Ph.D. Candidacy Proposal
(Third Draft)

Systematic Parameterized Complexity Analysis in Computational Biology and Linguistics

H. Todd Wareham
Department of Computer Science
University of Victoria
Victoria, BC Canada V8W 3P6
e-mail: harold@csr.uvic.ca

January 25, 2016

1 Introduction

Computational Complexity Theory is the area of computer science which determines whether or not computational problems have "good" algorithms, where algorithmic goodness is phrased in terms of the amounts of resources used by an algorithm to solve its associated problem. A complaint against classical theories of computational complexity like NP-completeness is that the the proofs used to show that a problem is unlikely to have a good algorithm give few clues about how to subsequently go about deriving the best possible algorithm for that problem.

The theory of Parameterized Computational Complexity [8, 9] offers a partial solution to this difficulty. Within this theory, any part of a problem's input instance can be separated into a a *parameter*, and various techniques can be invoked to determine if that problem's intractability can be phrased purely as a function of that parameter. Analyses of individual parameters have proven useful in answering questions about problems associated with many areas, e.g., computational biology and robotics [4, 5]. However, of potentially more interest is the analysis of sets of parameters. By looking at each possible subset of a set of parameters, such a systematic parameterized complexity analysis establishes which aspects can conspire to create, and hence are sources of, intractability in a problem.

In my Ph.D. thesis, I want to look at the the logic behind and the mechanics of doing systematic parameterized complexity analyses. My work to date suggests that such analyses are an aid not only in selecting the best possible algorithms for problems but also in refining the definitions of problems to more accurately describe natural processes or phenomena being modeled by those problems. I expect that the major contribution of my thesis research will be to illustrate both of the uses of systematic parameterized complexity analysis suggested above via analyses of two sets of problems associated with, respectively, methods for inferring evolutionary trees in computational biology and theories of phonological processing in linguistics.

This proposal is organized as follows. Section 2 gives an overview of classical and parameterized computational complexity theory, and explains the advantages of systematic parameterized complexity analysis. Sections 3.1 and 3.2 describe my proposed research in computational biology and linguistics, respectively. Finally, Section 4 contains a list of documents that may be of use in evaluating this proposal.

2 Systematic Parameterized Complexity Analysis

2.1 Computational Complexity Theory

Computational complexity theory establishes upper and lower bounds on how efficiently problems can be solved by algorithms, where "efficiency" is judged in terms of the computational resources, e.g., time or space, required by an algorithm to solve its associated problem. As noted by Rounds in his 1991 review, "The presuppositions of an already established theory, such as complexity theory, are perhaps the properties of the theory most easily ignored in making an application" [32, page 10]. With this in mind, the basics of computational complexity theory are reviewed in this section.

Perhaps the most important presuppositions alluded to by Rounds are implicit in the definitions of algorithm complexity and efficiency. The *complexity of an algorithm* is a function that summarizes, for each possible input size, the resource requirements of that algorithm over all inputs of that size. This summary can take many forms, e.g., best-case, average-case, worst-case. This paper is concerned with worst-case measures – that is, if $R(i)$ is the resource required by algorithm A to solve input i and I^n is the set of all inputs of size n, then the worst-case complexity of A for value n is $\max_{i \in I^n} R(i)$. This complexity is further "smoothed" by considering its behavior as n goes to infinity. This is typically stated in $\mathcal{O}$ ("big-Oh") notation, which gives the lowest function that is an asymptotic upper bound on the worst-case complexity of the algorithm, e.g., $3n^2 + 10n - 5 = \mathcal{O}(n^2)$, $\log_3 n/2 = \mathcal{O}(\log_2 n)$. An algorithm is *efficient* if its complexity satisfies some criterion of efficiency, e.g., the complexity function is a polynomial of the input size, and a problem is *tractable* if it has an efficient algorithm.

Computational complexity theory establishes not only what problems can be solved efficiently but also what problems *(probably) cannot* be solved efficiently. This is done by appropriately defining a class $\mathcal{F}$ of tractable problems, a class $\mathcal{C}$ such that it is either known or strongly conjectured that $\mathcal{F} \subset \mathcal{C}$, and a reducibility α between pairs of problems that preserves tractability, i.e., if $X \alpha Y$ and $Y \in \mathcal{F}$ then $X \in \mathcal{F}$. As a reducibility establishes the computational difficulty of problems relative to each other, e.g., if $X \alpha Y$ then Y is at least as computationally difficult as X, it can be used to isolate the hardest problems in a class $\mathcal{C}$ via the notions of hardness and completeness.[1] These notions are significant because if a given problem X is at least as hard as the hardest problem in $\mathcal{C}$, then X does not have an efficient algorithm modulo the strength of the assumption that $\mathcal{F} \subset \mathcal{C}$ (see Figure 1).

