In: Haruo Kubozono, Junko Ito, and Armin Mester, eds. Prosody and Prosodic Interfaces. Oxford Studies in Phonology and Phonetics. https://global.oup.com/academic/product/prosody-and-prosodic-interfaces-9780198869740?lang=en&cc=gb# ISBN: 9780198869740. Published: 12 May 2022.

15
Matching and alignment

Jennifer Bellik, Junko Ito, Nick Kalivoda, and Armin Mester

15.1 Introduction

Phonological phrases tend to mirror syntactic phrases, but sometimes there are mismatches. For example, in a well-known mismatch in Japanese first pointed out by Kubozono (1989), a uniformly left-branching four-word syntactic phrase is rebracketed into a symmetrically branching phonological phrase in (1).

(1) $[_{XP}\ [_{XP}\ [_{XP}\ A\ B]\ C]\ D]$ → $(_{\varphi}\ (_{\varphi}\ A\ B)\ (_{\varphi}\ C\ D)\ )$

We are assuming here the theory of recursive phrasing argued for in Ito and Mester (2012), where the phonological phrase (φ) is recursively deployed, and the minimal φ is the domain of accent cumulativity, and any φ the domain of initial rise and downstep. The recursive theory is superior to traditional models with binary category distinctions: minor phrase vs. major phrase in McCawley (1968) and Kubozono (1989), among others, or the roughly equivalent intermediate phrase vs. accentual phrase distinction in Pierrehumbert and Beckman (1988). Ito and Mester (2012: 289–94) show that the nonrecursive models provide both too much and too little structure to account for the empirical facts in Japanese.

The rebracketing in (1) has been the subject of several Optimality-Theoretic analyses (Ishihara 2014; Ito and Mester 2013; Kalivoda 2018; Selkirk 2011). What has been less discussed and analyzed is the fact that other three- and four-word syntactic structures do not show phonological rebracketing (Kubozono 1989; Shinya et al. 2004), so that, e.g., a uniformly right-branching four-word syntactic phrase is mapped to a matching prosodic structure in (2).

(2) $[_{XP}\ A\ [_{XP}\ B\ [_{XP}\ C\ D]]]$ → $(_{\varphi}\ A\ (_{\varphi}\ B\ (_{\varphi}\ C\ D)\ )\ )$

A general feature of previous studies is that they have often not considered the full range of possible prosodic parses, owing to the impossibility of manually generating and evaluating numerous tree structures. In this chapter, we reanalyze the Japanese pattern of prosodic matches and mismatches for all three- and four-word syntactic inputs, taking into account all possible prosodic parses of each

Jennifer Bellik, Junko Ito, Nick Kalivoda, and Armin Mester, *Matching and alignment*. In: *Prosody and Prosodic Interfaces*. Edited by Haruo Kubozono, Junko Ito, and Armin Mester, Oxford University Press.
 DOI: 10.1093/oso/9780198869740.003.0015

string of terminals. This was made possible through automatic candidate generation and evaluation by the SPOT App, developed as part of the larger SPOT project (Syntax-Prosody in Optimality Theory, https://spot.sites.ucsc.edu/).

SPOT is a collaborative research project in the Linguistics Department at the University of California, Santa Cruz, funded by National Science Foundation Grant #1749368, which aims to develop new tools for rigorously investigating the mapping from syntactic to prosodic structure. The group is developing the SPOT App, an application which performs automatic candidate generation and constraint evaluation over prosodic parses for work in Optimality Theory, in particular, Match Theory and Alignment Theory. This open-source application is accessible through the project website, and can also be downloaded from Github (https://github.com/syntax-prosody-ot).

Here, we present a new discovery concerning how to analyze the above well-known syntax–prosody mismatch in Japanese using Optimality Theoretic constraints—namely, that when the full set of syntactic inputs and possible prosodic outputs is considered, Match Theory (Selkirk 2011) and Align Theory (Selkirk 1986, 1996; Truckenbrodt 1999) are each insufficient, and must be combined in order to capture the Japanese pattern. In a system containing Match Theory's symmetrical constraints as the only syntax–prosody mapping constraints, there is no way to distinguish between the uniformly left-branching syntax that needs to be rebracketed in the prosodic output, and the uniformly right-branching syntax that must be matched in the prosodic output. Edge-based Align constraints can successfully distinguish those two syntactic inputs, so adding them solves the problem. However, the Align constraints cannot correctly predict all the Japanese syntax–prosody mappings when they are the only mapping constraints in the system. Align constraints encounter a problem when mixed branching syntactic trees such as [A[[BC]D]] are considered; they cannot favor the correct matching prosody (A((BC)D)) over the incorrectly rebracketed ((A(BC))D). When both Match and Align constraints are included in the system, all of the correct mappings are selected as optimal. Thus, Japanese phonological phrasing shows us that Match constraints cannot entirely supplant Align constraints, nor can Align constraints do all the work of Match constraints. This point is of some theoretical importance because Match Theory has often been portrayed (Selkirk 2011; Elfner 2012, etc.) as fully supplanting its predecessor Alignment Theory in syntax–prosody mapping.

