A remark on ties in Harmonic Serialism

John McCarthy

12/8/2009

Introduction

These thoughts were stimulated by Kathryn Pruitt's presentation about ties in HS. Ties present problems for the implementation in OT-Help 2, and I have a suggestion about how to address these problems.

Disclaimer

I'm focusing, as Kathryn did, on problematic ties, not intentional ties. This isn't about variation.

Assume

For the input I, there is a set of tied winning candidates $T = \{t_1, t_2, \ldots\}$ ($|T| > 1$).

The difference between I and t_j is precisely described by an operation o_j.[1] The set of all such operations, one for each member of T, is $O = \{o_1, o_2, \ldots\}$.

In practice, the operations in a given O are always of the same type: only deletion, only insertion, etc. No O can contain non-homogeneous operations because different faithfulness constraints would be violated, so the candidates they produce wouldn't be tied. (This comment presupposes minimal competence on the part of the analyst.)

Definitions

(1) *Order-independence*
Given I, T, and O, if $o_1(o_2(\ldots(I))) = o_2(o_1(\ldots(I))) = \ldots$ for all permutations of O, then O or T can be called *order-independent*.

(2) *Tie union*
If O is order-independent, the tie union $t^* = o_1(o_2(\ldots(I)))$. (If O is not order-independent, t* does not exist.)

(3) *Winning tie union*
If $t^* \succ t_j \; \forall t_j \in T$, then t* is a *winning tie union*.

[1] To "precisely characterize" the mapping from I to t_j, o_j has to say more than just "Insert" or "Delete". It has to say what is inserted or deleted, and where. In other words, $o_j(I)$ is a function. It is a localized unfaithful mapping in the sense of *Hidden Generalizations*.

Examples

Codas delete one at a time, but all eventually delete:

(4) NO-CODA >> MAX with input [pak_1pak_2]
   a. T = {$papak_2$, pak_1pa}.
   b. O = {"delete k_1", "delete k_2"}.
   c. O is order-independent because "delete k_1"("delete k_2"(pak_1pak_2)) = "delete k_2"("delete k_1"(pak_1pak_2)).
   d. Tie union t* = [papa].
   e. t* is a winning tie union because [papa] ≻ [papak], [pakpa].

Discussion: Example (4) is a typical convergent tie in HS. O is order-independent, so t* exists. Furthermore, t* is a winning tie union. It would be safe to declare that t* is the output of the evaluation, even though it is not among the candidates that were originally evaluated.

**LAPSE offers several locations for a foot. They are mutually inconsistent:*

(5) *LAPSE >> ALIGN-L(word, foot) with input [$o_1o_2o_3o_4o_5$] (assuming trochees)
   a. T = { $o_1(\text{'}o_2o_3)o_4o_5$, $o_1o_2(\text{'}o_3o_4)o_5$, $o_1o_2o_3(\text{'}o_4o_5)$}
   b. O = {"parse trochee o_2o_3", "parse trochee o_3o_4", "parse trochee o_4o_5"}.
   c. O is not order-independent, since the operations in O make inconsistent demands on the output. E.g.
      "parse trochee o_2o_3"("parse trochee o_3o_4"("parse trochee o_4o_5"($o_1o_2o_3o_4o_5$))) = $[o_1(\text{'}o_2o_3)\text{'}o_4o_5)]$
      ≠
      "parse trochee o_2o_3"("parse trochee o_4o_5"("parse trochee o_3o_4"($o_1o_2o_3o_4o_5$))) = $[o_1o_2(\text{'}o_3o_4)o_5]$
   d. Therefore, the tie union t* does not exist.

Discussion: Example (5) is the type of non-convergent tie that OT-Help currently handles incorrectly. OT-Help could check for the existence of a tie union; if one is not found, it could report the problem and refuse to proceed.

There are two possible ways of satisfying NO-CODA and only one is needed:

(6) NO-CODA >> MAX with input [pat.ki]
   a. T = {pa.ki, pa.ti}.
   b. aO = {"delete t", "delete k"}.
   c. aO is order independent.
   d. Tie union t* = [pa.i].
   e. But t* is not a winning tie union because [pa.ki], [pa.ti] ≻ [pa.i].

Discussion: Example (6) would be a tie in parallel OT too. OT-Help could refuse to proceed, or it could pick a winner at random and also issue a warning. (Or it could follow all of the derivational futures of the tied winners, but that would probably involve a lot of additional coding at this point.)

Questions

Do the statements about these particular examples generalize to all ties? Probably not, since subsequent derivational steps might produce convergence even in cases where there is no (winning) tie union. But maybe we should regard such "successes" as accidental and unstable, so losing sight of them may not be a great loss.

From an implementational perspective, the calculation needed to determine whether O is order-independent is unattractive since it requires time proportional to $|O|!$. Could we get away with a quicker but less accurate calculation, such as looking only at all orders on pairs of elements in O? Or is there a string-based approach to determining whether t* exists? This is only quadratic in $|O|$. We'd then be relying on the locality of operations in GEN to protect us from missing non-convergent interactions involving more than two operations. Another approach is to take a string-based rather than an order-based approach to finding the tie union (if one is to be found).

What if there's a tie union only on some proper subset of T? Is that interesting or useful information? Perhaps, since the analysis would be well-behaved if there were a way of eliminating from the tie those candidates that aren't in that subset. Perhaps this could be part of an informative error message when OT-Help refuses to proceed.

Theory versus implementation

The theory needs to tell us what to do even in situations where OT-Help might refuse to proceed. In classic OT, "tell us what to do" means "find the most harmonic candidate". In P&P, "tell us what to do" means "give an output ('converge') or give no output ('crash')". In HS-the-theory, the derivation forks at ties, and so an output is always provided. In non-convergent cases, more than one output may emerge. In HS-the-implementation, however, it might make more sense to crash in situations that we know to be undesirable.