Ideally, a complexity-theoretic analysis of a problem is not just a one-sided quest for either algorithms or hardness results. Rather, it is an ongoing dialogue in which both types of results are used to fully characterize the problem by showing which restrictions make that problem tractable and which don't [15, Section 4.1].

[1]Recall that a problem X is *$\mathcal{C}$-hard* if for all problems $Y \in \mathcal{C}$, $Y \alpha X$; if X is also in $\mathcal{C}$, then X is *$\mathcal{C}$-complete.*

Figure 1: The Utility of Hardness/Completeness Results in Computational Complexity Theory. See main text for explanation of symbols.

2.2 Parameterized Computational Complexity Theory

The theory of NP-completeness was inspired by the need to show that certain problems cannot have polynomial-time algorithms [15]; parameterized computational complexity theory was inspired by the following similar need. Most computational problems have input that consists of one or more items; for example, two items from the input to an object-recognition problem might be a grid of observed light/dark values and a set of patterns corresponding to 2-D projections of known objects. Call each such item in the input an *parameter*. When it can be proved that a problem X cannot have a polynomial algorithm, e.g., X is NP-hard, that problem will exhibit one of two algorithmic behaviors relative to any selected parameter k:

1. An algorithm can exist for X whose running time is super-polynomial in k but polynomial in all other parameters, e.g., $k^{k^2}n^2m$; or
2. All algorithms for X have running times that are either super-polynomial in both k and at least one other parameter, e.g., n^km^2, or polynomial in k but super-polynomial in at least one other parameter, e.g., 2^nm^2k.

The former kind of algorithm is often preferable if k has small values in typical instances of a problem, e.g., when $k = 10$ and $n = 1000$, $2^kn^3 = 10^{12} << 10^{30} = n^k$. It is not obvious from looking at the problems themselves which ones have such algorithms. Moreover, classical theories of computational complexity are insensitive to this algorithmic distinction – they can say only that a problem does not have polynomial time algorithms, and are silent on whether this super-polynomial behavior can be isolated relative to particular parameters.[2]

Parameterized computational complexity theory [8, 9] frames this issue via the following definitions:

Definition 1 *A* parameterized problem *is a set $L \subseteq \Sigma^* \times \Sigma^*$, where Σ is a fixed alphabet. For a parameterized problem L and $y \in \Sigma^*$, the* fixed-parameter problem L_y *is defined as the set $\{x|(x,y) \in L\}$.*

The y-component of elements of L is the parameter. It is important to note that y may consist of one or more parameters.

Definition 2 *A parameterized problem L is* fixed-parameter tractable *if there exists a constant α and an algorithm A to determine if (x,y) is in L in time $f(|y|) \cdot |x|^\alpha$, where $f : N \mapsto N$ is an arbitrary function.*

Parameterized computational complexity theory encompasses both a set of techniques for deriving fixed-parameter tractable algorithms and an appropriate set of classes for proving fixed-parameter intractability. Within this theory, class $\mathcal{F}$ defined in Section 2.1 corresponds to the class FPT of fixed-parameter tractable problems, and class $\mathcal{C}$ defined in Section 2.1 is one of the classes of the W hierarchy, $\{W[1], W[2], \ldots, W[P], \ldots, SP\}$, many of whose members are defined by successively more powerful solution-checking circuits (see [8, 9] for details). These classes are related as follows:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P] \subseteq \cdots \subseteq SP$$

It is conjectured that all inclusions in this hierarchy are proper. Hence, no $W[t]$-complete problem is fixed-parameter tractable unless all problems in $W[t]$ are fixed-parameter tractable. Essentially, a W-hardness result for the version of a problem X with parameter k suggests that the super-polynomial resource requirements of X are not just a function of k. Parameterized versions of over one hundred problems from areas as diverse as VLSI design, computational biology, and robotics have been classified within the W hierarchy [18, 27].

[2]This is not strictly true; the effects of a particular parameter on a problem's complexity can in principle be isolated within classical theories of computational complexity by considering the version of that problem in which that parameter is set to an arbitrary constant c. If there is a polynomial-time algorithm for that version, then the problem's super-polynomial behavior can be isolated to that parameter – else, if that version can be shown NP-hard, it cannot. Indeed, this is the strategy advocated in Garey and Johnson's discussion on how to use algorithms and hardness results to map a problem's "frontier of tractability" [15, Sections 4.1 and 4.3]. However, several difficulties severely limit the utility of this approach in practice:

- This approach may miss certain input parameters that contribute to super-polynomial behavior, e.g., a $O(n^km)$ time algorithm in which only k is fixed to a constant is polynomial-time though both k *and* n contribute to super-polynomial behavior; and
- It is not obvious that that one can prove *either* NP-hardness *or* give a polynomial-time algorithm for *every* possible parameter when that parameter has been fixed to a constant.

