Remarks and Replies

Evaluating the Complexity of Optimality Theory

Jeffrey Heinz
Gregory M. Kobele
Jason Riggle

Idsardi (2006) claims that Optimality Theory (OT; Prince and Smolensky 1993, 2004) is ''in general computationally intractable'' on the basis of a proof adapted from Eisner 1997a. We take issue with this conclusion on two grounds. First, the intractability result holds only in cases where the constraint set is not fixed in advance (contra usual definitions of OT), and second, the result crucially depends on a particular representation of OT grammars. We show that there is an alternative representation of OT grammars that allows for *efficient computation* of optimal surface forms and provides deeper insight into the sources of complexity of OT. We conclude that it is a mistake to reject OT on the grounds that it is computationally intractable.

Keywords: Optimality Theory, computational complexity, generation problem

1 Introduction

An Optimality Theory (OT) grammar (Prince and Smolensky 1993, 2004) can be thought of as a function that maps underlying forms to surface forms.[1] Idsardi (2006), adapting a proof from Eisner (1997a), claims to have shown that computing optimal surface form(s) can be as hard as computing solutions to the Hamiltonian-graph problem—that is, NP-hard. Wareham (1998) makes similar claims using a different NP-hard problem, DOMINATING SET (see Garey and Johnson 1979: 75).

However, these results are not the whole story. The hardness proofs presented by Eisner, Idsardi, and Wareham all rely on two assumptions: (a) that the constraint set is not fixed in

The authors would like to thank Ed Stabler, Bill Idsardi, and Colin Wilson for useful discussion, and more generally, all those responsible for the intellectually stimulating environment at UCLA, where this article was originally conceived.

[1] Because multiple candidates can have identical sets of violations (i.e., because ties are possible), it is more accurate to say that the grammar is a function mapping underlying forms to sets of violation-equivalent candidates.

Linguistic Inquiry, Volume 40, Number 2, Spring 2009
277–288

advance but instead forms part of the ''input'' to the optimization problem, and (b) that OT grammars are represented as lists of individual constraints. If the constraint set CON is fixed and therefore not a variable parameter in the generation task, then optimal surface form(s) can in fact be computed efficiently (Ellison 1994). Furthermore, even if the constraint set is not fixed in advance, these hardness results hold only for a constraint-list representation of the grammar and not necessarily for other representations (we provide an alternative, efficient representation below). These qualifications, which we explain in detail, are important in understanding the kinds of conclusions that can be drawn from studies of computational complexity of grammars.

Kornai (2006b; see also Kornai 2006a) has already pointed out that Eisner's and Idsardi's NP-hardness results hold only if the number of long-distance assimilatory and dissimilatory constraints is unbounded. If the number of such constraints is fixed a priori (which follows under most characterizations of OT), then the problem of computing surface forms cannot be NP-hard because the size of the problem instance is dominated by the size of the underlying form for all but finitely many inputs. This limiting-case perspective is at the heart of complexity theory and is why Ellison (1994) asserts that OT generation is efficient. Ellison shows that it is possible to generate optimal surface forms for underlying forms of length n in $nC \log(nC)$ steps—where C is a constant contributed by the constraint set. Thus, regardless of how large C is, it is dwarfed by n in almost every case.

In this article, we focus on clarifying the perspective that computational complexity theory brings to the analysis of OT.[2] In addition to drawing distinctions between *simple generation problems* and *universal generation problems* (cf. Garey and Johnson's (1979) distinction between the membership and universal membership problems), we introduce a third category, what we call *quasi-universal generation problems*, that we argue best captures the standard definition of OT. As Wareham (1996) points out, ''complexity-theoretic analysis . . . is an ongoing dialogue . . . to fully characterize the problem by showing which restrictions make the problem tractable and which don't.'' We show that this parameterization of the problem (cf. Downey and Fellows 1999) reveals that Eisner's and Idsardi's hardness results hold only for the universal generation problem. Furthermore, Eisner's and Idsardi's results, like Wareham's, depend crucially on assumptions about how OT grammars are represented. We provide an alternative representation for OT grammars, one that allows efficient computation of optimal surface forms and provides deeper insight into the complexity of OT. Consequently, though these earlier results are unassailable on purely formal grounds, it would be inappropriate to think of them as establishing that OT is ''computationally intractable.''

