constraints in syntax-prosody mapping, including constraints from both Match Theory and Align Theory, so that analyses couched in both frameworks can be created and compared directly. In fact, the system created here (Msp.Asp) is but one of several that we compared in Bellik et al. (to appear), a selection of which can be accessed via the Built-in systems menu at the top of the SPOT interface. Msp.Asp is represented there as "Japanese : MatchSP, AlignSP (Bellik, Ito, Kalivoda, & Mester to appear)"; selecting that menu item is a shortcut to bringing up the entire system. Others can also be selected to view alternative analyses that do ( ) or do not ( ) capture the asymmetric Japanese phrasing pattern.

The violation tableaux created by SPOT can be analyzed by hand, but ideally, they should be imported into other software for Optimality Theory analyses, such as OTWorkplace or OTSoft, which can quickly generate the typologies that those systems predict. By combining SPOT with OT software, syntax-prosody analysts can create more theoretically rigorous analyses of phenomena at the syntax-prosody interface.

References

Alber, Birgit, Natalie DelBusso & Alan Prince. 2016. From intensional properties to universal support. 92(2). e88–e116. https://doi.org/10.1353/lan.2016.0029.

Bane, Max & Jason Riggle. 2012. Consequences of candidate omission. 43(4). 695–706. https://doi.org/10.1162/ling_a_00112.

Bellik, Jennifer, Ozan Bellik & Nick Kalivoda. 2015-2021. Syntax-prosody in Optimality Theory (SPOT). Javascript application. http://spot.sites.ucsc.edu. Codebase at https://github.com/syntax-prosody-ot.

Bellik, Jennifer, Junko Ito, Nick Kalivoda & Armin Mester. To appear. Matching and alignment. In Haruo Kubozono, Junko Ito & Armin Mester (eds.), *S* *L* Oxford, UK: Oxford University Press.

Ishihara, Shinichiro. 2014. Match theory and the recursivity problem. In Shigeto Kawahara & Mika Igarashi (eds.), *"#$ %& " ' $ $ /* *0 , 7 : RUNS L QJ 3 DS HUV L Q / L QJ X L V WL F V* 69–88.

Ito, Junko & Armin Mester. 2003/1992. Weak layering and word binarity. In Takeru Honma, Masao Okazaki, Toshiyuki Tabata and Shin-ichi Tanaka (eds.), *# Q0 F* *S S W #"* *2 , 3* *R K V E* 26–65. Tokyo: Kaitakusha.

Ito, Junko & Armin Mester. 2013. Prosodic subcategories in Japanese. 124. 20–40. https://doi.org/10.1016/j.lingua.2012.08.016.

Karttunen, Lauri. 2006. The insufficiency of paper-and-pencil linguistics: The case of Finnish prosody. In Miriam Butt, Mary Dalrymple & Tracy Holloway King (eds.), *& 6 ' 7) ! 8*, 287–300. Stanford, CA: CSLI Publications.

Kubozono, Haruo. 1989. Syntactic and rhythmic effects on downstep in Japanese. Phonology 30(1). 39–67.

!" # $ % ! https://doi.org/10.1515/9783110977790.

Prince, Alan. 2017. What is OT? Slides. http://roa.rutgers.edu/content/article/files/1513_prince_4.pdf.

Prince, Alan. 2017. OT Checklist. http://roa.rutgers.edu/content/article/files/1615_prince_2.pdf.

Prince, Alan, Nazarré Merchant & Bruce Tesar. 2020. OTWorkplace.

https://sites.google.com/site/otworkplace/.
Selkirk, Elisabeth. 1978. On prosodic structure and its relation to syntactic structure. In *Nordic prosody II* ed. Fretheim T, 111–40. Trondheim: TAPIR .
Selkirk, Elisabeth. 1980. Prosodic domains in phonology: Sanskrit revisited. In Mark Aronoff & M.-L. Kean (eds.), *Juncture* (vol. 7), 107–29. Saratoga, CA: Anma Libri.
Selkirk, Elisabeth. 1984. *Phonology and syntax: The relation between sound and structure*. Cambridge, MA: MIT Press.

Appendix: Other inputs in Msp.Asp. Msp.Asp as defined above has 7 inputs, those in the 3w and 4w columns. Logically, we could also expand its input set to include analogous trees with one and two terminals, those in the 1w and 2w columns below.[7]

1w	2w	3w	4w
[a]	[a b]	[a [b c]] [[a b] c]	[a [b [c d]]] [a [[b c] d]] [[a b] [c d]] [[a [b c]] d] [[[a b] c] d]

Table 3. Extended inputs of Msp.Asp

We have calculated Msp.Asp with all of the inputs, and its FacTyp contains the 14 languages we just encountered with only three inputs (4wR, 4wM, 4wL). By removing inputs, we show that [a [b [c d]]], [[[a b] c] d], and one of either [a [[b c] d]] or [[a [b c]] d] are a universal support for Msp.Asp. As for the other inputs, the reasons they are redundant are explained in Table 4.

Input	Comment
[a]	Msp.Asp.GEN admits only one candidate for this cset: [a]→(a)
[a b]	Only 1 candidate in the cset is an optimum: [a b]→(a b)
[a [b c]]	Only 1 candidate in the cset is an optimum: [a [b c]]→(a (b c))
[[a b] c]	Only 1 candidate in the cset is an optimum: [[a b] c]→((a b) c)
[[a b] [c d]]	Only 1 candidate in the cset is an optimum: [[a b] [c d]]→((a b) (c d))
[[a [b c]] d]	In every language, this input behaves the same way as [a [[b c] d]]. *Matching case*: If [a [[b c] d]]→(a ((b c) d)) is optimal, [[a [b c]] d]→((a (b c)) d) is optimal. *Squishing case*: If [a [[b c] d]]→(a (b c) d) is optimal, [[a [b c]] d]→(a (b c) d) is optimal. *Rebracketing case*: If [a [[b c] d]]→((a b) (c d)) is optimal, [[a [b c]] d]→((a b) (c d)) is optimal.

Table 4. Redundant inputs

[7] The number of inputs on n words is Catalan number n–1. The nth Catalan number is (2n)!/(n+1)!n! for n≥0, and 1 for n=0 .