The theory of parameterized complexity described in this section gets around these problems by defining explicit mechanisms for isolating a parameter's contribution to algorithmic complexity, e.g., a parameterized problem's two-component input instance and the function f in the the definition of fixed-parameter tractability, and providing a richer series of intractable classes, i.e., the W hierarchy.

LONGEST COMMON SUBSEQUENCE

Instance: A set of k strings X_1, ..., X_k over an alphabet Σ, and a positive integer m.

Question: Is there a string $X \in \Sigma^*$ of length at least m that is a subsequence of X_i for $i = 1, ..., k$?

(a)

Parameter	Alphabet Size $\|\Sigma\|$		
	Unbounded	Parameter	Constant
–	NP-hard	NP-hard	NP-hard
k	W[t]-hard for $t \geq 1$	W[t]-hard for $t \geq 1$	?
m	W[2]-hard	FPT	FPT
k, m	W[1]-complete	FPT	FPT

(b)

Figure 2: A Systematic Parameterized Complexity Analysis of the LONGEST COMMON SUBSEQUENCE problem (reprinted from [4]). (a) The formal definition of the problem; (b) the systematic parameterized complexity analysis of the problem relative to $P = \{|\Sigma|, k, m\}$.

2.3 Systematic Parameterized Complexity Analysis: What It Is and Why It Is Useful

There may be many ways to analyze an intractable problem relative to a set of parameters. I will focus on the following one of these ways.

Definition 3 *Given a computational problem X and some subset $P = \{p_1, \ldots, p_n\}$ of the parameters associated with X, a* systematic parameterized complexity analysis of X relative to P *determines the parameterized complexity of X relative to all $2^n - 1$ non-empty subsets of P.*

Several such analyses have been done to date for various problems [4, 5, 12, 17, 43] (see Figure 2). A systematic parameterized complexity analysis is a lot of work, considering the effort that almost always goes into deriving fixed-parameter tractable algorithms and proving W-hardness. To see why such an analysis is worth doing, let us examine what an intractability result in a classical theory of computational complexity like NP-completeness really means.

Recall that any computational problem has an infinite number of associated algorithms. If a polynomial-time algorithm exists for that problem, we usually forget about all of the others.[3]

[3]A famous counterexample to this statement involves the simplex algorithm for the linear programming (LP) problem. Though this algorithm requires exponential time in the worst case, it almost always outperforms the polynomial-time ellipsoid algorithm on LP instances encountered in practice. My interpretation of this phenomenon is that there are restrictions in typical LP instances that are exploited by the simplex algorithm but are not recognized in the general formulation of LP problems. Such problems highlight the need for a type of parameterized analysis that find all practical super-polynomial time algorithms for problems that are known only to have impractical high-order polynomial-time algorithms. It does not seem to me that parameterized computational complexity theory as defined to date can underly this type of analysis. A theory that could would be a good topic for future research.

However, suppose that the problem is NP-hard and thus probably does not have a polynomial-time algorithm. The question is, what types of super-polynomial time algorithms exist for X? It is important to know the full spectrum of such algorithms because not all manners of super-polynomial behavior are equally bad under all situations [15, Section 4.1].

A first approximation to determining what types of super-polynomial time algorithms exist for such a problem is to make up a list of parameters that seem to be important for algorithms solving that problem or whose values are restricted in practice, and then to determine how each parameter in this list contributes to that problem‘s computational complexity via a parameterized analysis. Assuming that the classes of the W hierarchy are distinct and thus differ (at least qualitatively) in the amount of power that each encodes in the selected parameter, the higher a W-hardness result for the version of a problem with parameter k, the less of that problem's general super-polynomial behavior can be attributed to k [10, 43]. However, a consequence of the definitions in Section 2.2 is that a W-hardness result relative to a single parameter k of a problem X says *nothing* about which supersets of parameters that include k may be responsible for super-polynomial behavior. Thus, in order to fully "map" all sources of such behavior relative to the selected list of parameters, it is necessary to do a systematic parameterized complexity analysis.

The "intractability map" produced by a systematic parameterized complexity analysis over a list P of n parameters of a problem X is just a list of $2^n - 1$ fixed-parameter tractable algorithms and W-hardness results which shows the manners (if not all the forms) in which the intractability of X can manifest itself as algorithms whose running times are super-polynomial in the parameters of P. Note that such a map does not summarize *all* possible super-polynomial behaviors for X; however, my own experience in doing parameterized analyses of phonological processing systems [10, 42, 43] suggests that a systematic parameterized complexity analysis on some initial set of parameters will lead to the discovery and subsequent analysis of new and hopefully more relevant parameters. Thus a systematic parameterized complexity analysis should be seen not as a technique that generates all wanted answers at one go, but rather as part of an ongoing discovery procedure for analyzing problems, in which classical and parameterized computational complexity are used in an alternate and complementary fashion to first show that problems are intractable and then diagnose the sources of this intractability.