[2] Smolensky, Legendre, and Tesar (2006) summarize key results, concluding from Ellison 1994, Tesar 1995 et seq., Eisner 1997a, and Frank and Satta 1998 that efficient generation is possible for any given OT grammar but not for all OT grammars. Our analysis aims to clarify which generation problems are potentially hard and how the representation of the grammar matters in these cases.

2 Computational Complexity

2.1 Distinguishing the Problems

Complexity theory is about how hard various kinds of problems are to solve. As an example, take the problem of deciding whether or not a sentence *s* is produced by a Lambek categorical grammar *G*. In this problem, there are two parameters: the string *s* and the grammar *G*. If we allow the grammar to vary (i.e., if we are asking the so-called universal question, How hard is it to determine whether an arbitrary string is in the language of an arbitrary grammar?), this problem is NP-complete (Pentus 2006). On the other hand, we might be more concerned with a *particular* grammar *G*, and want to know how hard it is to determine whether an arbitrary string is in the language defined by *G*. In this case, because we ''know *G* in advance,'' we are free to perform as much preprocessing on *G* as we like, and the problem is solvable in polynomial time. This contrasts starkly[3] with the recognition problem for context-free grammars (which define exactly the same languages as Lambek categorical grammars (Pentus 1993)), for which both the universal and the nonuniversal problems are solvable in polynomial time.

OT theorists are often interested in the *generation* problem: the problem of finding the optimal output(s) for an input given a ranked set of constraints. Here there are three relevant parameters: the input (string) *s*, the constraint set Con, and the ranking *R*. It is useful to have names for the three most relevant problems: we call them the *simple* problem (where only the input string may vary but the constraint set and ranking are held fixed), the *universal* problem (where all are allowed to vary), and the *quasi-universal* problem (where the string and the ranking are allowed to vary but the constraint set is held fixed). We summarize these three problems in table 1, where $G_{Con,R}$ is the OT grammar that maps underlying forms to surface forms.

All three problems give insight into the nature of generation in OT. Which ones are most relevant depends of course on the real-world situation we think humans face. Under the usual characterization of OT, the constraint set is fixed and universal (i.e., ''given''), a fact that makes either the simple problem or the quasi-universal problem, but not the universal problem, most

Table 1
Three generation problems in Optimality Theory

	Given	Input
Simple	Con, R	x
Quasi-universal	Con	R, x
Universal	∅	Con, R, x
Problem	What is $G_{Con,R}(x)$?	

[3] Assuming that $P \neq NP$.

relevant. However, if we adopt a view of OT in which the constraint set is learned and language-specific (let us call this version *open OT*), then the universal problem becomes more relevant.

We show that, assuming that all constraints are finite-state,[4] the first two variants are efficiently solvable and that, under conditions to be discussed, the third is as well. We argue that the conclusion that OT is intractable arises as an artifact of the grammar representations used by Eisner, Idsardi, and Wareham. We show that, in fact, a *different representation* of constraint sets as a single machine, EVAL, not only allows for the universal problem to be solved efficiently, but also is more natural from a complexity-theoretic point of view.

2.2 Key Concepts of Computational-Complexity Analysis

In this section, we introduce the basic concepts needed to measure the complexity of a problem. An informal introduction follows here, and standard formal definitions are provided in an appendix. Interested readers may consult some of the standard texts on computational complexity theory (e.g., Papadimitriou 1994) for more detail.

Informally, a problem is thought of as a function that maps instances of the problem to its answers. Essentially, a problem is polynomial time computable if and only if there is a Turing machine that implements this function in fewer than $f(n)$ steps, where f is a polynomial function and n is the length of the input to the machine (this input is the problem instance). Note that there can be multiple ways of representing the same instance and so the standard definition assumes the adoption of some uniform representational scheme. For problems whose instances have multiple parameters, only parameters that are allowed to vary are standardly included as part of the problem instance.

