Remarks and Replies

A Simple Proof That Optimality Theory Is Computationally Intractable

William J. Idsardi

Adapting arguments from Eisner 1997, 2000, this remark provides a simple proof that the generation problem for Optimality Theory (Prince and Smolensky 2004) is NP-hard. The proof needs only the binary evaluation of constraints and uses only constraints generally employed in the Optimality Theory literature. In contrast, rule-based derivational systems are easily computable, belonging to the class of polynomial-time algorithms, P (Eisner 2000).

Keywords: Optimality Theory, computational complexity, NP-hard

1 Introduction

Eisner (1997), building on work by Ellison (1994), offers a proof that the generation problem for Primitive Optimality Theory is NP-hard. This proof involves constructing ''an intermediate candidate set consisting of all permutations of *r* digits'' (Eisner 2000:29). Problems involving permutation sets are well known in the literature on computational complexity and intractability (see, e.g., Garey and Johnson 1979; and, for specifically linguistic examples, Barton, Berwick, and Ristad 1987). However, Eisner's use of an idiosyncratic version of Optimality Theory (OT) and the unusual nature of some of the constraints employed in his proof have limited the impact of this important result. In this remark, I reconstruct Eisner's proof using only constraint types attested in actual OT analyses (e.g., Prince and Smolensky 2004, Kager 1999, Ito and Mester 2003). Furthermore, as suggested in Eisner 2000, I achieve this result assuming only the binary evaluation of constraints (i.e., constraints are satisfied or violated only once per candidate; there is no gradient evaluation of any sort).

2 Proof

The standard method for proving that a new problem is NP-hard is to encode a known intractable problem using the vocabulary and resources of the new problem. If there is a polynomial-time

I would like to thank the anonymous reviewers for their comments, and especially for their advice on clarifying the difference between NP-hard and NP-complete problems.

Linguistic Inquiry, Volume 37, Number 2, Spring 2006
271–275

reduction to the new problem, then the new problem is NP-hard,[1] as Garey and Johnson (1979) explain.

> Once we have proved a single problem NP-complete, the procedure for proving additional problems NP-complete is greatly simplified. Given a problem $\Pi \in$ NP, all we need do is show that some already known NP-complete problem Π' can be transformed to Π. (Garey and Johnson 1979:45)
>
> Although we have restricted our discussions so far mainly to problems that belong to NP, it should be apparent that the techniques used for proving NP-completeness also can be used for proving that problems outside of NP are hard. Any decision problem Π, whether a member of NP or not, to which we can transform an NP-complete problem will have the property that it cannot be solved in polynomial time unless P = NP. We might say that such a problem Π is ''NP-hard,'' since it is, in a sense, at least as hard as the NP-complete problems. (Garey and Johnson 1979:109)

The NP-complete problem to be transformed, as in Eisner 1997, is the directed Hamiltonian path problem (Garey and Johnson 1979:60, 199, Cameron 1994:167). (A Hamiltonian path is a tour that does not revisit any of the vertices in a graph.)

> A Hamiltonian path in a directed graph $G = (V, A)$ is an ordering of V as $\langle v_1, v_2, \ldots, v_n \rangle$, where $n = |V|$, such that $(v_i, v_{i+1}) \in A$ for $1 \leq i < n$. (Garey and Johnson 1979:60)

The trick now is to construct an OT grammar that computes directed Hamiltonian paths. We begin by taking an alphabet (i.e., a phoneme inventory) of n elements $I = \{\text{a, b, c, } \ldots\}$ representing the vertices of G in the obvious way (a = v_1, b = v_2, etc.). We then form an input word of n elements, /abc . . . / (in fact, any input word of n elements is sufficient). The infinite candidate set generated from this input by *Gen* is then evaluated by the following constraint hierarchy. I will present the constraints in a stratified hierarchy for ease of exposition; this is not in fact necessary for the proof, the ordering over the relevant constraints being immaterial since the solutions (the winning candidates) will not violate any constraints.

In the first stratum, we have the set of constraints of the form $*x$, where $x \notin I$. This eliminates any candidates with phonemes not in the alphabet for the problem. In OT, such constructions are used to ban certain sounds entirely from a language; examples are the statement that English does not contain any clicks and the statement that English does not contain the high front rounded vowel [y] (see Ito and Mester 2003:17–18). This reduces the candidate set to the infinite set I^*, the set of strings over I.