How does one use the results of systematic parameterized complexity analyses? I see at least two uses for the intractability maps produced by these analyses (see Figure 3):

- In cases where one is given an intractable problem and can derive typical ranges of parameter values from actual instances of the problem, e.g., DNA sequence reconstruction from sequence fragments [17] or inferring evolutionary trees for biological species (see Section 3.1), one can use an intractability map in conjunction with the knowledge of typical values of the parameters to establish what classes of problem instances are tractable in practice. This aids in finding the best possible algorithms for that problem.
- In cases where one is given an intractable problem that is used to model natural process or phenomenon that is known to be tractable, e.g., visual [38] or natural language (see Section 3.2) processing in the brain, one can use an intractability map to establish what aspects of that problem are responsible for intractability. This aids in refining the definition of the problem to (hopefully) better approximate the actual mechanisms in the phenomenon being

Parameter	p_3	
	Unbounded	Parameter
–	–	$W[1]$-complete
p_1	FPT A1: $O(2^{p_1})$	FPT A2: $O({p_1}^{p_3})$
p_2	W[2]-hard	W[2]-complete
p_1, p_2	FPT A3: $O({p_1}^{p_2})$	FPT A4: $O({p_3}^{p_2\sqrt{p_1}})$

(a)

p_1	p_2	p_3	
		small	large
small	small	A1, A2, A3, A4	A3, A4
small	large	A1, A2	–
large	small	A2, A3	A3
large	large	A2	–

(b)

Figure 3: Systematic Parameterized Complexity Analysis and Intractability Maps. (a) The intractability map resulting from the systematic parameterized complexity analysis of a hypothetical problem X relative to $P = \{p_1, p_2, p_3\}$. Note that relative to P, the sources of intractability in X are $\{p_1\}$, $\{p_1, p_3\}$, $\{p_1, p_2\}$, and $\{p_1, p_2, p_3\}$ with associated fixed-parameter tractable algorithms A1, A2, A3, and A4, respectively. The running time of each algorithm in terms of its parameters is given in O-notation. (b) The list of practical fixed-parameter tractable algorithms relative to the parameters in P when the typical values for each parameter are either small or large. Note that no given fixed-parameter tractable algorithm is practical when both p_2 and p_3 have large values; this is equivalent to saying that under this intractability map, only those problem instances in which the values of both p_2 and p_3 are large are intractable.

modeled.

It is my hope that the Ph.D. research proposed in the following section will provide useful illustrations of both of these applications of systematic parameterized complexity analysis.

3 Summary of Proposed Research

This section briefly describes the two groups of computational problems which I plan to examine in my Ph.D. thesis. Each sub-section gives a brief introduction to the computational problems, followed by a summary of previous research done on these problems and my proposed research.

Figure 4: Some Sample Phylogenies. Here are the three possible phylogenies relating the Human (H), Chimpanzee (C), and Gorilla (G) species.

3.1 Parameterized Complexity Analysis in Phylogeny Inference

3.1.1 Problems

Given a set of biological species, a *phylogeny* is a tree whose leaves are labeled with the species in the given set and that shows the speciation events by which the species in the given set were derived from their most recent common ancestor (see Figure 4). Phylogenies are used in many areas of evolutionary biology [14].

As the true phylogeny of a group of species is seldom known, an algorithmic method must often be invoked to reconstruct a phylogeny (or a set of equally-preferable phylogenies) from the available data. There are many methods for inferring phylogenies [19, 28, 36]. Methods reconstruct phylogenies by either ranking all possible phylogenies for the given data relative to an explicitly stated criterion and selecting the phylogeny (or phylogenies) whose value is optimal under this criterion (the *criterion-based* methods, e.g., maximum parsimony and distance-matrix fitting) or selecting the phylogeny output by a particular algorithm operating on the given data (the *algorithm-based* methods, e.g., hierarchical clustering and neighbor-joining) [36, pages 408–409]. Though the latter are often used in practice because they have efficient low-order polynomial-time algorithms, the former are preferred because they rank all possible phylogenies and thus make the selected phylogenies easier to compare against the phylogenies that are not selected.

Once the correspondence between mathematical and biological concepts is established, many of the criterion-based methods for inferring phylogeny can be formalized as combinatorial optimization problems. Indeed, one of the most popular of these methods, phylogenetic parsimony, is in its most basic form the following problem from communication network design. Let H_2^d denote the d-dimensional binary hypercube graph, i.e., $H_2^d = (V, E)$ where $V = \{0,1\}^d$, the set of all strings of length d on alphabet $\{0,1\}$, and $E = \{\{u,v\} |$ the strings for u and v differ in exactly one position$\}$. Given a tree T in H_2^d, let the *length* of T be the number of edges in T.