It follows immediately that

1. when measuring the complexity of a problem, the representation of the problem instance matters because the complexity of the problem is determined partly by the size of the representation of the problem instance; and
2. fixed parameters are not part of a problem instance, and thus the size of fixed parameters' representation does not play a role in determining a problem's complexity.

With respect to the problem of generation in OT, it follows that if a parameter—such as the constraint set—is fixed, then the size of its representation is irrelevant in determining the time complexity of the problem. It also follows that in the universal problem, the sizes of the representations of all three parameters do matter. However, in that case, the choice of representation matters as well. We elaborate on these points in the next section, which applies these definitions to the three problems described above.

[4] As observed by Eisner (1997b), without constraints on what counts as a possible constraint, there are no limits on the complexity of OT (pick a constraint that assigns a violation to a string just in case it is not a theorem of first-order logic, and let it outrank all other constraints). The majority of phonological phenomena seem amenable to finite-state treatment, and most constraints proposed by phonologists working within OT have natural finite-state analogues. Eisner, Idsardi, and Wareham each adopt constraints that can be represented individually as finite-state machines.

3 Evaluating the Complexity of Optimality Theory

3.1 The Simple Problem

The simple generation problem in OT, which asks for the optimal surface form(s) given an arbitrary underlying form *s* but a fixed constraint set CON and ranking *R*, is close to linear in the length of *s* (as shown by Ellison (1994) and noted by Eisner (1997a)). Ellison's proof is simple: he presents a general strategy for constructing a finite-state device representing CON under a total ranking *R*, for which computing the optimal surface form is nearly linear in the length of the underlying form. If the constraint set and ranking are given in advance, and are not a parameter of the problem, it is actually quite easy to determine the output for any particular input.

Compared with the universal problem, some might consider the simple problem to be too simple to be of linguistic interest. Nonetheless, understanding the complexity of the simple generation task clarifies the scope of any hardness results and helps to clarify the nature of complexity in the other generation problems for OT.

3.2 The Quasi-Universal Problem

We take the quasi-universal problem to be the standard generation problem in OT. Here, we are asked to generate a surface form given any possible underlying form and any ranking of a fixed constraint set.[5]

Considering the definitions in the appendix, it is clear that Idsardi's complexity result (likewise Eisner's) does not address the quasi-universal problem any more than it does the simple problem. In Idsardi's reduction of the Hamiltonian graph problem, one long-distance agreement constraint of the form *x . . . x is included for each phoneme x. Since each node in the graph represents a unique phoneme, the number of phonemes is determined by the number of nodes.[6] Thus, the number of constraints needed in the reduction is not fixed. In the same way that the number of nodes in the Hamiltonian graph problem is a parameter of the problem, the number of constraints becomes a parameter of the problem that Idsardi shows to be NP-hard.

Wareham's (1998) complexity result relies on a similar recasting of grammars as arbitrary graphs. Wareham provides a method whereby constraints represented as finite-state automata are intersected to create graphs in which optimization reveals whether the graph contains a *dominating set* of size *k* (i.e., given a graph with edges *E* and vertices *V*, is there a set of vertices $X \subseteq V$ of size at most *k* such that every vertex in *V* is either in *X* or adjacent to a vertex in *X*?). As with Eisner's and Idsardi's results, Wareham's result crucially requires an open constraint set to be

[5] It is true that, as Idsardi (2006) points out, OT analyses that adopt lexical-item- or morpheme-class-specific constraints are in conflict with the assumption of a closed constraint set. To the extent that such analyses are necessary and cannot be recast as language-specific parameterizations of constraints drawn from a closed universal set, they call into question the utility of this assumption.

