *data nets* of Lazić, Newcomb, Ouaknine, Roscoe, and Worrell [2008] extends Petri nets with the ability to manipulate data from some infinite domain $\mathbb{D}$. Different variants exist, all with an undecidable reachability problem, except for one case: *unordered data Petri nets*, where the system can only manipulate data as pure names through equality and disequality constraints. In more concrete terms, the configurations of such a system no longer carry a single vector from $\mathbb{N}^d$, but a finite multiset of them, padded with infinitely many $\mathbf{0}$'s. Transitions nondeterministically select some (bounded) number of such vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$, and apply some translations $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$ from $\mathbb{Z}^d$ to each one. The decidability of the reachability problem for this model is open, with an ACKERMANN lower bound proven by Lazić and Totzke [2017].

CHAPTER 6

Perspectives

The techniques presented in this thesis provide a comprehensive set of tools for studying the computational complexity of algorithms relying on wqos for their termination: lower bounds through weak implementations of Hardy-style computations (Section 4.2), upper bounds using length function theorems for bad controlled sequences (Chapter 3), and suitable complexity classes for the non-elementary complexities usually encountered in this setting (Section 4.1). We have seen how these techniques can be applied to the safety verification of lossy counter machines, where they allow to prove the $\mathbf{F}_\omega$-completeness of the coverability problem (Theorems 4.8 and 4.9); this is the simplest case among the coverability problems solved using this standard toolbox (see Table 4.1). As an additional application of these techniques, we have been able to derive the first known complexity upper bounds for VASS Reachability in Chapter 5, a long-standing open question for this central problem with numerous ramifications in logic, automata, process calculi, etc. [see I2, Section 5].

I have already pointed to a number of open issues at the end of each chapter, two of which I recall next in Sections 6.1 and 6.2. However, some of the most interesting perspectives connected to the work in this thesis deal with *ideals*; those have already proven useful in several algorithmic applications of wqos, which I shall describe in Sections 6.3 and 6.4.

6.1. Complexity of VASS Reachability

The most important open question is arguably the exact complexity of VASS Reachability. Our $\mathbf{F}_{\omega^2}$ complexity bound certainly beats not having any upper bound at all, but it leaves a gigantic gap with the EXPSPACE lower bound of Lipton [1976]. As pointed in Section 5.4.1, a possibly easier question is to close the complexity gap for the KLMST decomposition algorithm, between $\mathbf{F}_\omega$ and $\mathbf{F}_{\omega^2}$; this would also allow to prove tight bounds for the Downward VASS Language Inclusion Problem.

PERSPECTIVE. Show an ACKERMANN upper bound on the size of the KLMST decomposition.

6.2. Parameterised Bounds

The bounds in Theorems 4.8 and 4.9 for LCM Coverability do not quite match when the number of counters is fixed. A similar issue arises with the upper bounds for lossy channel machines parameterised by the size of the alphabet, analysed in [C14]. The outcome is that there are no examples of 'natural' decision problems for most of the intermediate complexity classes $(\mathbf{F}_\alpha)_\alpha$. Refined

upper bound analyses seem required to close these gaps and prove for instance the $\mathbf{F}_{|C|}$-completeness of LCM Coverability with a fixed number of counters.

Tightening such enormous complexity bounds might seem of purely theoretical interest. These are worst-case complexity bounds, with arguably very little bearing on the running time of the actual algorithms when run on practical instances coming from abstractions of real programs and protocols [e.g. Kaiser et al., 2014, 2017]. But the question is likely to involve a deeper understanding of how counters, and more generally resources, interact in a system: can we finely capture these interactions through a better parameterisation than just the number of counters? Classical work in implicit complexity theory should be an inspiration here [e.g. Bellantoni and Cook, 1992; Kristiansen and Niggl, 2004], and might even lead to tractable bounds for the systems arising in practice [see Ben-Amram and Kristiansen, 2012; Brázdil et al., 2017; Zuleger, 2017, for recent related work].

Perspective. Invent parameterisations of counter machines leading to tight parametric bounds and tractable upper bounds on practical instances.

6.3. Algorithmic Applications of Ideals

Let us consider again Backward Coverability from §2.3.3.2. The algorithm relies on two main properties of wqos:

- the Ascending Chain Condition, which ensures the constructed sequence of upwards-closed sets is finite, and
- the Finite Basis Property, which allows us to finitely represent upwards-closed sets and manipulate them algorithmically.

Dually, a wqo is also equivalently characterised by the *Descending Chain Condition*: any descending sequence $D_0 \supseteq D_1 \supseteq D_2 \supseteq \cdots$ of downwards-closed subsets eventually stabilises. Hence, provided we could represent these downwards-closed subsets finitely, we could rely on this dual condition for termination. The right objects for this are *ideals*.

6.3.1. Ideals. An *ideal* of a qo $\langle A, \leq \rangle$ is a downwards-closed and *directed* subset $I \subseteq A$, where this last condition ensures that I is non-empty and that, given any $x \in I$ and $y \in I$, there exists $z \in I$ with $x \leq z$ and $y \leq z$. We write $\mathrm{Idl}(A)$ for the set of ideals of A. For instance, $\{0, \ldots, 4\}$ and the whole of $\mathbb{N}$ are ideals of $\mathbb{N}$; in fact, regarding the former, $\downarrow x$ for $x \in A$ is always an ideal of A, and called a *principal* one. While ideals are well-known objects in order theory, their application to verification problems is quite recent: they have been popularised in a series of articles initiated by Finkel and Goubault-Larrecq [Blondin et al., 2017a; Finkel and Goubault-Larrecq, 2009, 2012a] on a *forward* alternative to the backward coverability algorithm.

A related notion is the following. A downwards-closed subset D of $\langle A, \leq \rangle$ is *irreducible* if and only if it is non-empty, and for any two downwards-closed subsets D_1, D_2 such that $D \subseteq D_1 \cup D_2$, D is contained in D_1 or in D_2 already; equivalently, D is non-empty and cannot be written as the union of two proper, downwards-closed subsets.

Fact 6.1 (Bonnet, 1975, Lemma 1). *Let D be a downwards-closed subset of a qo. The following are equivalent:*

(1) *D is an ideal;*

(2) D *is directed;*
(3) D *is irreducible.*

Ideals are especially useful when $\langle A, \leq \rangle$ is a wqo: for one thing, when A is countable, $\mathrm{Idl}(A)$ is then also countable [Bonnet, 1975, Theorem 1], and furthermore any downwards-closed subset has a decomposition as a finite union of ideals. Recall that a qo has the *finite antichain property* (is FAC) if all its antichains are finite.

FACT 6.2 (Bonnet, 1975, Lemma 2). *A qo is FAC if and only if every downwards-closed subset is a finite union of ideals.*

Thanks to Fact 6.2, any downwards-closed set has a representation using finitely many ideals. Furthermore, by Fact 6.1, there is a unique minimal such decomposition, which we call the *canonical* ideal decomposition of D. Should we manage to find *effective* representations of wqo ideals, this will provide us with algorithmic means to manipulate downward-closed sets. This last endeavour is the subject of [Finkel and Goubault-Larrecq, 2009; Goubault-Larrecq et al., 2017]. Not only do most of the wqos mentioned in Section 2.1.2 (and several more) come with finite ideal representations, but furthermore these representations can be manipulated algorithmically: the inclusion relation $I \subseteq J$, and the canonical decompositions of A, of $\downarrow x$, of $I \cap J$, and of $A \setminus \uparrow x$ can all be computed for elements x and ideals I, J of A.

6.3.2. A Dual Backward Coverability Algorithm. As an example of application of ideals, one can formulate a dual of the Backward Coverability Algorithm relying on ideals and the Descending Chain Condition. This allows to prove upper bounds in some cases where the approach through controlled bad sequences has proven inconclusive; for instance, the set of configurations of a VASS is $Q \times \mathbb{N}^d$, and this is isomorphic to that of a lossy counter machine, thus the bounds one can extract from Theorem 3.15 for the backward coverability algorithm are the same for both formalisms. However, coverability is ACKERMANN-complete in LCM but only EXPSPACE-complete in VASS.

Consider an instance of the Coverability Problem on a system $\langle S, \to, \leq \rangle$ and two configurations $s, t \in S$. We compute instead the limit of the sequence

$$S \setminus \uparrow t = D_0 \supseteq D_1 \supseteq \cdots \text{ where } D_{n+1} \stackrel{\text{def}}{=} D_n \cap \mathrm{Pre}_\forall(D_n) \tag{6.1}$$

using the *universal* predecessor set

$$\mathrm{Pre}_\forall(D) \stackrel{\text{def}}{=} \{s \in S \mid \forall s' \in S, (s \to s' \implies s' \in D)\} \,. \tag{6.2}$$

This dual algorithm requires effective ideal representations and predecessors. It computes exactly the complements $D_n = S \setminus U_n$ of the elements in the sequence defined by Equation (2.9), and therefore terminates after the same number of steps. Thus it might seem that we have not gained anything by considering this variant.

It turns out however that considering the dual algorithm sometimes allows to exhibit 'monotonicity' invariants on the sets D_n, from which tighter upper bounds can be derived. In joint work with Lazić [W1], we have shown how to derive the EXPSPACE upper bound for VASS Coverability, together with the upper bounds of Table 6.1, in a generic manner. The known bounds for these problems were adaptations of Rackoff's ad-hoc analysis for VASS Coverability [Demri et al., 2013; C9; J4]. We further show in [C5] a new upper bound for a variant of

TABLE 6.1. The complexity of Coverability in more families of WSTS.

WSTS	Complexity	Reference
Bottom-up Branching VASS	2-EXP-c.	[Demri et al., 2013; W1]
Top-down Alternating VASS	2-EXP-c.	[C9; W1]
Top-down Branching VASS	TOWER-c.	[J4; W1]
ν-Petri nets	$\mathbf{F}_{\omega \cdot 2}$-c.	[C5]

data nets called ν-Petri nets. All the lower bounds in Table 6.1 rely on counter interfaces, which were presented succinctly in Section 4.3.1.