Steiner Tree in the Binary Hypercube
Instance: A positive integer d, a subset S, $|S| = n$, of $\{0,1\}^d$, and a positive integer B.
Question: Is there a tree in H_2^d that includes S and whose length is less than or equal to B?

Note that the input to such problems is a $n \times d$ matrix, where matrix position (i, j) is interpreted as giving the value of character i for species j. This form of data is called an *object-by-character* *matrix*. There are also methods that take as input a *distance matrix*, that is, an $n \times n$ matrix in which matrix position (i, j) is interpreted as a measure of the evolutionary distance between species i and j. If one notes that any tree T on n leaves can be converted to a unique $n \times n$ matrix M_T in which matrix position (i, j) gives the sum of the lengths of the edges on the unique path between leaves i and j in the tree, one gets the following type of phylogeny-inference problem.

Closest Tree under Matrix-comparison Statistic S
Instance An $n \times n$ numerical matrix D and a positive integer B.
Question: Is there an edge-weighted tree T on n leaves such that $S(M_T, D) \leq B$?

There are many variants of these two types of problems on these two types of data. However, they all optimize the fit of the given data matrix to a tree subject to some criterion.

The major criterion-based methods and their associated computational problems are described in [19, 20, 28, 36, 41]. Though these methods are preferred on methodological grounds, their use in practice is often restricted by the exponential time requirements of the best current implementations (which invariably have to evaluate a given data-set on n species against all possible $\Omega(2^n)$ phylogenies). The main questions are: (1) What mechanisms are the sources of the computational difficulties in the various methods for inferring phylogenies?; and (2) Can such knowledge be exploited in conjunction with the characteristics of typically-encountered phylogenetic data matrices to derive efficient algorithms for phylogeny inference?

3.1.2 Previous Research

Since the early 1960's, a number of heuristics and exact-solution algorithms have been proposed and implemented for phylogeny inference [36]. In the mid 1980's, many of the computational problems associated with the phylogenetic parsimony, character compatibility, and distance-matrix fitting criteria were shown to be NP-complete by Day and his colleagues (see [41] and references). Since then, variants of several distance-matrix fitting and character compatibility problems have also been analyzed in terms of NP-hardness, algorithms for special cases, and polynomial-time (in)approximability (see [1, 13, 21] and references).

My M.Sc. research [41] focused on the computational complexity of phylogenetic parsimony, character compatibility, and distance matrix fitting methods for inferring phylogenies. The derived results that are relevant here were correct versions of the flawed reductions in [7], proofs of the PTAS non-approximability of almost all phylogeny inference problems examined to that time via slightly modified versions of their NP-hardness reductions, and a sketch of a polynomial-time factor-2 approximation algorithm applicable under several of the simplest versions of the phylogenetic parsimony criterion.

3.1.3 Proposed Research

In my Ph.D. thesis, I propose to do systematic parameterized complexity analyses of the phylogeny-inference methods examined in my M.Sc. thesis (note that I already done part of such an analysis for the character compatibility methods [40]). I also plan to extend the scope of my previous work to include the variants of distance-matrix fitting methods proposed in [1] and the "perfect phylogeny" variants of the character compatibility methods (see [21] and references), as well as maximum likelihood methods. The former two should be included for the sake of completeness, and the latter should be included because maximum likelihood is becoming a leading method of phylogeny inference for molecular data [36, page 407].

The initial list of parameters I intend to look at for each method includes:

- The number of species;
- The number of characters and number of states per character (where applicable); and
- The bound on the cost of the wanted phylogeny under the particular inference criterion.

Additional parameters will be drawn from characterizations of the output trees, e.g., maximum degree of any non-leaf vertex or maximum weight associated with an edge, and the data matrices.

Parameterized complexity analyses of these problems are relevant because it is often extremely important to derive phylogenies that are optimal under the selected criterion. The exponential time and space requirements of exact-solution methods forces investigators interested in inferring phylogenies for more than 20 species to use heuristic methods. These methods are fast, but there are no known bounds on how close the phylogenies they produce are to optimal. This is important because non-optimal phylogenies typically have different structures than optimal phylogenies, and hence different implications for hypotheses of evolutionary change. There are many examples in the biological literature of hypotheses that have been modified or retracted in light of different estimates of the optimal tree, e.g., the "Out of Africa" hypothesis based on human mitochondrial DNA phylogenies [26, 35]. Hence, in light of the NP-hardness of many phylogeny inference problems, fixed-parameter tractable algorithms which can feasibly produce optimal phylogenies for certain "small" but typical inputs would be preferable to polynomial-time algorithms that produce phylogenies of unknown quality for all inputs.

3.2 Parameterized Complexity Analysis in Computational Phonology

3.2.1 Problems

Phonology is the area of linguistics which studies regularities in the mapping between deep (mental/lexical) and surface (spoken/phonetic) representations in natural language [22]. These regularities are realized as rules or constraints on phonological representations. As such, the phonological component of natural language is concerned with relations between mental and spoken representations of speech, and the computational component of phonological theories is concerned with the mechanisms implementing these relations.