[6] At first glance, this suggests that Idsardi's proof requires an unbounded number of phonemes, contra the traditional assumption of a fixed set of features. However, it is possible to recast Idsardi's proof with a finite number of phonemes in two steps. First, in graphs with more nodes than phonemes, each node can be represented with sequences of phonemes. Second, adopt constraints *x . . . x, where now x represents a sequence of phonemes used to represent some node. There is precedent for this kind of sequence-based markedness constraint in the literature (Evans 1995, Yip 1995).

able to pose the dominating set query for arbitrarily large graphs. Thus, Eisner, Idsardi, and Wareham are all investigating the complexity of the universal generation problem for OT and not the simple generation problem or the quasi-universal generation problem.

The quasi-universal problem, like the simple problem, can be solved in linear time. Riggle (2004) shows how to represent CON as a finite-state device *independently* of any ranking.[7] In this way, Riggle's finite-state representation differs from Ellison's (1994) representation, which encodes both CON and the ranking. Riggle's device can be thought of as a kind of meta-EVAL because, once given a ranking, it instantiates that particular EVAL. Riggle (2004:156) shows that from this meta-EVAL, computing the optimal surface forms given an arbitrary underlying form *and an arbitrary ranking* can be done in linear time. This means that it is possible to generate optimal candidates for any (or all) of the grammars under factorial permutation of CON without needing to recompute EVAL. Thus, the quasi-universal problem, where the constraint set but not the ranking is fixed, is efficiently computable.

3.3 The Universal Problem

Eisner's, Idsardi's, and Wareham's results apply to the universal problem. In this problem, neither the constraint set, nor the ranking, nor the underlying form is known in advance. This problem is most relevant for those who want to write programs to aid phonologists who use OT or for those who do not subscribe to the idea of a bounded constraint set.

Recall from section 2.2 that the representations of parameters matter when determining bounds on time complexity. For this reason, what Eisner, Idsardi, and Wareham have shown should be interpreted more narrowly. They have shown that the universal generation problem *under a particular representation of OT grammars*—where grammars are represented by a sequence of finite-state constraints—is NP-hard. Note that because the size of the representation of the grammar forms part of the argument to the complexity function, the *choice of representation* plays a crucial role in the complexity of universal problems. In other words, whether a universal generation problem is computable in polynomial time depends (in part) on the representation of the grammar.

For instance, given an arbitrary enumeration of finite-state transducers, we can represent OT grammars as sequences of natural numbers (where the first number corresponds to the first occurrence of the highest-ranked constraint in the enumeration, the second to the first occurrence of the next highest, and so on). If the enumeration is not computable, then neither is the universal generation problem (as we will not be able to decode the grammar from its representation). Idsardi and Eisner have shown that if grammars are represented as sequences of finite-state transducers (in effect, precompiling the decoding necessary given the above representation), then the universal generation problem is computable, though NP-hard. On the other hand, if an OT grammar is represented, as suggested by Ellison (1994), as a single finite-state transducer (the result of combin-

[7] This representation of CON is necessary for generating complete sets of non–harmonically bounded candidates in Riggle's CONTENDERS algorithm.

ing the individual constraints into a single evaluating function; e.g., Riggle's (2004) meta-EVAL), then even the universal generation problem is efficiently computable.

The difference between Ellison's proposed representation of the grammar and Eisner's is that in Ellison's representation some of the work has already been done by combining all of the constraints into a single function via an operation similar to intersection. It is easy to see why this representation makes such a striking difference in the hardness of the problem. Eisner's hard cases are hard precisely because they necessitate the intersection of a large number of finite-state machines—a process whose difficulty is well known for growing explosively in the worst cases (Hopcroft, Motwani, and Ullman 2001).

Representing the grammars as ranked individual finite-state constraints is more compact and can, as Eisner (1997a) demonstrates, facilitate some optimizations, but this representation results in intractability for the universal generation problem because computing the intersection of arbitrary sets of finite-state machines cannot be done efficiently. On the other hand, representing the grammar as a single finite-state transducer may yield a less concise representation, but compensates for this by eliminating the intractability arising from the run-time combination of all of the constraints into a single machine.