15.2 Japanese phrasing

In this chapter, we adopt a view of phonology on which a syntactic tree structure is mapped onto a prosodic tree composed of purely prosodic elements. The latter tree is the domain of phonological processes, which have no direct access to the syntax.

That is, we assume the indirect reference approach of Selkirk (1986), Nespor and Vogel (1986), and others. One significant consequence of this approach is that it allows for *mismatches* between syntactic and prosodic structure; as independent entities, the constituent structures of the syntactic and prosodic structures need not coincide perfectly. The syntax-to-prosody mapping in (1)—the focus of this chapter—is such a case.

According to the prosodic hierarchy theory adopted here, while syntactic structures are composed of heads (X^0) and phrases (XP), prosodic trees are made of distinct prosodic constituents: prosodic word (ω), phonological phrase (φ), and intonational phrase (ι). The grammar of the syntax-phonology interface includes constraints responsible for mapping an input syntactic structure onto an output prosodic structure.

Kubozono (1988, 1989, 1992, 1993) presents phonetic data showing that phonological phrasing in Tokyo Japanese is sensitive to syntactic structure. The beginning of a phonological phrase in Tokyo Japanese is marked with a pitch rise. The branching XPs in the syntactic structures on the left in (3) are produced with pitch rises that are aligned with their left edges "$(_\varphi$," indicating that they are matched by corresponding phonological phrases. For example, the left-branching three-member structure in (3a) is produced with a pitch rise only on the first word *naomi-no*, since this is the only word that is at the left edge of a syntactic phrase (two syntactic phrases, in fact). On the other hand, in the right-branching three-member structure (3b), there are rises on the first and second words, *naomi-no* and *marui*, which both occur at the left edge of syntactic phrases.[1]

(3) a. [[naomi-no ane-no] yunomi] → $(_\varphi$ $(_\varphi$ naomi-no ane-no) yunomi)
Naomi-GEN sister-GEN teacup
'Naomi's sister's teacup'

b. [naomi-no [marui yunomi]] → $(_\varphi$ naomi-no $(_\varphi$ marui yunomi))
Naomi-GEN round teacup
'Naomi's round teacup'

For presentational purposes, the representations here and below exclude one-word syntactic phrases, which, when unaccented, are not mapped to phonological phrases because BINARITY dominates MATCH-XP (Ito and Mester 2013: 29). This is equivalent to saying that a non-branching XP with only one phonologically overt terminal X^0 is invisible to the phonology, whose attention will be focused on X^0.[2]

[1] Since there are cues in Japanese only for left edges of phonological phrases, not for right edges, an alternative structure is (A B C) with no internal bracketing (see Selkirk 2011 for discussion). Match Theory resolves this indeterminacy in a general way, by always giving priority to exactly matching structures—here, ((A B) C)—ceteris paribus, i.e., as long as this is not forestalled by dominant and conflicting markedness constraints.

[2] In this, our analysis resembles Bennett, Elfner, and McCloskey (2016), who state that when two syntactic projections of different categories dominate the same set of terminals, as in a structure like $[_{XP}\ X^0]$, the phonology will not produce a non-branching structure like $(_\varphi\ (_\omega\ X^0))$.

When we turn to four-member structures, we encounter the contrast mentioned in the introduction. A strictly right-branching four-member structure gets faithfully mapped onto an isomorphic right-branching prosodic structure (4a), but a strictly left-branching structure gets rebracketed into a balanced prosodic tree (4b).

(4) a. [naomi-no [marui [omoi yunomi]]] → (φ naomi-no (φ marui (φ omoi yunomi)))
Naomi-GEN round heavy teacup
'Naomi's round heavy teacup'

b. [[[naomi-no ane-no] yunomi-no] iro] → (φ (φ naomi-no ane-no) (φ yunomi-no iro))
Naomi-GEN sister-GEN teacup-GEN color
'the color of the teacup of the sister of Naomi'

The remaining four-member configurations in (5) shows that they are faithfully mapped into isomorphic prosodic structures.