Many phonological theories have been developed over the last thirty years. Broadly speaking, these theories differ in three aspects:

1. **Rule-Based vs. Constraint-Based Mechanism**: The transformation from deep to surface structure can be implemented by either a set of rewriting rules which operate on a deep

structure to create the wanted surface structure or as a set of constraints which select the surface structure from a set of candidate surface structures that are derived from the deep structure.

2. **Sequential vs. Parallel Application**: The rules or constraints may be applied one at a time in a sequential manner in some (not necessarily total) order or all at once in parallel. When the latter occurs, there are further choices dealing with how one handles rules or constraints that conflict, i.e., match on the same part of the candidate representation but cause incompatible results.

3. **Context-Sensitive vs. Finite-State Implementation**: The rules or constraints can vary in permitted expressive power such that they encode context-sensitive or finite-state transducers or automata.[4]

It is interesting that theories have developed historically such that they adopt *either* the first *or* the second option over almost all of these aspects. The earliest of the modern phonological theories was proposed in Chomsky and Halle's *The Sound Pattern of English* (commonly abbreviated as *SPE*) in 1968 [6], and was based on the sequential application of context-sensitive rules. Such systems have the following basic computational problem.

Grammar Derivation
Instance: An alphabet Σ, a grammar G, a partial order O on the rules of G, and strings $d, s \in \Sigma^*$.
Question: Can the rules of G be applied to d in a manner consistent with O to produce s?

Subsequent theories have been based on the parallel evaluation of finite-state rules [23, 25] or constraints [3, 11, 29, 33]. These systems have the following basic computational problem.

Finite-State Intersection
Instance: An alphabet Σ and a set A of k finite-state automata over Σ^*.
Question: Is there a string in Σ^* that is accepted by every finite-state automaton in A?

The switch to theories based on finite-state constraints and rules that are applied in parallel was motivated in part by the computational intractability inherent in systems of sequentially-applied context-sensitive rules [16, 24]. However, researchers have found that even though the new finite-state mechanisms are simpler, their interactions in parallel can be a source of unforeseen computational difficulties. The main questions are: (1) What mechanisms are the sources of the computational difficulties in the various phonological theories?; and (2) Can these mechanisms be restricted to yield practical and psychologically realistic theories of phonological processing?

3.2.2 Previous Research

A number of systems have been developed that implement heuristics for coping with the phonological components of natural language processing systems [34]; however, the mechanisms in such systems are seldom based on phonological theories within linguistics. When systems have been

[4] I am not aware of any phonological theory that implements its rules or constraints with context-free rules or automata. This is a marked and curious contrast with modern work on syntax, which has focused almost exclusively on context-free rule systems.

built based on extant theory, they have typically included restrictions to ensure efficient operation, e.g., Optimality Phonology [37]. The necessity of these restrictions has been shown theoretically for several of the older phonological theories, namely *SPE* Phonology [6] and the KIMMO system [23], via proofs that basic computational problems associated with these theories are NP-hard [2, 31].

Over the last several years, I have done parameterized analyses of problems associated with *SPE* Phonology and the KIMMO system [10, 42]. Though these frameworks are outdated, the effort has been vindicated by my recent realization that analyses of KIMMO also apply to two current theories, Optimality Phonology and Declarative Phonology (and indeed give the first classical, i.e., NP-completeness, results for problems associated with these theories) [43]. Using the parameterized framework, I have been able to counter the intuitions of several authors about the efficient implementation of both KIMMO and Optimality Phonology [42, 43].

3.2.3 Proposed Research

In my Ph.D. thesis, I propose to do more detailed parameterized analyses of the phonological theories that I have examined in [10, 42, 43], and extend the scope of my previous work to include the Theory of Constraints and Repair Strategies [24]. Where possible, I also plan to comment on the motivation behind computationally-costly theoretical mechanisms and speculate on the nature of revised theories that restrict or eliminate such mechanisms, *sensu* [30, 31].

The initial list of parameters I intend to look at for each theory includes:

- The size of the alphabet;
- The size of the input strings;
- The number of rules or constraints; and
- The maximum allowable size of any rule or constraint, e.g., number of transitions/ states.

Further parameters will be drawn from the allowed degree of conflict and parallelism in rule/constraint application and characterizations of the expressive power of the rules/constraints. The latter will probably involve formalizations of constraints other than finite-state automata, e.g., first-order logic.