6.3.3. Beyond Well-Quasi-Orders. Ideals also appear in two extensions of wqos and well-structured transition systems.

The first one is to consider *Noetherian spaces*. A topological space is Noetherian if there are no infinite descending chains $C_0 \supsetneq C_1 \supsetneq \cdots$ of closed sets. This generalises well-partial-orders with the Alexandroff topology, which defines closed sets as the downwards-closed ones; however not every Noetherian space is the Alexandroff topology of a wpo. The notion of well-structured transition systems has been extended by Goubault-Larrecq [2007, 2010] to work on Noetherian spaces with a continuity condition instead of compatibility as defined in (2.6). For instance, polynomial automata as used by Benedikt et al. [2017] fall within this extended setup, while the complexity of their equivalence problem was analysed using the techniques mentioned just before in Section 6.3.2.

The second one is to work with *FAC quasi-orders*. As seen in Fact 6.2, all we need in order for finite ideal decompositions to exist is the absence of infinite antichains. Blondin et al. [2017c] have accordingly proposed a generalisation of well-structured transition systems based on FAC quasi-orders and the usual compatibility condition. While the backward coverability procedure does not always terminate on such systems (as the ascending chain condition does not hold), Coverability is nevertheless still decidable (under effectiveness conditions), thanks to two semi-decision procedures, one of which amounting to finding a downwards-closed *forward invariant*: a downwards-closed set containing the source configuration s and closed under the transition relation $\rightarrow$; this relies on ideal decompositions to enumerate candidate downwards-closed sets.

PERSPECTIVE. Design techniques for analysing the computational complexity of algorithms relying on Noetherian spaces and FAC orders.

Incidentally, there are relatively few examples of problems and classes of systems that can be tackled through Noetherian spaces and FAC orders, but not through classical WSTS algorithms, and this seems worth developing.

6.4. Reachability in VASS Extensions

We have seen a number of VASS extensions where the decidability of the reachability problem is still open in Table 5.2, where Branching VASS are of special importance. Ideals provide a potential attack to these problems, through extensions of the Decomposition Theorem proven with Leroux in [C8].

6.4.1. Branching VASS Reachability. Branching vector addition systems with states (BVASS) form a natural extension of VAS invented independently in several fields (see [C21] for a survey, in particular of the linguistic applications). The reachability problem for BVASS is a notorious open problem in theoretical computer science [Bojańczyk, 2014]. Several communities have indeed arrived to the same roadblock: it is equivalent to provability in multiplicative exponential linear logic [de Groote et al., 2004]—the sole fragment of propositional linear logic with unknown decidability status, open since 1990—it was also defined independently in computational linguistics [Rambow, 1994], cryptographic protocol verification [Verma and Goubault-Larrecq, 2005], verification of APIs for distributed computing [Bouajjani and Emmi, 2013], and a slight extension of it is also equivalent to satisfiability of data logics for XML processing [Jacquemard et al., 2016] and observational equivalence of functional program in ML [Cotton-Barratt et al., 2017]; finally, it has the dubious honour of having an incorrect published decidability proof [Bimbó, 2015]. While decidability is open, we know however that the problem has non-elementary complexity [J4]. Regarding fixed dimensions, only dimension 1 is known to be decidable [Figueira et al., 2017; Göller et al., 2016].

6.4.2. Ideal Decomposition of the Set of Executions. A very promising angle of attack to the problem is provided by the Decomposition Theorem in [C8]. This theorem shows that the celebrated algorithms for reachability in VASS developed in by Mayr [1981], Kosaraju [1982], and Lambert [1992] can be understood as computing a finite ideal decomposition of the downward-closure of the set of executions of a VASS.

Formally, we see an execution $(q_0, \boldsymbol{u}_0) \xrightarrow{t_1}_{\mathcal{V}} (q_1, \boldsymbol{u}_1) \xrightarrow{t_2}_{\mathcal{V}} \cdots \xrightarrow{t_n}_{\mathcal{V}} (q_n \boldsymbol{u}_n)$ from $c_0 \stackrel{\text{def}}{=} (q_0, \boldsymbol{u}_0)$ to $c_n \stackrel{\text{def}}{=} (q_n, \boldsymbol{u}_n)$ in a VASS $\mathcal{V}$ as a triple (c_0, σ, c_n) where σ is the sequence of triples $((q_i, \boldsymbol{u}_i), t_{i+1}, (q_{i+1}, \boldsymbol{u}_{i+1}))_{0 \leq i < n}$. Therefore an execution is an element of PreRuns $\stackrel{\text{def}}{=}$ *Confs* $\times$ (*Confs* $\times T \times$ *Confs*)$^* \times$ *Confs*. This set is naturally endowed with a well-quasi-odering $\trianglelefteq$ using Dickson's and Higman's lemmata [Jančar, 1990], and we can consider the downward-closure $\downarrow\text{Runs}_{\mathcal{V}}(c_0, c_n)$ of the set of executions from c_0 to c_n. By facts 6.1 and 6.2, this downwards-closed set has a unique finite decomposition into ideals.

THEOREM 6.3 (Decomposition Theorem; C8). *The KLMST decomposition algorithm computes a representation for the ideal decomposition of* $\downarrow\text{Runs}_{\mathcal{V}}(c_0, c_n)$.

The main ingredients of the proofs of Mayr, Kosaraju, and Lambert can be recast in this light: the manipulated 'marked witness graph sequences' turn out to be representations for ideals of a specific shape, the decomposition algorithm is an abstraction refinement loop that gets closer and closer to the desired downward-closure, and the termination criterion is an instance of *adherence membership*. This last condition, of a topological flavour, asks whether there exists executions of the system arbitrarily close to the limit denoted by the ideal, and is shown to be equivalent to the perfectness (aka θ) condition described in Section 5.2 for the ideals represented by marked witness graph sequences.

In a general setting, *adherence membership* can be formulated for finitely represented subsets $X \subseteq A$ of a wqo $\langle A, \leq \rangle$,—e.g. $\text{Runs}_{\mathcal{V}}(c_0, c_n) \subseteq$ PreRuns in the Decomposition Theorem, which is represented by the VASS $\mathcal{V}$ and the configurations c_0 and c_n—and finitely represented ideals I.

Problem (Adherence Membership).

instance: A subset $X \subseteq A$ over a wqo $\langle A, \leq \rangle$ and an ideal I of A.
question: Is I *adherent* to X, i.e. does there exist a directed subset $\Delta \subseteq X$ such that $I = {\downarrow}\Delta$?

A connection between adherence membership and computing downward-closures was exhibited in [C4]: under effectiveness assumptions, the problem of computing downward-closures of subsets X of A and the one of deciding adherence membership are Turing-equivalent [C4, Proposition 16]. However, these effectiveness assumptions do *not* hold in the case of VASS executions, and one can show that adherence membership is undecidable for VASS executions and *arbitrary* ideals of PreRuns [C8, Theorem V.2]. Nevertheless, this provides an essential insight into what the KLMST decomposition is really performing; a crucial point in the Decomposition Theorem is that we only need to handle the rather specific ideals that arise during the decomposition, which are ideals represented by marked witness graph sequences.

Extending this approach to handle the branching executions of BVASS is a very challenging endeavour, but at least we have now a general framework in which we can attack the problem.

Perspective. Extend the framework of the Decomposition Theorem to solve BVASS Reachability.

As related problems, the decidability of reachability is also open in unordered data Petri nets and pushdown VASS, and could be approached from the same angle (c.f. Table 5.2). In fact, the decidability proof of Reinhardt [2008] for VASS with a single zero test or hierarchical zero tests also seems approachable with these techniques. In turn, this approach motivates further investigations of the basic properties, finite representations, and algorithmics of ideals and of the adherence membership problem.

APPENDIX A

Technical Appendix

A.1. Ordinals

From a set-theoretic perspective, an *ordinal* is an equivalence class of well-orders up to order isomorphism. We also employ ordinals, or rather ordinal terms denoting ordinals, in order to define subrecursive functions in Section 3.2.

A.1.1. Ordinals Terms. Ordinals in ε_0 can be canonically represented as *ordinal terms* α abstract syntax in Cantor normal form

$$\alpha = \omega^{\alpha_1} + \cdots + \omega^{\alpha_p} \tag{CNF}$$

with *exponents* $\alpha > \alpha_1 \geq \cdots \geq \alpha_p$. We write as usual 1 for the term ω^0 and ω for the term ω^1. Grouping equal exponents yields the strict form

$$\alpha = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_p} \cdot c_p$$

with $\alpha > \alpha_1 > \cdots > \alpha_p$ and *coefficients* $0 < c_1, \ldots, c_p < \omega$. The ordinal ε_0, i.e. the least solution of $\omega^x = x$, is the supremum of the ordinals presentable in this manner.

A.1.1.1. *Operations on Ordinal Terms.* Thanks to the bijection between ordinal terms in CNF and ordinals below ε_0, the usual ordinal operations can also be expressed 'syntactically' on ordinal terms. Later we will use terms as representatives for isomorphism classes of well orders with order types below ε_0. We therefore include these syntactic characterisations here for the sake of completeness.

The ordinal ordering has a syntactic characterisation for ordinal terms in CNF: $\alpha = \sum_{i=1}^{p} \omega^{\alpha_i} < \beta = \sum_{j=1}^{m} \omega^{\beta_j}$ if and only if there exists $1 \leq j \leq m$ such that $\alpha_i = \beta_i$ for all $1 \leq i < j$ and either $j \leq p$ and $\alpha_j < \beta_j$ or $j > p$.

The *direct sum* $\alpha + \beta$ of two ordinals can also be defined on their CNFs $\alpha = \sum_{i=1}^{p} \omega^{\alpha_i}$ and $\beta = \sum_{j=1}^{m} \omega^{\beta_j}$: thanks to the associativity of $+$, it suffices to consider $\omega^{\alpha_p} + \omega^{\beta_1}$, which is already in CNF if $\alpha_p \geq \beta_1$, and is otherwise equated with ω^{β_1}. For instance $\omega^2 + \omega + \omega^3 = \omega^2 + \omega^3 = \omega^3$; note that the direct sum is not commutative. Similarly, the *direct product* $\alpha \cdot \beta$ of $\alpha = \sum_{i=1}^{p} \omega^{\alpha_i}$ and $\beta = \sum_{j=1}^{m} \omega^{\beta_j}$ can be defined thanks to left distributivity as $\sum_{j=1}^{m} \alpha \cdot \omega^{\beta_j}$, where in turn $\alpha \cdot \omega^{\beta_j} = \omega^{\alpha_1 + \beta_j}$ if $\beta_j > 0$ and $\alpha \cdot 1 = \alpha$ otherwise. For instance $2 \cdot \omega = \omega$ and $(\omega + 2) \cdot \omega = \omega^2$; note that direct products are neither commutative nor right distributive.