(5) a. [[naomi-no ane-no] [marui yunomi]] → (φ (φ naomi-no ane-no) (φ marui yunomi))
Naomi-GEN sister-GEN round teacup
'Naomi's sister's round teacup'

b. [[naomi-no [ue-no ane-no]] yunomi] → (φ (φ naomi-no (φ ue-no ane-no)) yunomi)
Naomi-GEN upper-GEN sister-GEN teacup
'Naomi's eldest sister's teacup'

c. [naomi-no [[ume-no iro-no] yunomi]] → (φ naomi-no (φ (φ ume-no iro-no) yunomi))
Naomi-GEN plum-GEN color-GEN teacup
'Naomi's plum-colored teacup'

All noninitial rises in (3)–(5) are downstepped (see Kubozono 1988, etc.), indicating that there is some domain that spans the entire phrase.

The examples above are all lexically unaccented words. Because of high-ranking ACCENT-AS-HEAD (see Ito and Mester 2013), lexically accented words are individually mapped to minimal phonological phrases, as shown in (6).

(6) a. [náoko-no [nagái [aói erímaki]]] → (φ (φ náoko-no) (φ (φ nagái) (φ (φ aói) (φ erímaki))))
Naoko-GEN long blue muffler
'Naoko's long blue muffler'

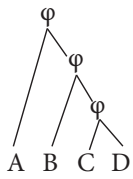
b. [[[náoko-no áni-no] erímaki-no] iromóyoo]
Naoko-GEN brother-GEN muffler-GEN color pattern
→ (φ (φ (φ náoko-no) (φ áni-no)) (φ (φ erímaki-no)(φ iromóyoo)))
'The color pattern of the muffler of the brother of Naoko'

As the hierarchical tree structures in (7) clearly illustrate, there is an extra (minimal) level of φ-phrasing for accented word combinations (7b). However, the nonminimal phrasings of the accented word combinations (7b) are identical to the

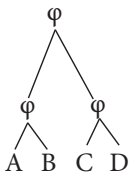
unaccented word combinations (7a), including the difference between the faithful right-branching structure and the rebracketed left-branching structure. This also holds for cases with a combination of accented and unaccented words.

(7) a. Unaccented ω's

=(4a)

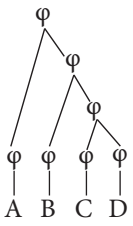


=(4b)

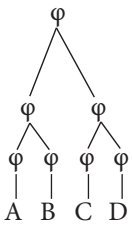


b. Accented ω's

=(6a)



=(6b)



An important additional factor in accented right-branching structures such as (6b) is Kubozono's metrical boost (Kubozono 1989: 53–8) which "raises pitch contours whenever right branching is involved," here the beginning of the second and third words. In our terms, the locus of metrical boost is the beginning of two φ's.

Concerning the syntax–prosody mappings depicted in (3)–(6), we can see that every XP's left edge is mapped to the left edge of a phonological phrase (φ) in the prosody. Furthermore, every φ in (3), (4a), (5), and (6a) corresponds to an XP in the syntax. However, when the uniformly left-branching syntax is pronounced, the XP *naomi-no ane-no yunomi-no* in (4b) or *náoko-no áni-no erímaki-no* in (6b) is not mapped to a φ. Instead, an unexpected rise occurs on *yunomi-no* and a larger-than-expected rise (i.e., Kubozono's metrical boost) occurs on *erímaki-no*, indicating that the output of the phonology includes a φ that has no XP-correspondent. Our goal in the remainder of this chapter is to use Optimality Theory to capture this match/mismatch pattern, in which the uniformly left-branching syntax is rebracketed into a balanced prosodic tree, while all other input syntactic structures receive a matching prosodic parse. We will demonstrate that key syntax–prosody mapping constraints from both Match Theory and Alignment Theory are necessary to describe the specific mapping between syntax and prosody observed in Japanese.