Parameterized complexity analyses of these problems are relevant in light of the odd and incomplete tapestry of tractability and intractability results associated with various phonological theories. In particular, when I compare the tractability of systems based on extremely simple finite-state rules which are not allowed to conflict [25] with the intractability results mentioned above, it seems to me that there are interesting interactions between both the degree of permitted conflict and parallelism and the expressive power of the rules/constraints which need to be untangled via a complexity-theoretic investigation. Such an investigation has obvious implications for practical implementations of these phonological theories. It is also part of the process (described more fully in [30, 38, 39, 43]) of iteratively refining such theories to accurately reflect the actual mechanisms used in language processing in the human brain.

4 List of Available/Attached Documents

The following documents may be useful in evaluating the research proposed above. Several of these have already been given to members of my committee. Starred documents have been provided with this proposal. All others are available by request.

- H. Todd Wareham. 1992. W[1]-completeness/hardness proofs for the Character Compatibility problem. E-mail messages to M. Fellows. (*)
- H. Todd Wareham. 1993a. On the Computational Complexity of Inferring Evolutionary Trees. M.Sc. Thesis. Technical Report 9301, Department of Computer Science, Memorial University of Newfoundland.
- H. Todd Wareham. 1993b. W[t]-hardness proofs for problems in the theory of *SPE* Phonology. Manuscript. (*)
- H. Todd Wareham. 1995. Parameterized Complexity Analysis in Computational Phonology. Submitted to the Annual Meeting of the Association for Computational Linguistics (*ACL-96*); not accepted. (*)
  - Referee comments on above paper. (*)
- H. Todd Wareham. 1996. The Role of Parameterized Computational Complexity Theory in Cognitive Modeling. AAAI-96 Workshop Working Notes: *Computational Cognitive Modeling: Source of the Power.* (*)

References

[1] Richa Agarwala, Vineet Bafna, Martin Farach, Babu Narayanan, Michael Paterson, and Mikkel Thorup. 1995. On the Approximability of Numerical Taxonomy: Fitting Distances by Tree Metrics. In *SODA'96*.

[2] G. Edward Barton, Robert C. Berwick, and Eric S. Ristad. 1987. *Computational Complexity and Natural Language.* MIT Press, Cambridge, MA.

[3] Steven Bird and T. Mark Ellison. 1994. One-Level Phonology: Autosegmental Representations and Rules as Finite Automata. *Computational Linguistics*, 20(1), 55–90.

[4] Hans L. Bodlaender, Rod G. Downey, Michael R. Fellows, Michael T. Hallett, and H. Todd Wareham. 1995. Parameterized Complexity Analysis in Computational Biology. *Computer Applications in the Biosciences*, 11(1), 49-57.

[5] Marco Cesati and H. Todd Wareham. 1995. Parameterized Complexity Analysis in Robot Motion Planning. In *Proceedings of the 25th IEEE International Conference on Systems, Man, and Cybernetics: Volume 1*, pages 880–885. IEEE Press, Los Alamitos, CA.

[6] Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English.* Harper and Row, New York.

[7] William H. E. Day. 1983. Computationally Difficult Parsimony Problems in Phylogenetic Systematics. *Journal of Theoretical Biology*, 103, 429–438.

[8] Rod G. Downey and Michael R. Fellows. 1995a. Fixed-parameter tractability and completeness I. Basic results. *SIAM Journal on Computing*, 24(4), 873–921.

[9] Rod G. Downey and Michael R. Fellows. 1995b. Fixed-parameter tractability and completeness II. On completeness for $W[1]$. *Theoretical Computer Science*, 141, 109–131.

[10] Rod G. Downey, Michael R. Fellows, Bruce M. Kapron, Michael T. Hallett, and H. Todd Wareham. 1994. Parameterized Complexity of Some Problems in Logic and Linguistics (Extended Abstract). In A. Nerode and Yuri V. Matiyasevich (eds.) *Logical Foundations of Computer Science*, pages 89–101. Lecture Notes in Computer Science no. 813. Springer-Verlag, Berlin.

[11] T. Mark Ellison 1994. Phonological Derivation in Optimality Theory. In *COLING'94*, pages 1007–1013.

[12] Patricia A. Evans. 1995. Adding Arcs to the Longest Common Subsequence Problem. Manuscript.

[13] Martin Farach, Sampath Kannan, and Tandy Warnow. 1995. A Robust Model for Finding Optimal Evolutionary Trees. *Algorithmica*, 13, 155–179.

[14] Vicki A. Funk and Daniel R. Brooks. 1990. *Phylogenetic Systematics as the Basis of Comparative Biology.* Smithsonian Institution Press, Washington, DC.

[15] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco.

[16] John Goldsmith (ed.). 1993. *The Last Phonological Rule: Reflections on Constraints and Derivations.* The University of Chicago Press.

[17] Michael T. Hallett. 1996. An Integrated Complexity Analysis of Problems from Computational Biology. Ph.D. thesis, Department of Computer Science, University of Victoria.

[18] Michael T. Hallett and H. Todd Wareham. 1994. A Compendium of Parameterized Complexity Results. *SIGACT News*, 25(3), 122-123.