The *natural sum* $\alpha \oplus \beta$ of two ordinals with CNFs $\alpha = \sum_{i=1}^{p} \omega^{\alpha_i}$ and $\beta = \sum_{j=1}^{m} \omega^{\beta_j}$ can be defined as $\omega^{\gamma_1} + \cdots + \omega^{\gamma_{p+m}}$ where the exponents $\gamma_1 \geq \cdots \geq \gamma_{p+m}$ are a reordering of $\alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_m$. Their *natural product* $\alpha \otimes \beta$ is $\bigoplus_{1 \leq i \leq p, 1 \leq j \leq m} \omega^{\alpha_i \oplus \beta_j}$. Unlike the direct operations, the natural operations are

commutative and distributive: $(\omega^2+\omega)\oplus\omega^3=\omega^3+\omega^2+\omega$, $2\otimes\omega=\omega\cdot 2$, and $(\omega+2)\otimes\omega=\omega^2+\omega\cdot 2$.

A.1.1.2. *Computing Order Types.* The order types $o(A,\leq_A)$ of the well orders $\langle A,\leq_A\rangle$ we already mentioned in this document are well-known: $o([d],\leq)=d$, $o(\mathbb{N},\leq)=\omega$, $o(A\times B,\leq_{\text{lex}})=o(B,\leq_B)\cdot o(A,\leq_A)$, and $o(\mathbb{M}(A),\leq_{\text{m}})=\omega^{o(A,\leq_A)}$.

By extension, we also write $o(x,\leq)$ for the ordinal term in $o(A,\leq)$ associated to an element x in A. Formally, this is the order type of the restriction of A to elements no larger than x: $o(x,\leq)\stackrel{\text{def}}{=}o(\{y\in A\mid x\not\leq y\},\leq)$. For instance in $\langle\mathbb{N}^d,\leq_{\text{lex}}\rangle$, $o((n_1,\ldots,n_d),\leq_{\text{lex}})=\omega^{d-1}\cdot n_1+\cdots+\omega\cdot n_{d-1}+n_d$.

A.1.2. Maximal Order Types. De Jongh and Parikh [1977] extend the notion of order types to wpos $\langle A,\leq\rangle$:

$$o(A,\leq)\stackrel{\text{def}}{=}\sup\{o(A,\preceq)\mid\preceq\text{ is a linearisation of }\leq\}\ . \tag{A.1}$$

This is called the *maximal order type* of $\langle A,\leq\rangle$, since de Jongh and Parikh show that there always exists a linearisation $\langle A,\preceq\rangle$ that reaches this order type, i.e. 'sup' can be replaced by 'max' in (A.1) (Blass and Gurevich [2008] also call this the *stature* of $\langle A,\leq\rangle$). The notion extends to wqos by considering the associated quotient wpos.

The maximal order types of the wqos used in this document can be computed using $o(Q,=)=|Q|$ for Q finite, $o(A\sqcup B,\leq_\sqcup)=o(A,\leq_A)\oplus o(B,\leq_B)$, $o(A\times B,\leq_\times)=o(A,\leq_A)\otimes o(B,\leq_B)$ [de Jongh and Parikh, 1977]. Regarding Higman's Lemma and Kruskal's Theorem, the maximal order types were computed by Schmidt [1979], while the case of Robertson and Seymour's Graph Minor Theorem is still unsettled—but see [Van der Meeren, 2015] for recent advances.

Maximal order types provide a measure of the 'strength' of wqos. They might however conceal some important distinctions; for instance $o(\mathbb{N}^d,\leq_\times)=\omega^d=o(\mathbb{N}^d,\leq_{\text{lex}})$.

A.2. Subrecursive Functions

A.2.1. Monotonicity Properties. Assume h is monotone and inflationary. Then both h^α and h_α are monotone and inflationary [see Cichoń and Tahhan Bittar, 1998; Schwichtenberg and Wainer, 2012; L1]. However, those hierarchies are not monotone in the ordinal indices: for instance, $H^\omega(x)=2x+1<2x+2=H^{x+2}(x)$ although $\omega>x+2$.

Some refinement of the ordinal ordering is needed in order to obtain monotonicity of the hierarchies. Define for this the *pointwise ordering* $\prec_x$ at some x in $\mathbb{N}$ as the smallest transitive relation such that

$$\alpha\prec_x\alpha+1\ ,\qquad\qquad\lambda(x)\prec_x\lambda\ . \tag{A.2}$$

The relation '$\beta\prec_x\alpha$' is noted '$\beta\in\alpha[x]$' in [Schwichtenberg and Wainer, 2012, pp. 158–163]. The $\prec_x$ relations form a strict hierarchy of refinements of the ordinal ordering $<$:

$$\prec_0\subsetneq\prec_1\subsetneq\cdots\subsetneq\prec_x\subsetneq\cdots\subsetneq<\ . \tag{A.3}$$

As desired, our hierarchies are monotone for the pointwise ordering [Cichoń and Tahhan Bittar, 1998; Schwichtenberg and Wainer, 2012; L1]:

(A.4) $\beta \prec_x \alpha$ implies $h_\beta(x) \leq h_\alpha(x)$ and $h^\beta(x) \leq h^\alpha(x)$.

A.2.1.1. *Ordinal Norms.* Recall the ordinal norm defined in Equation (3.4): the *norm* of an ordinal as the maximal coefficient that appears in its associated CNF, i.e. if $\alpha = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_p} \cdot c_p$ with $\alpha_1 > \cdots > \alpha_p$ and $c_1, \ldots, c_p > 0$, then

$$N\alpha \stackrel{\text{def}}{=} \max\{c_1, \ldots, c_p, N\alpha_1, \ldots, N\alpha_p\} .$$

The relation between ordinal norms and the pointwise ordering is that [Schwichtenberg and Wainer, 2012; L1]

(A.5) $\beta < \alpha$ implies $\beta \prec_{N\beta} \alpha$.

Together with (A.3) and (A.4), this entails the following statement.

Lemma A.1 (Eventual Majoration). *Let $\beta < \alpha$ be two ordinals and h be a monotone inflationary function. Then, for all $x \geq N\beta$, $h_\beta(x) \leq h_\alpha(x)$ and $h^\beta(x) \leq h^\alpha(x)$.*

References

Abdulla, P.A. and Jonsson, B., 1996. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101. doi:10.1006/inco.1996.0053. Cited on pages 15 and 17.

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 pages 1, 2, 9, 15, and 17.

Abdulla, P.A., 2010. Well (and better) quasi-ordered transition systems. *Bulletin of Symbolic Logic*, 16(4):457–515. doi:10.2178/bsl/1294171129. Cited on page 15.

Abdulla, P.A., Mayr, R., Sangnier, A., and Sproston, J., 2013. Solving parity games on integer vectors. In *Proceedings of Concur 2013*, volume 8052 of *Lecture Notes in Computer Science*, pages 106–120. Springer. doi:10.1007/978-3-642-40184-8_9. Cited on page 7.

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 32 and 33.

Alechina, N., Bulling, N., Demri, S., and Logan, B., 2016. On the complexity of resource-bounded logics. In *Proceedings of RP 2016*, volume 9899 of *Lecture Notes in Computer Science*, pages 36–50. Springer. doi:10.1007/978-3-319-45994-3_3. Cited on page 7.

Alias, C., Darte, A., Feautrier, P., and Gonnord, L., 2010. Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In *Proceedings of SAS 2010*, volume 6337 of *Lecture Notes in Computer Science*, pages 117–133. Springer. doi:10.1007/978-3-642-15769-1_8. Cited on page 31.

Alur, R. and Dill, D.L., 1994. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235. doi:10.1016/0304-3975(94)90010-8. Cited on page 1.

Arnold, A. and Latteux, M., 1978. Récursivité et cones rationnels fermés par intersection. *CALCOLO*, 15(4):381–394. doi:10.1007/BF02576519. Cited on page 17.

Atig, M.F. and Habermehl, P., 2011. On Yen's path logic for Petri nets. *International Journal of Fundations of Computer Science*, 22(4):783–799. doi:10.1142/S0129054111008428. Cited on page 66.

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 70.

Ben-Amram, A.M., 2002. General size-change termination and lexicographic descent. In *The Essence of Computation*, volume 2566 of *Lecture Notes in Computer Science*, pages 3–17. Springer. doi:10.1007/3-540-36377-7_1. Cited on pages 26

and 31.

Ben-Amram, A.M. and Kristiansen, L., 2012. On the edge of decidability in complexity analysis of loop programs. *International Journal of Fundations of Computer Science*, 23(7):1451–1464. doi:10.1142/S0129054112400588. Cited on page 70.

Ben-Amram, A.M. and Genaim, S., 2014. Ranking functions for linear-constraint loops. *Journal of the ACM*, 61(4:26). doi:10.1145/2629488. Cited on page 13.

Benedikt, M., Duff, T., Sharad, A., and Worrell, J.B., 2017. Polynomial automata: Zeroness and applications. In *Proceedings of LICS 2017*. IEEE. doi:10.1109/LICS.2017.8005101. Cited on pages 4, 33, and 72.

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 17.

Bimbó, K., 2015. The decidability of the intensional fragment of classical linear logic. *Theoretical Computer Science*, 597:1–17. doi:10.1016/j.tcs.2015.06.019. Cited on page 73.

Blass, A. and Gurevich, Y., 2008. Program termination and well partial orderings. *ACM Transactions on Computational Logic*, 9(3). doi:10.1145/1352582.1352586. Cited on pages 31 and 76.