The representation of an OT grammar as EVAL—a single finite-state machine that combines the whole constraint set and ranking into one evaluation function—is more than just a clever ''padding out'' of the input to the universal generation problem in order to ensure that there is enough time to perform a number of steps exponential in the length of the size of the constraint-list representation; it shows something deep about the problem of generation in OT. While the number of steps needed to compute an answer to a particular instance of the universal generation problem is upward-bounded by an exponential function of the size of the constraint-list representation (plus the length of the input), the *actual* number of steps needed may vary wildly below this upper bound, depending on the specific constraints involved. When the grammar is represented as EVAL, on the other hand, the number of steps needed to compute an answer to a particular instance of the universal generation problem is tightly related to the size of the grammar. In other words, the size of EVAL is a much better predictor of the actual number of steps needed to solve the generation problem than is the size of the constraint-list representation.

For a concrete example, let us define the size of a constraint-list representation of an OT grammar to be the sum of the number of states in the constraints, and the size of the EVAL representation to be the number of states in EVAL.[8] Let us assume that simple markedness constraints like *X, *XY, and *X . . . X (where X, Y are phonemes) require one, two, and two states for their representations, respectively.[9] Now consider the following three grammars. G1 consists of 20 markedness constraints of the form *X, where X is a different phoneme in each constraint. G2 consists of 10 markedness constraints of the form *XY, where X is a different phoneme in each. G3 consists of 10 markedness constraints of the form *X . . . X, where X is a different

[8] We can omit the arcs from the size of the machines when all constraints are deterministic because in such a case the number of arcs will be a multiple of the number of states. If constraints are nondeterministic, the number of arcs should figure into the size as well.

[9] For details and justification, see Ellison 1994, Eisner 1997a, Riggle 2004, and Albro 2005.

phoneme in each. Under the constraint-list representation, these three grammars are all the same size: 20 states. On the other hand, under the EVAL representation, G1 has 1 state, G2 has 11 states, and G3 has 1,024 states. In each case, the complexity of the universal generation problem is linear in the size of the underlying form plus EVAL.[10]

The fact that the EVAL representation can be much larger than the constraint-list representation is not especially relevant. While the latter can be more compact, the former is always more transparent (in terms of the complexity of using the grammar). The relevance of this representation scheme is that

1. if CON and the ranking R are represented as a single device, then even the universal problem is efficiently solvable; and
2. the EVAL representation is more natural from a computational point of view because it is a better predictor of the number of steps needed to solve a given instance of the universal generation problem.

In short, the reason the size of the constraint-list representation is such a poor predictor of the number of steps required to solve a particular instance of the universal generation problem is that it is a poor predictor of the size of EVAL, which *is* directly correlated with this number. Thus, the NP-hardness results of Eisner, Idsardi, and Wareham should not be viewed as properties of OT per se; rather, they should be viewed as reflecting the difficulty of translating the constraint-list representation into the EVAL representation.

4 Choosing between Representations

The EVAL representation and the constraint-list representation are different modes of presenting the same function. There is no a priori way to decide which of the possible representations for a grammar is ''right'' (it is not even clear what form an answer to this question would take). Instead, we need to ask which representation is most useful or relevant for a particular situation. Investigating the complexity of many different problems is useful precisely because it allows us to focus our linguistic investigation on those situations where complexity is reasonable.

Consider the problem of learning an OT grammar. If we think that a learner deals with constraints directly (perhaps by inventing new ones), then we will not be able to circumvent the costly computation of EVAL from constraints. The results presented by Eisner, Idsardi, and Wareham tell us that in this case it will be important that such a learner does not hypothesize arbitrary constraints, because doing so may require unreasonable amounts of work in the construction of EVAL. Instead, we should look for restrictions on the kinds of constraints hypothesized, which make their combination ''easy.'' However, if we are able to state the relevant steps in the

[10] Each state in EVAL corresponds to a unique phonological environment that the grammar is sensitive to. Consequently, constraints that refer to the same environments do not increase the size of EVAL in the intersection process, but constraints that refer to different environments can multiply the number of states in EVAL. In this way, this representation of EVAL makes concrete for OT a notion from *SPE* (Chomsky and Halle 1968): that the introduction of a phonological rule that refers to the same environment as another rule is less costly in terms of the size of the grammar as a whole than a rule that introduces a new environment.

learning process directly over a machine representation of EVAL,[11] then Eisner's, Idsardi's, and Wareham's results are not relevant for this case. In other words, since we know (thanks to Eisner, Idsardi, and Wareham) that doing optimization with EVAL built from lists of individual constraints is hard, we should think of ways to avoid having to postulate that it gets done.