In the next stratum, we have MAX and DEP for segments, whose combined effect is to limit the candidate set to strings of the same length as the input string. This reduces the candidate set at this point to the finite set I^n, which has n^n candidates. Nothing more is needed to do this work

[1] A problem is NP-hard if it is at least as hard as any of the problems in NP. An NP-complete problem is an NP-hard problem that is also known to be in NP. In some circumstances, then, NP-hard problems can be harder to solve (i.e., even more intractable) than NP-complete problems; see Garey and Johnson 1979 for further discussion. Restricted subsets of OT, such as those using only the binary-evaluated constraints employed in section 2, can be shown to be within NP. However, since there is no clear proposal specifying the class of allowable constraints in OT, it is not possible to show generally that OT problems lie within NP; hence, the results show that OT generation is NP-hard, though, in fact, it may be even harder.

than a binary evaluation of MAX and DEP over the entire form. We do not even need to assume the unbounded evaluation of constraints invoked in McCarthy 2003.

> It is sufficient for any constraint to assign one violation-mark for *each instance* of the marked structure or unfaithful mapping in the candidate under evaluation. (McCarthy 2003:130; emphasis added)

Since there are candidates without *any* MAX or DEP violations, it is unnecessary to assign more than one violation mark per candidate for present purposes. It is not necessary to assign violation marks for each MAX or DEP violation, as the candidate will be ruled out at the first such violation.

In the next stratum, we have the set of self-conjoined constraints (Ito and Mester 2003:29) $*\alpha^2 = \{*a^2, *b^2, *c^2, \ldots\}$. Self-conjoined constraints forbid two instances of the same segment (or segment type in the case of incomplete feature specifications) within the same domain, here the word (i.e., the whole candidate). Such constraints are found in Japanese (Lyman's Law), Sanskrit (Grassmann's Law), Semitic (Greenberg 1950), and many other languages, as discussed by Ito and Mester (2003:30 and passim). Again, these constraints can be evaluated in a binary fashion, as a single violation will be enough to rule out a candidate, there being candidates without any violations of these constraints. At this point, the candidate set has been reduced to the set of strings of length n without repetitions, that is, the permutations over I, which has $n!$ candidates. We have now constructed a brute-force enumeration of all possible paths over V, regardless of whether or not the edges in the paths are elements of A.

In the next stratum, we have the constraints against including sequences of vertices not in G, that is, restricting the candidates to those with paths drawn from A. For this purpose, we have constraints of the form $*\alpha\beta$ where $\alpha = v_i$, $\beta = v_j$, and $v_i v_j \notin A$. Such constraints against certain bigram sequences are the bread and butter of standard phonological analyses, as they bar certain sequences of segments, such as *NC̥ (Kager 1999, Ito and Mester 2003) and countless other cases. All candidates remaining after this stratum are directed Hamiltonian paths (so, if desired, we can insert the MPARSE constraint at this point in the hierarchy, so that the null candidate will win if there are no directed Hamiltonian paths). Thus, we have transformed an NP-complete problem (finding directed Hamiltonian paths) into an OT problem and therefore have demonstrated that the generation problem for OT is NP-hard. The transformation is obviously polynomial-time because the construction of the input is linear-time and the construction of the required constraint hierarchy is at most quadratic-time, the input having n elements and the constraint hierarchy having at most $n^2 + n + 3$ constraints (n^2 possible sequence constraints; n self-conjoined constraints; and MAX, DEP, and MPARSE).

3 Conclusions

In this remark, following the work of Eisner (1997, 2000), I have offered a simple proof that the generation problem for OT is NP-hard. Under the current understanding of computational complexity results for which $P \neq NP$ (even though this hypothesis still evades proof), this makes OT in general computationally intractable. In contrast, rule-based derivational systems are easily computable, belonging to the class of polynomial-time algorithms, P (Eisner 2000:32). Other NP-complete problems could serve as the basis for additional such proofs, including for example

3SAT by adapting the Kimmo 3SAT argument presented in Barton, Berwick, and Ristad 1987 and PATH WITH FORBIDDEN PAIRS (Garey and Johnson 1979:203) by adapting the present proof.