Blondin, M., Finkel, A., Göller, S., Haase, C., and McKenzie, P., 2015. Reachability in two-dimensional vector addition systems with states is PSPACE-complete. In *Proceedings of LICS 2015*, pages 32–43. IEEE. doi:10.1109/LICS.2015.14. Cited on pages 6, 65, and 66.

Blondin, M., Finkel, A., and Goubault-Larrecq, J., 2017a. Forward analysis for WSTS, part III: Karp-Miller trees. In *Proceedings of FSTTCS 2017*, *Leibniz International Proceedings in Informatics*. LZI. To appear. Cited on page 70.

Blondin, M., Finkel, A., Haase, C., and Haddad, S., 2017b. The logical view on continuous Petri nets. *ACM Transactions on Computational Logic*, 18(3:24):1–28. doi:10.1145/3105908. Cited on pages 2 and 66.

Blondin, M., Finkel, A., and McKenzie, P., 2017c. Well behaved transition systems. *Logical Methods in Computer Science*, 13(3:24):1–19. doi:10.23638/LMCS-13(3:24)2017. Cited on page 72.

Bojańczyk, M., David, C., Muscholl, A., Schwentick, T., and Segoufin, L., 2011. Two-variable logic on data words. *ACM Transactions on Computational Logic*, 12(4:27):1–26. doi:10.1145/1970398.1970403. Cited on pages 5 and 53.

Bojańczyk, M., 2014. Some open problems in automata and logic. *ACM SIGLOG News*, 1(2):3–12. doi:10.1145/2677161.2677163. Cited on page 73.

Bonfante, G., Cichoń, A.E., Marion, J.Y., and Touzet, H., 2001. Algorithms with polynomial interpretation termination proof. *Journal of Functional Programming*, 11:33–53. Cited on page 31.

Bonnet, R., 1975. On the cardinality of the set of initial intervals of a partially ordered set. In *Infinite and finite sets: to Paul Erdős on his 60th birthday, Vol. 1*, *Colloquia Mathematica Societatis János Bolyai*, pages 189–198. North-Holland. Cited on pages 70 and 71.

Bonnet, R., 2013. *Theory of Well-Structured Transition Systems and Extended Vector-Addition Systems*. Thèse de doctorat, ENS Cachan. Cited on pages 65 and 67.

Bouajjani, A. and Emmi, M., 2013. Analysis of recursively parallel programs. *ACM Transactions on Programming Languages and Systems*, 35(3:10):1–49. doi:10.1145/2518188. Cited on page 73.

Brázdil, T., Jančar, P., and Kučera, A., 2010. Reachability games on extended vector addition systems with states. In *Proceedings of ICALP 2010*, volume 6199 of *Lecture Notes in Computer Science*, pages 478–489. Springer. doi:10.1007/978-3-642-14162-1_40. Cited on page 7.

Brázdil, T., Chatterjee, K., Kučera, A., Novotnỳ, P., and Velan, D., 2017. Efficient algorithms for checking fast termination in VASS. arXiv:1708.09253 [cs.LO]. Cited on page 70.

Bresolin, D., Della Monica, D., Montanari, A., Sala, P., and Sciavicco, G., 2012. Interval temporal logics over finite linear orders: The complete picture. In *Proceedings of ECAI 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 199–204. IOS. doi:10.3233/978-1-61499-098-7-199. Cited on page 2.

Brockschmidt, M., Cook, B., Ishtiaq, S., Khlaaf, H., and Piterman, N., 2016. T2: temporal property verification. In *Proceedings of TACAS 2016*, volume 9636 of *Lecture Notes in Computer Science*, pages 387–393. Springer. doi:10.1007/978-3-662-49674-9_22. Cited on page 13.

Buchholz, W., Cichoń, E.A., and Weiermann, A., 1994. A uniform approach to fundamental sequences and hierarchies. *Mathematical Logic Quaterly*, 40(2):273–286. doi:10.1002/malq.19940400212. Cited on pages 3, 24, 25, and 31.

Buchholz, W., 1995. Proof-theoretic analysis of termination proofs. *Annals of Pure and Applied Logic*, 75(1–2):57–65. doi:10.1016/0168-0072(94)00056-9. Cited on page 31.

Chambart, P. and Schnoebelen, Ph., 2008. The ordinal recursive complexity of lossy channel systems. In *Proceedings of LICS 2008*, pages 205–216. IEEE. doi:10.1109/LICS.2008.47. Cited on pages 4, 41, and 49.

Cichoń, E.A., 1993. Termination orderings and complexity characterisations. In *Proof Theory*, pages 171–194. Cambridge University Press. doi:10.1017/CBO9780511896262.008. Cited on pages 24, 25, and 31.

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 3, 21, 23, 24, 32, 76, and 77.

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 page 32.

Colcombet, T., Daviaud, L., and Zuleger, F., 2014. Size-change abstraction and max-plus automata. In *Proceedings of MFCS 2014*, volume 8634 of *Lecture Notes in Computer Science*, pages 208–219. Springer. doi:10.1007/978-3-662-44522-8_18. Cited on page 26.

Cook, B., Podelski, A., and Rybalchenko, A., 2006. Termination proofs for systems code. In *Proceedings of PLDI 2006*. ACM. doi:10.1145/1133981.1134029. Cited

on page 14.

Cook, B., See, A., and Zuleger, F., 2013. Ramsey vs. lexicographic termination proving. In *Proceedings of TACAS 2013*, volume 7795 of *Lecture Notes in Computer Science*, pages 47–61. doi:10.1007/978-3-642-36742-7_4. Cited on page 13.

Cotton-Barratt, C., Murawski, A., and Ong, L., 2017. ML and extended branching VASS. In *Proceedings of ESOP 2017*, volume 10201 of *Lecture Notes in Computer Science*, pages 314–340. Springer. doi:10.1007/978-3-662-54434-1_12. Cited on page 73.

Courant, N. and Urban, C., 2017. Precise widening operators for proving termination by abstract interpretation. In *Proceedings of TACAS 2017*, volume 10205 of *Lecture Notes in Computer Science*, pages 136–152. Springer. doi:10.1007/978-3-662-54577-5_8. Cited on page 13.

Crespi-Reghizzi, S. and Mandrioli, D., 1977. Petri nets and Szilard languages. *Information and Control*, 33(2):177–192. doi:10.1016/S0019-9958(77)90558-7. Cited on page 53.

Decker, N., Habermehl, P., Leucker, M., and Thoma, D., 2014. Ordered navigation on multi-attributed data words. In *Proceedings of Concur 2014*, volume 8704 of *Lecture Notes in Computer Science*, pages 497–511. Springer. doi:10.1007/978-3-662-44584-6_34. Cited on page 67.

Decker, N. and Thoma, D., 2016. On freeze LTL with ordered attributes. In *Proceedings of 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 pages 4, 49, and 67.

de Groote, Ph., Guillaume, B., and Salvati, S., 2004. Vector addition tree automata. In *Proceedings of LICS 2004*, pages 64–73. IEEE. doi:10.1109/LICS.2004.51. Cited on page 73.

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 pages 1, 29, and 76.

Delzanno, G., Raskin, J.F., and Van Begin, L., 2001. Attacking symbolic state explosion. In *Proceedings of CAV 2011*, volume 2102 of *Lecture Notes in Computer Science*, pages 298–310. Springer. doi:10.1007/3-540-44585-4_28. Cited on page 2.

Delzanno, G., Sangnier, A., and Zavattaro, G., 2010. Parameterized verification of ad hoc networks. In *Proceedings of Concur 2010*, volume 6269 of *Lecture Notes in Computer Science*, pages 313–327. Springer. doi:10.1007/978-3-642-15375-4_22. Cited on page 33.

Demri, S., 2013. On selective unboundedness of VASS. *Journal of Computer and System Sciences*, 79(5):689–713. doi:10.1016/j.jcss.2013.01.014. Cited on page 66.

Demri, S., Figueira, D., and Praveen, M., 2016. Reasoning about data repetitions with counter systems. *Logical Methods in Computer Science*, 12(3:1):1–54. doi:10.2168/LMCS-12(3:1)2016. Cited on pages 5 and 53.

Demri, S., Jurdziński, M., Lachish, O., and Lazić, R., 2013. 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 pages 71 and 72.

Dershowitz, N. and Manna, Z., 1979. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476. doi:10.1145/359138.359142. Cited on pages 11, 26, 60, and 61.

Dickson, L.E., 1913. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422. doi:10.2307/2370405. Cited on pages 3 and 11.

Ding, G., 1992. Subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 16 (5):489–502. doi:10.1002/jgt.3190160509. Cited on page 33.

D'Osualdo, E., Ong, L., and Tiu, A., 2017. Deciding secrecy of security protocols for an unbounded number of sessions: The case of depth-bounded processes. In *Proceedings of CSF 2017*. IEEE. To appear. Cited on page 32.

Englert, M., Lazić, R., and Totzke, P., 2016. Reachability in two-dimensional unary vector addition systems with states is NL-complete. In *Proceedings of LICS 2016*, pages 477–484. ACM. doi:10.1145/2933575.2933577. Cited on pages 6, 65, and 66.

Esparza, J. and Nielsen, M., 1994. Decidability issues for Petri nets — a survey. *Bulletin of the EATCS*, 52:244–262. Cited on page 53.

Esparza, J., 1998. Decidability and complexity of Petri net problems — an introduction. In *Proceedings of Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer. doi:10.1007/3-540-65306-6_20. Cited on pages 53 and 63.

Esparza, J., Ledesma-Garza, R., Majumdar, R., Meyer, P., and Niksic, F., 2014. An SMT-based approach to coverability analysis. In *Proceedings of CAV 2014*, volume 8559 of *Lecture Notes in Computer Science*, pages 603–619. Springer. doi:10.1007/978-3-319-08867-9_40. Cited on pages 2 and 66.

Esparza, J., Ganty, P., Leroux, J., and Majumdar, R., 2017. Verification of population protocols. *Acta Informatica*, 54(2):191–215. doi:10.1007/s00236-016-0272-3. Cited on page 53.

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 page 5.