Finally, we address some of the concerns that have been raised about representing an OT grammar as a finite-state implementation of EVAL. Eisner (1997a) gives three reasons why this might be problematic.

Two of these concerns are of a more practical nature, focusing as they do on the current feasibility of large-scale implementation of OT grammars on computers. The first is that although the complexity function is polynomial (or even linear), there may be very large constants (due to the large size of this representation of the grammar) that make the computation impractical. The second is related: not all candidate sets may be concisely represented by finite-state automata.

We agree that these are real concerns, relevant to the purpose that Eisner has in mind (which is, in this case, large-scale implementation of OT grammars so as to provide practical assistance to phonologists and corporations). However, as the questions we are asking here relate more to stable complexity-theoretic properties than those that concern Eisner (in this particular section of the cited paper), and as our interests are not limited to what is currently technologically feasible, but instead relate to what may ultimately become so, these concerns are not relevant here.[12]

Only the final point raised by Eisner is addressed at the level of investigation at which work on complexity theory, and generative linguistics, is conducted. Eisner suggests that representing the grammar as a list of ranked constraints instead of as EVAL is more relevant to a theory of phonology because ''the grammar is not fixed in all circumstances: both linguists and children crucially experiment with different theories'' (1997a:319). We agree that the NP-hardness results of Eisner, Idsardi, and Wareham have real ramifications for a theory of learning in OT. Their results should be taken together with Ellison's to point out the direction that future learning models should explore; we should look for learning algorithms that do not often compute EVAL from arbitrary constraints (note that learning algorithms that just rank a fixed set of constraints meet this condition).

5 Conclusions

Computational complexity theory investigates how the difficulty of solving particular problems grows as the size of the problem instances grows. Applying complexity-theoretic analyses to computational theories can give insight into how we should move forward with them, as well as

[11] For instance, if it were possible to efficiently add and subtract constraints in EVAL constructed incrementally over several observations, then, if the number of new constraints for any given generation task was bounded, the complexity of each modification to EVAL would be bounded as well.

[12] Regarding constant factors, a textbook on complexity theory has this to say:

> Constants are, of course, of great interest to computer science. Unfortunately, it does not seem possible to build an elegant theory that accounts for them. Furthermore, advances in hardware have in the past improved the performance of computers so rapidly, that algorithm designers can compete only by improving the *rate of growth* of the time requirements of their algorithms. (Papadimitriou 1994:32; emphasis in the original)

into the kinds of data structures we should use to implement them (Marr 1982). Garey and Johnson (1979:3) write that ''discovering that a problem is NP-complete is usually just the beginning of work on that problem.'' It is in this spirit that we have endeavored here to precisely identify the sources of complexity in OT and to explore versions of the generation problem in OT that do not run afoul of intractable complexity.

The results of Eisner, Idsardi, and Wareham show that, even if constraints are finite-state and ''natural looking,'' the universal generation problem is intractable when grammars are represented as lists of constraints. We have identified the process of combining the constraints (intersection) as the source of hardness. Given that complexity theory should be used to help determine how the theory should develop, we are led to ask what other conditions or formulations make the universal problem not hard. Idsardi offers some suggestions, such as placing restrictions on GEN. Likewise, Wareham suggests placing additional restrictions on CON. Here we have brought to light another interesting alternative: namely, that tractability can be maintained in the universal problem if we are willing to avoid learners that compute EVAL from individual constraints. This is simply done if we stick to the standard version of OT in which constraints are antecedently given (i.e., the quasi-universal problem), but it can also be done even if we move to open OT (where the learner is free to invent constraints). In the latter case, generation in open OT might remain tractable if learning algorithms are stated directly over an EVAL-like representation. We hope these remarks spur interest in such learners, in finite-state approaches to OT and phonology, and in formal investigations of linguistic theory more generally.