[19] David M. Hillis, Mark W. Allard, and Michael M. Miyamoto. 1993. Analysis of DNA Sequence Data: Phylogenetic Inference. *Methods in Enzymology*, 224, 456–487.

[20] Frank K. Hwang, Dana S. Richards, and Pawel Winter. 1992. *The Steiner Tree Problem.* Annals of Discrete Mathematics no. 53. North-Holland, Amsterdam.

[21] Sampath E. Kannan and Tandy J. Warnow. 1994. Inferring Evolutionary History from DNA Sequences. *SIAM Journal on Computing*, 23(4), 713–737.

[22] Michael J. Kenstowicz. 1994. *Phonology in Generative Grammar.* Blackwell, Cambridge, MA.

[23] Kimmo Koskenniemi. 1983. Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production. Ph.D. thesis, University of Helsinki.

[24] Darlene Lacharité and Carole Paradis. 1993. The Emergence of Constraints in Generative Phonology and a Comparison of Three Current Constraint-Based Models. *Canadian Journal of Linguistics*, 38(2), 127–163.

[25] Eric Laporte 1996. Rational Transductions for Phonetic Conversion and Phonology. Manuscript.

[26] David R. Maddison, Maryann Ruvolo, and David L. Swofford. 1992. Geographic Origins of Human Mitochondrial DNA: Phylogenetic Evidence from Control Region Sequences. *Systematic Biology*, 41(1), 111–124.

[27] Parameterized Complexity Home Page. 1995. URL: `http://www-csc.uvic.ca/home/harold/W_hier/W_hier.html`.

[28] David Penny, Michael D. Hendy, and Michael A. Steel. 1992. Progress with Methods for Constructing Evolutionary Trees. *Trends in Ecology and Evolution*, 7(3), 73–79.

[29] Alan Prince and Paul Smolensky. 1993. Optimality Theory: Constraint Interaction in Generative Grammar. Technical Report RuCCS TR-2, Rutgers University Center for Cognitive Science.

[30] Eric S. Ristad. 1993a. *The Language Complexity Game.* MIT Press, Cambridge, MA.

[31] Eric S. Ristad. 1993b. Complexity of the Simplified Segmental Phonology. Technical report CS-TR-388-92 (revised May 1993), Department of Computer Science, Princeton University.

[32] William Rounds. 1991. The Relevance of Computational Complexity Theory to Natural Language Processing. In Peter Sells, Stuart Shieber, and Thomas Wasow (eds.) *Foundational Issues in Natural Language Processing*, pages 9–29. MIT Press, Cambridge, MA.

[33] James M. Scobbie. 1992. Towards Declarative Phonology. In Steven Bird (ed.) *Declarative Perspectives in Phonology*, pages 1–27. Edinburgh Working Papers in Cognitive Science, Volume No. 7. University of Edinburgh.

[34] Richard Sproat. 1992. *Morphology and Computation.* MIT Press, Cambrdige, MA.

[35] M. Stoneking, S.T. Sherry, and L. Vigilant. 1992. Geographic Origin of Human Mitochondrial DNA Revisited. *Systematic Biology*, 41(3), 384–391.

[36] David L. Swofford, Gary J. Olsen, Peter J. Waddell, and David M. Hillis. 1996. Phylogeny Reconstruction. In David M. Hillis, Craig Moritz, and Barbara K. Mable (eds.) *Molecular Systematics* (Second Edition), pages 407–514. Sinauer Associates, Sunderland, MA.

[37] Bruce B. Tesar. 1995. Computational Optimality Theory. Ph.D. thesis, Department of Computer Science, University of Colorado.

[38] John K. Tsotsos. 1990. Analyzing vision at the complexity level. *Behavioral and Brain Science*, 13, 423–469.

[39] John K. Tsotsos. 1993. The Role of Computational Complexity in Perceptual Theory. In Sergio C. Masin (ed.) *Foundations of Perceptual Theory*, pages 261–296. North-Holland, Amsterdam.

[40] H. Todd Wareham. 1992. W[1]-completeness/hardness proofs for the Character Compatibility problem. E-mail messages to M. Fellows.

[41] H. Todd Wareham. 1993. On the Computational Complexity of Inferring Evolutionary Trees. M.Sc. Thesis. Technical Report 9301, Department of Computer Science, Memorial University of Newfoundland.

[42] H. Todd Wareham. 1995. Parameterized Complexity Analysis in Computational Phonology. Manuscript. Submitted to the Annual Meeting of the Association for Computational Linguistics (*ACL-96*); not accepted.

[43] H. Todd Wareham. 1996. The Role of Parameterized Computational Complexity Theory in Cognitive Modeling. AAAI-96 Workshop Working Notes: *Computational Cognitive Modeling: Source of the Power*.