Fairtlough, M.V.H. and Wainer, S.S., 1998. Hierarchies of provably recursive functions. In *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 5, 23, 24, 36, 37, 41, and 50.

Fearnley, J. and Jurdziński, M., 2015. Reachability in two-clock timed automata is PSPACE-complete. *Information and Computation*, 243:26–36. doi:10.1016/j.ic.2014.12.004. Cited on pages 65 and 66.

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 2.

Figueira, D., Lazic, R., Leroux, J., Mazowiecki, F., and Sutre, G., 2017. Polynomial-space completeness of reachability for succinct branching VASS in dimension one. In *Proceedings of ICALP 2017*, volume 80 of *Leibniz International Proceedings in Informatics*, article 119, 14 pages. LZI. doi:10.4230/LIPIcs.ICALP.2017.119. Cited on page 73.

Finkel, A., 1987. A generalization of the procedure of Karp and Miller to well structured transition systems. In *Proceedings of ICALP 1987*, volume 267 of *Lecture Notes in Computer Science*, pages 499–508. Springer. doi:10.1007/3-540-18088-5_43. Cited on page 15.

Finkel, A., 1994. Decidability of the termination problem for completely specificied protocols. *Distributed Computing*, 7(3):129–135. doi:10.1007/BF02277857. Cited on page 15.

Finkel, A. and Schnoebelen, Ph., 2001. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92. doi:10.1016/S0304-3975(00)00102-X. Cited on pages 1, 2, 9, 15, and 17.

Finkel, A. and Goubault-Larrecq, J., 2009. Forward analysis for WSTS, part I: Completions. In *Proceedings of STACS 2009*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 433–444. LZI. doi:10.4230/LIPIcs.STACS.2009.1844. Cited on pages 70 and 71.

Finkel, A. and Goubault-Larrecq, J., 2012a. Forward analysis for WSTS, part II: Complete WSTS. *Logical Methods in Computer Science*, 8(3:28). doi:10.2168/LMCS-8(3:28)2012. Cited on page 70.

Finkel, A. and Goubault-Larrecq, J., 2012b. The theory of WSTS: the case of complete WSTS. In *Proceedings of Petri Nets 2012*, volume 7347 of *Lecture Notes in Computer Science*, pages 3–31. Springer. doi:10.1007/978-3-642-31131-4_2. Cited on page 15.

Fischer, P.C., Meyer, A.R., and Rosenberg, A.L., 1968. Counter machines and counter languages. *Mathematical Systems Theory*, 2(3):265–283. doi:10.1007/BF01694011. Cited on page 45.

Floyd, R.W., 1967. Assigning meaning to programs. In *Proceedings of Mathematical Aspects of Computer Science*, volume 19 of *Proceedings of Symposia in Applied Mathematics*, pages 19–32. AMS. Cited on page 12.

Fraigniaud, P., Rajsbaum, S., and Travers, C., 2016. Minimizing the number of opinions for fault-tolerant distributed decision using well-quasi orderings. In *Proceedings of LATIN 2016*, volume 9644 of *Lecture Notes in Computer Science*, pages 497–508. Springer. doi:10.1007/978-3-662-49529-2_37. Cited on page 32.

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 32.

Ganian, R., Hliněný, P., Nešetřil, J., Obdržálek, J., Ossona de Mendez, P., and Ramadurai, R., 2012. When trees grow low: Shrubs and fast MSO_1. In *Proceedings of MFCS 2012*, volume 7464 of *Lecture Notes in Computer Science*, pages 419–430. Springer. doi:10.1007/978-3-642-32589-2_38. Cited on page 33.

Ganty, P. and Majumdar, R., 2012. Algorithmic verification of asynchronous programs. *ACM Transactions on Programming Languages and Systems*, 34(1:6):1–48. doi:10.1145/2160910.2160915. Cited on pages 5 and 53.

Geffroy, T., Leroux, J., and Sutre, G., 2016. Occam's Razor applied to the Petri net coverability problem. In *Proceedings of RP 2016*, volume 9899 of *Lecture Notes in Computer Science*, pages 77–89. Springer. doi:10.1007/978-3-319-45994-3_6. Cited on pages 2 and 66.

Geffroy, T., Leroux, J., and Sutre, G., 2017. Backward coverability with pruning for lossy channel systems. In *Proceedings of SPIN 2017*, pages 132–141. ACM. doi:10.1145/3092282.3092292. Cited on page 2.

Genest, B., Muscholl, A., Serre, O., and Zeitoun, M., 2008. Tree pattern rewriting systems. In *Proceedings of ATVA 2008*, volume 5311 of *Lecture Notes in Computer Science*, pages 332–346. doi:10.1007/978-3-540-88387-6_29. Cited on page 32.

Geeraerts, G., Raskin, J.F., and Van Begin, L., 2005. Expand, enlarge and check... made efficient. In *Proceedings of CAV 2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 394–407. Springer. doi:10.1007/11513988_38. Cited on page 2.

German, S.M. and Sistla, A.P., 1992. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735. doi:10.1145/146637.146681. Cited on page 53.

Geser, A., 1990. *Relative Termination*. Ph.D. Thesis, Universität Passau. doi:10.18725/OPARU-2427. Cited on page 14.

Gischer, J., 1981. Shuffle languages, Petri nets, and context-sensitive grammars. *Communications of the ACM*, 24(9):597–605. doi:10.1145/358746.358767. Cited on page 53.

Göller, S., Haase, C., Lazic, R., and Totzke, P., 2016. A polynomial-time algorithm for reachability in branching VASS in dimension one. In *Proceedings of ICALP 2016*, volume 55 of *Leibniz International Proceedings in Informatics*, article 105, 13 pages. LZI. doi:10.4230/LIPIcs.ICALP.2016.105. Cited on page 73.

Goubault-Larrecq, J., 2007. On Noetherian spaces. In *Proceedings of LICS 2007*, pages 453–462. IEEE. doi:10.1109/LICS.2007.34. Cited on page 72.

Goubault-Larrecq, J., 2010. Noetherian spaces in verification. In *Proceedings of ICALP 2010*, volume 6199 of *Lecture Notes in Computer Science*, pages 2–21. Springer. doi:10.1007/978-3-642-14162-1_2. Cited on page 72.

Goubault-Larrecq, J., Halfon, S., Karandikar, P., Narayan Kumar, K., and Schnoebelen, Ph., 2017. The ideal approach to computing closed subsets in well-quasi-orderings. In preparation. Cited on page 71.

Greibach, S.A., 1978. Remarks on blind and partially blind one-way multi-counter machines. *Theoretical Computer Science*, 7(3):311–324. doi:10.1016/0304-3975(78)90020-8. Cited on page 56.

Grzegorczyk, A., 1953. Some classes of recursive functions. *Rozprawy Matematyczne*, 4. Cited on page 5.

Gulwani, S., 2009. SPEED: Symbolic complexity bound analysis. In *Proceedings of CAV 2009*, volume 5643 of *Lecture Notes in Computer Science*, pages 51–62.

Springer. doi:10.1007/978-3-642-02658-4_7. Cited on page 31.

Haase, C., Kreutzer, S., Ouaknine, J., and Worrell, J., 2009. Reachability in succinct and parametric one-counter automata. In *Proceedings of Concur 2009*, volume 5710 of *Lecture Notes in Computer Science*, pages 369–383. Springer. doi:10.1007/978-3-642-04081-8_25. Cited on pages 65 and 66.

Habermehl, P., Meyer, R., and Wimmel, H., 2010. The downward-closure of Petri net languages. In *Proceedings of ICALP 2010*, volume 6199 of *Lecture Notes in Computer Science*, pages 466–477. Springer. doi:10.1007/978-3-642-14162-1_39. Cited on pages 64 and 65.

Hack, M.H.T., 1975. *Decidability questions for Petri nets.* Ph.D. Thesis, MIT. Cited on pages 64 and 65.

Higman, G., 1952. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(2):326–336. doi:10.1112/plms/s3-2.1.326. Cited on pages 1, 3, and 11.

Hirokawa, N. and Moser, G., 2008. Automated complexity analysis based on the dependency pair method. In *Proceedings of IJCAR 2008*, volume 5195 of *Lecture Notes in Computer Science*, pages 364–379. Springer. doi:10.1007/978-3-540-71070-7_32. Cited on page 31.

Hofbauer, D., 1992. Termination proofs by multiset path orderings imply primitive recursive derivation lengths. *Theoretical Computer Science*, 105(1):129–140. doi:10.1016/0304-3975(92)90289-R. Cited on page 31.

Hopcroft, J.E. and Pansiot, J.J., 1979. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8:135–159. doi:10.1016/0304-3975(79)90041-0. Cited on pages 54, 56, 66, and 67.

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 66.

Jacquemard, F., Segoufin, L., and Dimino, J., 2016. $\mathrm{FO}^2(<, +1, \sim)$ on data trees, data tree automata and branching vector addition systems. *Logical Methods in Computer Science*, 12(2:3). doi:10.2168/LMCS-12(2:3)2016. Cited on page 73.

Jančar, P., 1990. Decidability of a temporal logic problem for Petri nets. *Theoretical Computer Science*, 74(1):71–93. doi:10.1016/0304-3975(90)90006-4. Cited on pages 6 and 73.

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 41, 64, and 65.

Jančar, P., 2008. Bouziane's transformation of the Petri net reachability problem and incorrectness of the related algorithm. *Information and Computation*, 206 (11):1259–1263. doi:10.1016/j.ic.2008.06.003. Cited on page 53.

Jouannaud, J.P. and Lescanne, P., 1982. On multiset orderings. *Information Processing Letters*, 15(2):57–63. doi:10.1016/0020-0190(82)90107-7. Cited on page 11.

Kaiser, A., Kroening, D., and Wahl, T., 2014. A widening approach to multi-threaded program verification. *ACM Transactions on Programming Languages*

and Systems, 36(4:14):1–29. doi:10.1145/2629608. Cited on pages 2, 66, and 70.

Kaiser, A., Kroening, D., and Wahl, T., 2017. Lost in abstraction: Monotonicity in multi-threaded programs. *Information and Computation*, 252:30–47. doi:10.1016/j.ic.2016.03.003. Cited on pages 2 and 70.