Appendix: Formal Definitions

We first define polynomial-time computability over problems with one parameter and then generalize the definition to problems with multiple parameters. We identify problems p with functions $\gamma : A \rightarrow B$, where A is the set of *instances* of p, and B is the set of possible answers to the problem (*yes* and *no*, in the simplest case). A function $\gamma : A \rightarrow B$ is said to be *polynomial-time computable* in $f : \mathbb{N} \rightarrow \mathbb{N}$ under the representation $r : A \cup B \rightarrow \Sigma^*$ if and only if there is some Turing machine M_γ over alphabet Σ, and an encoding $r_a = r(a)$ of each $a \in A$ and $r_b = r(b)$ of each $b \in B$ into the language of the Turing machine (i.e., r is a one-to-one function), such that when M_γ is started with a representation r_a of any $a \in A$ on its input tape, it stops after no more than $f(|r_a|)$ steps, where

1. $|r_a|$ is the length of r_a, and
2. f is a polynomial function, and
3. either
   a. the machine M_γ is in a rejecting state if γ is not defined on a, or
   b. M_γ is in an accepting state with r_b on its output tape and $\gamma(a) = b$.

Note that the time complexity function f is determined with respect to the length of the representation of an instance a and that there can be different representational schemes r for the set $A \cup B$. In other words, the time complexity function is dependent not only on γ, but also on the choice of the representation r.

When there are multiple parameters in our problem p, as is the case in the OT generation problem, we treat the domain A of the function γ representing the problem as the set of sequences $A = A_1 \times \cdots \times A_n$, where A_i is the set of possible instances of the ith parameter of p. Given a function $\gamma: A_1 \times \cdots \times A_n \rightarrow B$, the problem obtained by ''holding the first parameter constant (with the value $\bar{a}$)'' is represented by the function $\gamma_{\bar{a}}: A_2 \times \cdots \times A_n \rightarrow B$, such that, on input $a_2, \ldots, a_n$, $\gamma_{\bar{a}}$ behaves just as γ does on input $\bar{a}, a_2, \ldots, a_n$ (i.e., $\gamma_{\bar{a}}(a_2, \ldots, a_n) = \gamma(\bar{a}, a_2, \ldots, a_n)$). (Holding nonfirst parameters constant is defined analogously.) Definitionally, parameters that are held constant are no longer part of the problem instance.

This matters in determining what bounds (if any) exist over the time complexity function f. When there are multiple parameters, the length of the representation of the vector $a = (a_1, a_2, \ldots, a_n) \in A$ is equivalent to the sum of the length of its parts: $|r_a| = |r_1(a_1)| + |r_2(a_2)| + \ldots + |r_n(a_n)|$. This means that if the first parameter is held constant with the value $\bar{a}$, then the length of the representation of $\bar{a}$ (i.e., $|r_1(\bar{a})|$) does not figure into the length of the representation of the problem r_a, which forms the argument to the time complexity function f. Consequently, parameters that are held constant (i.e., fixed) are not included as part of the problem instance, and thus the size of their representation is not a factor in determining the time complexity of the problem itself.

References

Albro, Daniel. 2005. A large-scale LPM-OT analysis of Malagasy. Doctoral dissertation, UCLA, Los Angeles, CA.

Chomsky, Noam, and Morris Halle. 1968. *The sound pattern of English*. New York: Harper and Row.

Downey, Rod G., and Michael R. Fellows. 1999. *Parameterized complexity*. Berlin: Springer-Verlag.

Eisner, Jason. 1997a. Efficient generation in primitive Optimality Theory. In *Proceedings of the 35th Annual ACL and 8th EACL*, 313–320. Madrid.