The present proof uses only binary evaluation of constraints and employs only simple OT constraints of types generally found in OT analyses of natural languages. The crux of the problem, as pointed out by Eisner (2000:29), is the generation of an intermediate candidate set of permutations over an alphabet. We accomplished this with MAX, DEP, and the set of self-conjoined constraints $*\alpha^2$. It is important to notice that it is not the initial infinitude of candidates that renders the problem computationally intractable; rather, it is the exponential *growth-rate* in this relevant intermediate candidate space.[2] Thus, any attempt to modify OT in order to render it computationally tractable will have to eliminate (or severely limit) the use of *Gen* or one of these constraint types, presumably the set of self-conjoined constraints. However, such restrictions must still somehow allow their use in the problems that Ito and Mester (2003) address, or invent new mechanisms to deal with effects like those captured by Lyman's Law and Grassmann's Law. But any such new mechanisms must be constructed so as not to be able to be used to construct permutation sets or to model other intractable computational problems.

References

Barton, G. Edward, Jr., Robert C. Berwick, and Eric S. Ristad. 1987. *Computational complexity and natural language*. Cambridge, Mass.: MIT Press.

Cameron, Peter J. 1994. *Combinatorics: Topics, techniques, algorithms*. Cambridge: Cambridge University Press.

Eisner, Jason. 1997. Efficient generation in Primitive Optimality Theory. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics*, 313–320.

Eisner, Jason. 2000. Easy and hard constraint ranking in Optimality Theory: Algorithms and complexity. In *Finite-state phonology: Proceedings of the 5th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*, ed. by Jason Eisner, Lauri Karttunen, and Alain Thériault, 22–33.

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

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

Greenberg, Joseph H. 1950. The patterning of root morphemes in Semitic. *Word* 6:162–181.

Ito, Junko, and Armin Mester. 2003. *Japanese morphophonemics: Markedness and word structure*. Cambridge, Mass.: MIT Press.

[2] A reviewer makes two relevant points. First, in Eisner's proof if the grammar is given beforehand, the winners can be computed in reasonable time. As the reviewer notes, Eisner (1997:319) also offers three appropriate rejoinders to this observation, which are also appropriate here. In the present case, for instance, such a technique will require exponential space for the storage of the compiled grammar, as it will potentially encode all permutations of *I*. Thus, the precompiled grammar simply trades space-complexity for time-complexity and does not solve the inherent computational difficulties. Second, the reviewer suggests a review of ''average-case'' or ''real-world'' performance. Unfortunately, these measures do not lend themselves to the algebraic evaluation of complexity, and computer simulations for this purpose are beyond the scope of the present remarks (though a most useful area for further research).

Kager, René. 1999. *Optimality Theory*. Cambridge: Cambridge University Press.
McCarthy, John J. 2003. OT constraints are categorical. *Phonology* 20:75–138.
Prince, Alan, and Paul Smolensky. 2004. *Optimality Theory: Constraint interaction in generative grammar*. Malden, Mass.: Blackwell.

Department of Linguistics
1401 Marie Mount Hall
University of Maryland
College Park, Maryland 20742-7505

idsardi@umd.edu

In Defense of a Quantificational Account of Definite DPs

Daniela Isac

Two types of arguments support a quantificational view of definite DPs. First, definite DPs share properties with other quantified expressions. In particular, they pattern together in antecedent-contained deletion constructions, they show weak crossover effects, and at least some of them interact scopally with other quantified expressions. Second, the apparent failure of (some) definite DPs to interact scopally with other quantified expressions and to exhibit island effects stems from two properties of definite DPs: they are all principal filters, and the witness set of singular definite DPs is a singleton. These two properties have the effect of rendering the wide and narrow scope readings of definite DPs indistinguishable.

Keywords: quantificational definite DPs, principal filters, singletons

Two types of accounts have been proposed in the literature for the wide scope interpretation of definite DPs: (a) quantificational accounts, which assume that definite DPs undergo Quantifier Raising (QR) at LF to a wide scope position, and (b) referential approaches, which assume that definite DPs behave like proper names and are interpreted in situ, without moving at LF. The former view originated with Russell (1905) and was adopted by Milsark (1974), Barwise and Cooper (1981), May (1985), Chierchia and McConnell-Ginet (1990), and others. The latter view was initially proposed by Frege (1892) and then defended by Strawson (1950), Kaplan (1972), Hornstein (1984), and others.

I am grateful to Charles Reiss, Fred Mailhot, and two anonymous reviewers for helpful comments and suggestions. Remaining errors are my own. This work was supported by Canadian Employment Insurance funds granted to the author as maternity benefits.

Linguistic Inquiry, Volume 37, Number 2, Spring 2006
275–288