Kanovich, M.I., 1995. Petri nets, Horn programs, linear logic and vector games. *Annals of Pure and Applied Logic*, 75(1–2):107–135. doi:10.1016/0168-0072(94)00060-G. Cited on pages 7 and 53.

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 56, 60, and 62.

Ketonen, J. and Solovay, R., 1981. Rapidly growing Ramsey functions. *Annals of Mathematics*, 113(2):27–314. doi:10.2307/2006985. Cited on pages 3, 32, and 50.

König, B. and Stückrath, J., 2017. Well-structured graph transformation systems. *Information and Computation*, 252:71–94. doi:10.1016/j.ic.2016.03.005. Cited on page 33.

Kopylov, A.P., 1995. Decidability of linear affine logic. In *Proceedings of LICS 1995*, pages 496–504. IEEE. doi:10.1109/LICS.1995.523283. Cited on page 7.

Kosaraju, S.R., 1982. Decidability of reachability in vector addition systems. In *Proceedings of STOC 1982*, pages 267–281. ACM. doi:10.1145/800070.802201. Cited on pages 5, 6, 53, 54, 57, 58, 60, and 73.

Kreisel, G., 1952. On the interpretation of non-finitist proofs–Part II. *Journal of Symbolic Logic*, 17(1):43–58. doi:10.2307/2267457. Cited on page 31.

Kripke, S.A., 1959. The problem of entailment. In *Proceedings of ASL 1959*, *Journal of Symbolic Logic*, page 324. doi:10.2307/2963903. Abstract. Cited on page 7.

Kristiansen, L. and Niggl, H., 2004. On the computational complexity of imperative programming languages. *Theoretical Computer Science*, (1–2):139–161. doi:10.1016/j.tcs.2003.10.016. Cited on page 70.

Kroening, D., Sharygina, N., Tsitovich, A., and Wintersteiger, C.M., 2010. Termination analysis with compositional transition invariants. In *Proceedings of CAV 2010*, volume 6174 of *Lecture Notes in Computer Science*, pages 89–103. Springer. doi:10.1007/978-3-642-14295-6_9. Cited on page 14.

Kruskal, J.B., 1960. Well-quasi-ordering, the Tree Theorem, and Vazsonyi's Conjecture. *Transactions of the American Mathematical Society*, 95(2):210–225. doi:10.2307/1993287. Cited on page 11.

Kruskal, J.B., 1972. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13(3):297–305. doi:10.1016/0097-3165(72)90063-5. Cited on pages 1 and 10.

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 pages 5, 6, 53, 54, 57, 58, and 73.

Lasota, S., 2009. EXPSPACE lower bounds for the simulation preorder between a communication-free Petri net and a finite-state system. *Information Processing Letters*, 109(15):850–855. doi:10.1016/j.ipl.2009.04.003. Cited on page 7.

Lazić, R., Newcomb, T., Ouaknine, J.O., Roscoe, A.W., and Worrell, J.B., 2008. Nets with tokens which carry data. *Fundamenta Informaticae*, 88(3):251–274. Cited on page 68.

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(3:16). doi:10.1145/2874774. Cited on pages 41 and 50.

Lazić, R. and Totzke, P., 2017. What makes Petri nets harder to verify: Stack or data? In *Concurrency, Security, and Puzzles - Essays Dedicated to Andrew William Roscoe on the Occasion of His 60th Birthday*, volume 10160 of *Lecture Notes in Computer Science*, pages 144–161. Springer. doi:10.1007/978-3-319-51046-0_8. Cited on pages 50, 65, 67, and 68.

Lee, C.S., Jones, N.D., and Ben-Amram, A.M., 2001. The size-change principle for program termination. In *Proceedings of POPL 2001*, pages 81–92. ACM. doi:10.1145/360204.360210. Cited on page 26.

León Sánchez, O. and Ovchinnikov, A., 2016. On bounds for the effective differential Nullstellensatz. *Journal of Algebra*, 449:1–21. doi:10.1016/j.jalgebra.2015.10.009. Cited on page 33.

Lepper, I., 2001. Derivation lengths and order types of Knuth-Bendix orders. *Theoretical Computer Science*, 269(1–2):433–450. doi:10.1016/S0304-3975(01)00015-9. Cited on page 31.

Leroux, J. and Sutre, G., 2004. On flatness for 2-dimensional vector addition systems with states. In *Proceedings of Concur 2004*, volume 3170 of *Lecture Notes in Computer Science*, pages 402–416. Springer. doi:10.1007/978-3-540-28644-8_26. Cited on page 67.

Leroux, J., 2010. The general vector addition system reachability problem by Presburger inductive invariants. *Logical Methods in Computer Science*, 6(3):1–25. doi:10.2168/LMCS-6(3:22)2010. Cited on pages 57, 59, 60, and 64.

Leroux, J., 2011. Vector addition system reachability problem: a short self-contained proof. In *Proceedings of POPL 2011*, pages 307–316. ACM. doi:10.1145/1926385.1926421. Cited on pages 6 and 64.

Leroux, J., 2013. Vector addition system reversible reachability problem. *Logical Methods in Computer Science*, 9(1:5). doi:10.2168/LMCS-9(1:5)2013. Cited on pages 65 and 66.

Leroux, J., Praveen, M., and Sutre, G., 2013. A relational trace logic for vector addition systems with application to context-freeness. In *Proceedings of Concur 2013*, volume 8052 of *Lecture Notes in Computer Science*, pages 137–151. Springer. doi:10.1007/978-3-642-40184-8_11. Cited on page 66.

Leroux, J. and Schnoebelen, Ph., 2014. On functions weakly computable by Petri nets and vector addition systems. In *Proceedings of RP 2014*, volume 8762 of *Lecture Notes in Computer Science*, pages 190–202. Springer. doi:10.1007/978-3-319-11439-2_15. Cited on page 49.

Lincoln, P., Mitchell, J., Scedrov, A., and Shankar, N., 1992. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56(1–3):239–311. doi:10.1016/0168-0072(92)90075-B. Cited on page 7.

Lipton, R.J., 1976. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University. Cited on pages 4, 6, 50, 53, 63, 65, 66, 67, and 69.

Löb, M.H. and Wainer, S.S., 1970. Hierarchies of number theoretic functions, I. *Archiv für Mathematische Logik und Grundlagenforschung*, 13:39–51. doi:10.1007/BF01967649. Cited on pages 5, 32, 36, 37, and 41.

Lomazova, I.A. and Schnoebelen, P., 2000. Some decidability results for nested Petri nets. In *Proceedings of PSI 1999*, volume 1755 of *Lecture Notes in Computer Science*, pages 208–220. Springer. doi:10.1007/3-540-46562-6_18. Cited on page 67.

Mayr, E.W., 1981. An algorithm for the general Petri net reachability problem. In *Proceedings of STOC 1981*, pages 238–246. ACM. doi:10.1145/800076.802477. Cited on pages 5, 6, 53, 54, 56, 57, and 73.

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 page 4.

Mayr, R., 2000. Process rewrite systems. *Information and Computation*, 156(1–2):264–286. doi:10.1006/inco.1999.2826. Cited on page 67.

Mayr, R., 2003. Undecidable problems in unreliable computations. *Theoretical Computer Science*, 297(1–3):337–354. doi:10.1016/S0304-3975(02)00646-1. Cited on page 15.

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 3 and 32.

Meyer, A.R. and Ritchie, D.M., 1967. The complexity of loop programs. In *Proceedings of ACM '67*, pages 465–469. doi:10.1145/800196.806014. Cited on pages 5 and 36.

Meyer, A.R., 1975. Weak monadic second order theory of successor is not elementary-recursive. In *Proceedings of Logic Colloquium 1972–73*, volume 453 of *Lecture Notes in Mathematics*, pages 132–154. Springer. doi:10.1007/BFb0064872. Cited on page 5.

Meyer, R., 2008. On boundedness in depth in the π-calculus. In *Proceedings of IFIP TCS 2008*, volume 273 of *IFIP AICT*, pages 477–489. Springer. doi:10.1007/978-0-387-09680-3_32. Cited on page 32.

Milner, R., 1990. Operational and algebraic semantics of concurrent processes. In *Handbook of Theoretical Computer Science*, volume B, chapter 19, pages 1201–1242. Elsevier. doi:10.1016/B978-0-444-88074-1.50024-X. Cited on page 15.

Moreno Socías, G., 1992. Length of polynomial ascending chains and primitive recursiveness. *Mathematica Scandinavica*, 71(2):181–205. doi:10.2307/24492715. Cited on page 32.

Müller, H., 1985. The reachability problem for VAS. In *Advances in Petri Nets 1984*, volume 188 of *Lecture Notes in Computer Science*, pages 376–391. Springer. doi:10.1007/3-540-15204-0_21. Cited on pages 6, 57, and 63.

Nash-Williams, C.St.J.A., 1963. On well-quasi-ordering finite trees. *Mathematical Proceedings of the Cambridge Philosophical Society*, 59(4):833–835. doi:10.1017/

S0305004100003844. Cited on page 11.

Omri, E. and Weiermann, A., 2009. Classifying the phase transition threshold for Ackermannian functions. *Annals of Pure and Applied Logic*, 158(3):156–162. doi:10.1016/j.apal.2007.02.004. Cited on page 41.

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 2.

Petri, C.A., 1962. *Kommunikation mit Automaten*. Ph.D. Thesis, Universität Bonn. Cited on page 56.

Podelski, A. and Rybalchenko, A., 2004. Transition invariants. In *Proceedings of LICS 2004*, pages 32–41. IEEE. doi:10.1109/LICS.2004.1319598. Cited on pages 11 and 14.

Pouzet, M. and Sobrani, M., 2003. The order type of the collection of finite series-parallel posets. *Discrete Mathematics*, 265(1–3):189–211. doi:10.1016/S0012-365X(02)00580-0. Cited on page 33.

Rackoff, C., 1978. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231. doi:10.1016/0304-3975(78)90036-1. Cited on pages 4, 65, 66, and 71.

