"Further thoughts on ties in Harmonic Serialism"
John J. McCarthy
December, 2009

Further thoughts on ties

Situation

The input I has 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 $O = \{o_1, o_2, \ldots\}$.

Remark

The operations in O are probably always the same type: only deletion, only insertion, etc.. If they involved violating different faithfulness constraints, the candidates they produce wouldn't be tied. (Of course, this doesn't reckon with incompetent analysis.)

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) is *order-independent.*

(2) *Tie union*
If O is order-independent, the tie union $t^* = o_1(o_2(\ldots(I)))$.

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

Examples

(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].

papa ≻ pakpakpa, pakpapak, papakpak

[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. It is a localized unfaithful mapping in the sense of *Hidden Generalizations.*

(5) *LAPSE > > ALIGNWDL 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.
d. Therefore, the tie union t* does not exist.

(6) NO-CODA > > MAX with input [patki]

a. T = {paki, pati}.
b. aO = {"delete t", "delete k"}.
c. aO is order independent.
d. Tie union t* = [pai].
e. But t* is not a winning tie union because [paki], [pati] ≻ [pai].

Observations about these examples

Example (4) is a typical convergent tie in HS. The members of T are harmonically bounded by the tie union t* and I. It would be safe to declare that t* is the output of the evaluation, even though it is not among the candidates originally evaluated.

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 would report the problem and refuse to proceed.

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 could perhaps in principle produce convergence even in cases where there is no (winning) tie union. But maybe we should regard such "successes" as accidental and unstable, and so rule them out.

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 calculation, such as looking only at all orders on pairs of elements in O? 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.

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 "give an output (or more)". In P&P, "tell us what to do" means "give an output ('converge') or give no output ('crash')". In HS-the-theory, we should probably say that the derivation always forks at ties, and so an output (or more) is always provided. In HS-the-implementation, however, it might make more sense to crash in situations that we know to be undesirable.

Ties and variation

If OT has taught us anything about ties, it has taught us that they are not a very useful model of language variation. To my knowledge, tied evaluation over the entire constraint set plays no role in these models. Variation, then, is a complete red herring.