Eisner, Jason. 1997b. What constraints should OT allow? Handout from a talk presented at the annual meeting of the Linguistic Society of America, Chicago. Rutgers Optimality Archive ROA 204-0797. Available at http://roa.rutgers.edu/.

Ellison, Mark. 1994. Phonological derivation in Optimality Theory. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, 2:1007–1013. Kyoto.

Evans, Nick. 1995. Current issues in the phonology of Australian languages. In *The handbook of phonological theory*, ed. by John Goldsmith, 723–761. Cambridge, MA: Blackwell.

Frank, Robert, and Giorgio Satta. 1998. Optimality Theory and the generative complexity of constraint violability. *Computational Linguistics* 24:307–315.

Garey, Michael R., and David S. Johnson. 1979. *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco: W. H. Freeman.

Hopcroft, John, Rajeev Motwani, and Jeffrey Ullman. 2001. *Introduction to automata theory, languages, and computation*. Boston: Addison-Wesley.

Idsardi, William. 2006. A simple proof that Optimality Theory is computationally intractable. *Linguistic Inquiry* 37:271–275.

Kornai, András. 2006a. Guarded optimalism. Rutgers Optimality Archive ROA 841-0606. Available at http://roa.rutgers.edu/.

Kornai, András. 2006b. Is OT NP-hard? Rutgers Optimality Archive ROA 838-0606. Available at http://roa.rutgers.edu/.

Marr, David. 1982. *Vision*. San Francisco: W. H. Freeman.

Papadimitriou, Christos H. 1994. *Computational complexity*. Reading, MA: Addison-Wesley.
Pentus, Mati. 1993. Lambek grammars are context free. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*, 429–433. Los Alamitos, CA: IEEE Computer Society Press.
Pentus, Mati. 2006. Lambek calculus is NP-complete. *Theoretical Computer Science* 357:186–201.
Prince, Alan, and Paul Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Technical report 2, Rutgers University, New Brunswick, NJ, and University of Colorado, Boulder.
Prince, Alan, and Paul Smolensky. 2004. *Optimality Theory: Constraint interaction in generative grammar*. Oxford: Blackwell. Revised version of Prince and Smolensky 1993.
Riggle, Jason. 2004. Generation, recognition, and learning in finite state Optimality Theory. Doctoral dissertation, UCLA, Los Angeles, CA.
Smolensky, Paul, Géraldine Legendre, and Bruce Tesar. 2006. Optimality Theory: The structure, use, and acquisition of grammatical knowledge. In Paul Smolensky and Géraldine Legendre, *The harmonic mind: From neural computation to optimality-theoretic grammar*. Vol. 1, *Cognitive architecture*, 453–544. Cambridge, MA: MIT Press.
Tesar, Bruce. 1995. Computational Optimality Theory. Doctoral dissertation, University of Colorado, Boulder.
Wareham, H. Todd. 1996. The role of parameterized computational complexity theory in cognitive modeling. In AAAI-96 Workshop Working Notes: *Computational cognitive modeling: Source of the power*. Available at http://web.cs.mun.ca/~harold/Papers/AAAI.pdf.
Wareham, H. Todd. 1998. Systematic parameterized complexity analysis in computational phonology. Doctoral dissertation, University of Victoria.
Yip, Moira. 1995. Repetition and its avoidance: The case of Javanese. In *Proceedings of the 1995 Southwestern Workshop on Optimality Theory*, ed. by Keiichiro Suzuki and Dirk Elzinga, 238–262. Tucson: University of Arizona, Department of Linguistics.

(Heinz)
Department of Linguistics and Cognitive Science
University of Delaware
42 E. Delaware Ave.
Newark, DE 19716

heinz@udel.edu

(Kobele)
Institut für deutsche Sprache und Linguistik
Humboldt University of Berlin
Unter den Linden 6
10099 Berlin
Germany

kobele@rz.hu-berlin.de

(Riggle)
Department of Linguistics
University of Chicago
1010 E. 59th St.
Chicago, IL 60637

jriggle@uchicago.edu