Rambow, O., 1994. Multiset-valued linear index grammars: imposing dominance constraints on derivations. In *Proceedings of ACL 1994*, pages 263–270. ACL Press. doi:10.3115/981732.981768. Cited on pages 68 and 73.

Rathjen, M., 2006. The art of ordinal analysis. In *Proceedings of ICM 2006*, volume 2, pages 45–69. European Mathematical Society. Cited on page 50.

Reinhardt, K., 2008. Reachability in Petri nets with inhibitor arcs. In *Proceedings of RP 2008*, volume 223 of *Electronic Notes in Theoretical Computer Science*, pages 239–264. doi:10.1016/j.entcs.2008.12.042. Cited on pages 65, 67, and 74.

Render, E. and Kambites, M., 2009. Rational subsets of polycyclic monoids and valence automata. *Information and Computation*, 207(11):1329–1339. doi:10.1016/j.ic.2009.02.012. Cited on page 53.

Reutenauer, C., 1990. *The mathematics of Petri nets*. Masson and Prentice. Cited on page 57.

Ritchie, R.W., 1965. Classes of recursive functions based on Ackermann's function. *Pacific Journal of Mathematics*, 15(3):1027–1044. doi:10.2140/pjm.1965.15.1027. Cited on page 41.

Robertson, N. and Seymour, P.D., 2010. Graph minors. XXIII. Nash-Williams' immersion conjecture. *Journal of Combinatorial Theory, Series B*, 100(2):181–205. doi:10.1016/j.jctb.2009.07.003. Cited on page 11.

Rosa-Velardo, F., 2017. Ordinal recursive complexity of unordered data nets. *Information and Computation*, 254(1):41–58. doi:10.1016/j.ic.2017.02.002. Cited on pages 3, 4, 32, 41, and 49.

Rosier, L.E. and Yen, H.C., 1986. A multiparameter analysis of the boundedness problem for vector addition systems. *Journal of Computer and System Sciences*, 32(1):105–135. doi:10.1016/0022-0000(86)90006-1. Cited on page 66.

Sacerdote, G.S. and Tenney, R.L., 1977. The decidability of the reachability problem for vector addition systems. In *Proceedings of STOC 1977*, pages 61–76. ACM. doi:10.1145/800105.803396. Cited on pages 5 and 56.

Schmidt, D., 1979. *Well-partial orderings and their maximal order types*. Habilitationsschrift, Heidelberg. Cited on pages 33 and 76.

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 2, 4, 15, 35, 43, and 49.

Schnoebelen, Ph., 2010a. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In *Proceedings of 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 4, 15, 35, 41, 43, and 49.

Schnoebelen, Ph., 2010b. Lossy counter machines decidability cheat sheet. In *Proceedings of 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 15.

Schwichtenberg, H. and Wainer, S.S., 2012. *Proofs and Computation. Perspectives in Logic*. Cambridge University Press. Cited on pages 23, 50, 76, and 77.

Simpson, S.G., 1988. Ordinal numbers and the Hilbert basis theorem. *Journal of Symbolic Logic*, 53(3):961–974. doi:10.2307/2274585. Cited on page 50.

Steila, S., 2016. *Terminating via Ramsey's Theorem*. Ph.D. Thesis, Universitá degli Studi di Torino. Cited on pages 31 and 32.

Turing, A.M., 1949. Checking a large routine. In *Proceedings of EDSAC 1949*, pages 67–69. Cited on pages 11, 12, and 13.

Urquhart, A., 1990. The complexity of decision procedures in relevance logic. In *Truth or Consequences: Essays in honour of Nuel Belnap*, pages 61–76. Kluwer. doi:10.1007/978-94-009-0681-5_5. Cited on page 7.

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 4, 7, 15, 32, 35, 41, 43, and 49.

Van der Meeren, J., 2015. *Connecting the Two Worlds: Well-partial-orders and Ordinal Notation Systems*. Ph.D. Thesis, Universiteit Gent. Cited on pages 33 and 76.

Vardi, M.Y., 2009. From philosophical to industrial logics. In *Proceedings of ICLA 2009*, volume 5378 of *Lecture Notes in Computer Science*, pages 89–115. Springer. doi:10.1007/978-3-540-92701-3_7. Cited on page 1.

Velner, Y., Chatterjee, K., Doyen, L., Henzinger, T.A., Rabinovich, A., and Raskin, J.F., 2015. The complexity of multi-mean-payoff and multi-energy games. *Information and Computation*, 241:177–196. doi:10.1016/j.ic.2015.03.001. Cited on page 7.

Verma, K.N. and Goubault-Larrecq, J., 2005. Karp-Miller trees for a branching extension of VASS. *Discrete Mathematics and Theoretical Computer Science*, 7 (1):217–230. Cited on page 73.

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 36, 37, and 41.

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 23.

Weiermann, A., 1994. Complexity bounds for some finite forms of Kruskal's Theorem. *Journal of Symbolic Computation*, 18(5):463–488. doi:10.1006/jsco.1994.1059. Cited on pages 3, 31, 32, and 33.

Weiermann, A., 1995. Termination proofs for term rewriting systems by lexicographic path orderings imply multiply recursive derivation lengths. *Theoretical Computer Science*, 139(1–2):355–362. doi:10.1016/0304-3975(94)00135-6. Cited on page 31.

Wies, T., Zufferey, D., and Henzinger, T.A., 2010. Forward analysis of depth-bounded processes. In *Proceedings of FoSSaCS 2010*, volume 6014 of *Lecture Notes in Computer Science*, pages 94–108. Springer. doi:10.1007/978-3-642-12032-9_8. Cited on page 32.

Zetzsche, G., 2016. The complexity of downward closure comparisons. In *Proceedings of ICALP 2016*, volume 55 of *Leibniz International Proceedings in Informatics*, article 123, 14 pages. LZI. doi:10.4230/LIPIcs.ICALP.2016.123. Cited on pages 64 and 65.

Zuleger, F., 2017. Ranking functions for vector addition systems. arXiv:1710.10292 [cs.LO]. Cited on page 70.

List of Publications

Invited Papers

[I1] Leroux, J. and Schmitz, S., 2016. Ideal decompositions for vector addition systems. In *Proceedings of STACS 2016*, volume 47 of *Leibniz International Proceedings in Informatics*, article 1, 13 pages. LZI. doi:10.4230/LIPIcs.STACS.2016.1.

[I2] Schmitz, S., 2016. Automata column: The complexity of reachability in vector addition systems. *ACM SIGLOG News*, 3(1):3–21. doi:10.1145/2893582.2893585. Cited on pages 53, 54, and 69.

[I3] Schmitz, S., 2014. Complexity bounds for ordinal-based termination. In *Proceedings of 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 pages 3, 9, 11, 19, 24, 25, and 53.

[I4] Schmitz, S. and Schnoebelen, Ph., 2013. The power of well-structured systems. In *Proceedings of Concur 2013*, volume 8052 of *Lecture Notes in Computer Science*, pages 5–24. Springer. doi:10.1007/978-3-642-40184-8_2. Cited on pages 9, 19, 35, and 43.

Journal Papers

[J1] Schmitz, S., 2016. Implicational relevance logic is 2-ExpTime-complete. *Journal of Symbolic Logic*, 81(2):641–661. doi:10.1017/jsl.2015.7. Cited on pages 2 and 7.

[J2] Chambart, P., Finkel, A., and Schmitz, S., 2016. Forward analysis and model checking for trace bounded WSTS. *Theoretical Computer Science*, 637:1–29. doi:10.1016/j.tcs.2016.04.020.

[J3] Schmitz, S., 2016. Complexity hierarchies beyond Elementary. *ACM Transactions on Computation Theory*, 8(1:3):1–36. doi:10.1145/2858784. Cited on pages 3, 5, 33, 35, 36, 39, 40, 41, 42, 45, and 49.

[J4] Lazić, R. and Schmitz, S., 2015. 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 pages 2, 4, 7, 32, 50, 65, 68, 71, 72, and 73.

[J5] Haase, C., Schmitz, S., and Schnoebelen, Ph., 2014. The power of priority channel systems. *Logical Methods in Computer Science*, 10(4:4):1–39. doi:10.2168/LMCS-10(4:4)2014. Cited on pages 4, 32, 35, 41, 43, and 49.

[J6] Héam, P.C., Nicaud, C., and Schmitz, S., 2010. Parametric random generation of deterministic tree automata. *Theoretical Computer Science*, 411 (38–39):3469–3480. doi:10.1016/j.tcs.2010.05.036.

[J7] Schmitz, S., 2010. An experimental ambiguity detection tool. *Science of Computer Programming*, 75(1–2):71–84. doi:10.1016/j.scico.2009.07.002.

Conference Papers

[C1] Colcombet, Th., Jurdziński, M., Lazić, R., and Schmitz, S., 2017. Perfect half-space games. In *Proceedings of LICS 2017*. IEEE. doi:10.1109/LICS.2017.8005105. Cited on page 7.

[C2] Bérard, B., Haar, S., Schmitz, S., and Schwoon, S., 2017. The complexity of diagnosability and opacity verification for Petri nets. In *Proceedings of Petri Nets 2017*, volume 10258 of *Lecture Notes in Computer Science*, pages 200–220. Springer. doi:10.1007/978-3-319-57861-3_13.

[C3] Baelde, D., Lunel, S., and Schmitz, S., 2016. A sequent calculus for a modal logic on finite data trees. In *Proceedings of CSL 2016*, volume 62 of *Leibniz International Proceedings in Informatics*, article 32, 16 pages. LZI. doi:10.4230/LIPIcs.CSL.2016.32.

[C4] Goubault-Larrecq, J. and Schmitz, S., 2016. Deciding piecewise testable separability for regular tree languages. In *Proceedings of ICALP 2016*, volume 55 of *Leibniz International Proceedings in Informatics*, article 97, 15 pages. LZI. doi:10.4230/LIPIcs.ICALP.2016.97. Cited on page 74.

[C5] Lazić, R. and Schmitz, S., 2016. The complexity of coverability in ν-Petri nets. In *Proceedings of LICS 2016*, pages 467–476. ACM. doi:10.1145/2933575.2933593. Cited on pages 4, 50, 63, 71, and 72.

[C6] Hofman, P., Lasota, S., Lazić, R., Leroux, J., Schmitz, S., and Totzke, P., 2016. Coverability trees for Petri nets with unordered data. In *Proceedings of FoSSaCS 2016*, volume 9634 of *Lecture Notes in Computer Science*, pages 445–461. Springer. doi:10.1007/978-3-662-49630-5_26.

[C7] Jurdziński, M., Lazić, R., and Schmitz, S., 2015. Fixed-dimensional energy games are in pseudo-polynomial time. In *Proceedings of 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 7.

[C8] Leroux, J. and Schmitz, S., 2015. Demystifying reachability in vector addition systems. In *Proceedings of LICS 2015*, pages 56–67. IEEE. doi:10.1109/LICS.2015.16. Cited on pages 6, 53, 54, 60, 61, 62, 64, 72, 73, and 74.

[C9] Courtois, J.B. and Schmitz, S., 2014. Alternating vector addition systems with states. In *Proceedings of MFCS 2014*, volume 8634 of *Lecture Notes in Computer Science*, pages 220–231. Springer. doi:10.1007/978-3-662-44522-8_19. Cited on pages 4, 7, 68, 71, and 72.

[C10] Schmitz, S., 2014. Implicational relevance logic is 2-ExpTime-complete. In *Proceedings of RTA-TLCA 2014*, volume 8560 of *Lecture Notes in Computer Science*, pages 395–409. Springer. doi:10.1007/978-3-319-08918-8_27. Superseded by [J1].

[C11] Lazić, R. and Schmitz, S., 2014. Non-elementary complexities for branching VASS, MELL, and extensions. In *Proceedings of CSL-LICS 2014*, article 61, 10 pages. ACM. doi:10.1145/2603088.2603129. Superseded by [J4].

[C12] Haase, C., Schmitz, S., and Schnoebelen, Ph., 2013. The power of priority channel systems. In *Proceedings of Concur 2013*, volume 8052 of *Lecture Notes in Computer Science*, pages 319–333. Springer. doi:10.1007/978-3-642-40184-8_23. Superseded by [J5].

[C13] Boral, A. and Schmitz, S., 2013. Model checking parse trees. In *Proceedings of LICS 2013*, pages 153–162. IEEE Press. doi:10.1109/LICS.2013.21.

[C14] Karandikar, P. and Schmitz, S., 2013. The parametric ordinal-recursive complexity of Post embedding problems. In *Proceedings of FoSSaCS 2013*, volume 7794 of *Lecture Notes in Computer Science*, pages 273–288. Springer. doi:10.1007/978-3-642-37075-5_18. Cited on pages 4, 49, and 69.

[C15] Bertsch, E., Nederhof, M.J., and Schmitz, S., 2013. On LR parsing with selective delays. In *Proceedings of CC 2013*, volume 7791 of *Lecture Notes in Computer Science*, pages 244–263. Springer. doi:10.1007/978-3-642-37051-9_13.

[C16] Haddad, S., Schmitz, S., and Schnoebelen, Ph., 2012. The ordinal recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In *Proceedings of LICS 2012*, pages 355–364. IEEE Press. doi:10.1109/LICS.2012.46. Cited on pages 4, 35, 41, 43, 49, and 50.

[C17] Blockelet, M. and Schmitz, S., 2011. Model-checking coverability graphs of vector addition systems. In *Proceedings of 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 60 and 66.

[C18] Schmitz, S. and Schnoebelen, Ph., 2011. Multiply-recursive upper bounds with Higman's Lemma. In *Proceedings of ICALP 2011*, volume 6756 of *Lecture Notes in Computer Science*, pages 441–452. Springer. doi:10.1007/978-3-642-22012-8_35. Cited on pages 3, 19, 24, 25, 27, 29, and 32.

[C19] Figueira, D., Figueira, S., Schmitz, S., and Schnoebelen, Ph., 2011. Ackermannian and primitive-recursive bounds with Dickson's Lemma. In *Proceedings of LICS 2011*, pages 269–278. IEEE Press. doi:10.1109/LICS.2011.39. Cited on pages 3, 14, 19, 26, 31, 32, 42, 53, 60, and 62.

[C20] Chambart, P., Finkel, A., and Schmitz, S., 2011. Forward analysis and model checking for trace bounded WSTS. In *Proceedings of Petri Nets 2011*, volume 6709 of *Lecture Notes in Computer Science*, pages 49–68. Springer. doi:10.1007/978-3-642-21834-7_4. Superseded by [J2].

[C21] Schmitz, S., 2010. On the computational complexity of dominance links in grammatical formalisms. In *Proceedings of ACL 2010*, pages 514–524. ACL Press. Cited on pages 4, 68, and 73.

[C22] Héam, P.C., Nicaud, C., and Schmitz, S., 2009. Random generation of deterministic tree (walking) automata. In *Proceedings of CIAA 2009*, volume 5642 of *Lecture Notes in Computer Science*, pages 115–124. Springer. doi:10.1007/978-3-642-02979-0_15. Superseded by [J6].

[C23] Schmitz, S., 2007. Conservative ambiguity detection in context-free grammars. In *Proceedings of ICALP 2007*, volume 4596 of *Lecture Notes in Computer Science*, pages 692–703. Springer. doi:10.1007/978-3-540-73420-8_60.

[C24] Schmitz, S., 2006. Noncanonical LALR(1) parsing. In *Proceedings of DLT 2006*, volume 4036 of *Lecture Notes in Computer Science*, pages 95–107. Springer. doi:10.1007/11779148_10.

[C25] Fortes Gálvez, J., Schmitz, S., and Farré, J., 2006. Shift-resolve parsing: Simple, linear time, unbounded lookahead. In *Proceedings of CIAA 2006*, volume 4094 of *Lecture Notes in Computer Science*, pages 253–264. Springer. doi:10.1007/11812128_24.

Workshop Papers

[W1] Lazić, R. and Schmitz, S., 2015. The ideal view on Rackoff's coverability technique. In *Proceedings of RP 2015*, volume 9328 of *Lecture Notes in Computer Science*, pages 1–13. Springer. doi:10.1007/978-3-319-24537-9_8. Cited on pages 4, 71, and 72.

[W2] Gardent, C., Parmentier, Y., Perrier, G., and Schmitz, S., 2014. Lexical disambiguation in LTAG using left context. In *Proceedings of LTC 2011*, volume 8387 of *Lecture Notes in Computer Science*, pages 67–79. Springer. doi:10.1007/978-3-319-08958-4_6.

[W3] Schmitz, S., 2011. A note on sequential rule-based POS tagging. In *Proceedings of FSMNLP 2011*, pages 83–87. ACL Press. Short paper.

[W4] Schmitz, S. and Le Roux, J., 2008. Feature unification in TAG derivation trees. In *Proceedings of TAG+9*, pages 141–148.

[W5] Schmitz, S., 2008. An experimental ambiguity detection tool. In *Proceedings of LDTA 2007*, volume 203(2) of *Electronic Notes in Theoretical Computer Science*, pages 69–84. Elsevier Science Publishers. doi:10.1016/j.entcs.2008.03.045. Superseded by [J7].

French Conference Papers

[F1] Schmitz, S. and Le Roux, J., 2008. Calculs d'unification sur les arbres de dérivation TAG. In *Proceedings of TALN 2008*, pages 320–329. French version of [W4].

Thesis

[T1] Schmitz, S., 2007. *Approximating Context-Free Grammars for Parsing and Verification.* Ph.D. Thesis, Université de Nice - Sophia Antipolis.

Research Reports

[R1] Schmitz, S., 2006. Modular syntax demands verification. Technical Report I3S/RR-2006-32-FR, Laboratoire I3S, Université de Nice - Sophia Antipolis & CNRS.

Preprints

[P1] Džamonja, M., Schmitz, S., and Schnoebelen, Ph., 2017. On the width of FAC orders, a somewhat rediscovered notion. arXiv:1711.00428 [math.LO].

Lecture Notes

[L1] Schmitz, S. and Schnoebelen, Ph., 2012. Algorithmic aspects of wqo theory. Lecture notes, *ESSLLI 2012*, 108 pages. Cited on pages 4, 9, 19, 30, 35, 43, 76, and 77.

Algorithmic Complexity of Well-Quasi-Orders

Abstract. This document is dedicated to the algorithmic complexity of well-quasi-orders, with a particular focus on their applications in verification, where they allow to tackle systems featuring an infinite state-space, representing for instance integer counters, the number of active threads in concurrent settings, real-time clocks, call stacks, cryptographic nonces, or the contents of communication channels.

The document presents a comprehensive framework for studying such complexities, encompassing the definition of complexity classes suitable for problems with non-elementary complexity and proof techniques for both upper and lower bounds, along with several examples where the framework has been applied successfully. In particular, as a striking illustration of these applications, it includes the proof of the first known complexity upper bound for reachability in vector addition systems and Petri nets.

Key words. Well-quasi-order, verification, infinite-state system, fast-growing complexity, vector addition system, Petri net.

Complexité algorithmique des beaux pré-ordres

Résumé. Ce document est consacré à l'étude de la complexité algorithmique des beaux pré-ordres, et se concentre particulièrement sur leurs applications en vérification, où ils permettent de raisonner sur des systèmes dotés d'une infinité d'états, car modélisant des variables à valeurs entières, un nombre de processus actifs non borné dans un contexte concurrent, des horloges à valeurs réelles dans un contexte temporisé, des piles d'appels de programmes récursifs, des nonces cryptographiques, ou encore des canaux de communication.

Le document présente un cadre théorique pour l'étude de telles complexités, qui inclut la définition de classes de complexité adaptées pour des problèmes de complexité non élémentaire, et des techniques pour établir des bornes supérieures ou inférieures de complexité, ainsi que des exemples pour lesquels ce cadre a pu être appliqué avec succès. En particulier, un exemple frappant de telles applications est celui du problème d'accessibilité dans les systèmes d'addition de vecteurs et les réseaux de Petri, pour lequel ce cadre fournit la première borne supérieure connue.

Mots clefs. Beaux pré-ordres, vérification, système infini, complexité à croissance rapide, système d'addition de vecteurs, réseau de Petri.