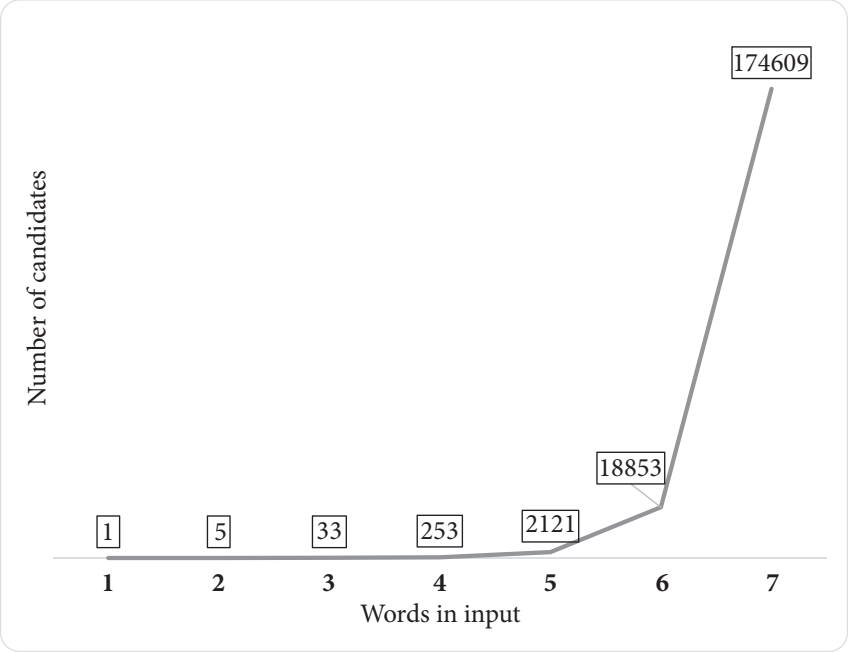
15.3 Considering all candidates

For a candidate to be *optimal* under a ranking in OT, it must be better than all of its competitors, which are defined by the generator function GEN (Prince and Smolensky 2004). If a candidate admitted by GEN is omitted from consideration, serious errors can occur (Bane and Riggle 2012). In the worst case, the allegedly optimal candidate may prove to be harmonically bounded (suboptimal under every possible ranking) when a forgotten candidate is included. It is therefore crucial

to know the full set of candidates admitted by GEN, and how they perform on every constraint.

In syntax–prosody mapping, the full candidate sets admitted by GEN are often quite large, making it necessary to automate their generation. We do so using the SPOT application. Suppose that a prosodic tree consists of an intonational phrase ι (the root node), one or more phonological phrases (the non-terminal nodes, possibly recursive), and a series of prosodic words ω (the terminal string). Even assuming that words are never reordered, inserted, or deleted, the number of parses for an input containing *n* words rises rapidly as *n* increases, as shown in (8).

(8) Number of candidates under SPOT's GEN_{EX}[3]



An example of candidate omission in the study of Japanese phrasing comes from Selkirk (2011), who attributes the rebracketing in (1) to a ranking of three constraints: BINMAX(φ,ω) ≫ MATCH(XP,φ), MATCH(φ,XP), defined here as shown in (9).[4]

[3] We subscript GEN here with "EX" for "exhaustivity" because this function does not allow prosodic level-skipping. When nonexhaustive parsing is allowed, the number of parses is even greater. SPOT includes GEN_{EX} and nonexhaustive GEN, among other versions.

[4] Throughout this section, the input contains a one-word XP [A], as in Ito and Mester (2013) and Ishihara (2014). It thereby differs from Selkirk's (2011: 469) representation in her example (37), and our representation in (1) and in the following sections, non-branching syntactic phrases are abstracted away from altogether. The presence of [A] does not affect the structure of the argument, or its implications for later sections.

(9) a. BINMAX(φ,ω): Assign one violation for every node of category φ in the prosodic tree that dominates more than two nodes of category ω.
b. MATCH(XP,φ): Assign one violation for every node of category XP in the syntactic tree such that there is no node of category φ in the prosodic tree that dominates all and only the same terminal nodes as XP.
c. MATCH(φ,XP): Assign one violation for every node of category φ in the prosodic tree such that there is no node of category XP in the syntactic tree that dominates all and only the same terminal nodes as φ.

In this and all following tableaux, we partially adopt the comparative tableau format of Prince (2002). The optimal output is shown in row (10a), indicated by "→". Nonoptimal outputs are shown in subsequent rows. Violation counts for constraints are indicated for the optimum with integers. For nonoptimal candidates, the violation count is shown in subscript after a "W", "L", or "e". A "W" in a cell means that the constraint in that column favors the intended winner over the intended loser; an "L" means that the constraint favors the intended loser over the intended winner; and an "e" means that the constraint penalizes the winner and the loser equally. Filled with W's, L's, and e's, each loser row constitutes an *Elementary Ranking Condition* (ERC). All columns are separated by solid lines, which should not be taken to indicate ranking. The ranking (which is not always a total order on the constraint set) can be read off the comparative tableaux in the following way: in each loser row (ERC), there must be some W that precedes every L. This is because for each winner-loser pair, there must be some constraint favoring the winner that dominates every constraint favoring the loser. For example, row (10b) contains W for BINMAX(φ,ω), L for MATCH(XP,φ), and L for MATCH(φ,XP), indicating that BINMAX(φ,ω) favors the winner (10a) over the loser (10b), while the two MATCH constraints favor the loser (10b) over the winner (10a). From this row, we conclude that BINMAX(φ,ω) dominates both MATCH constraints. From this winner-loser pair, we cannot conclude anything about the relative ranking of MATCH(XP,φ) and MATCH(φ,XP). For an extensive explanation of comparative tableaux and Elementary Ranking Conditions, we refer the reader to Prince (2002).

(10) Selkirk's (2011) analysis of Japanese rebracketing

	[[[[A] B] C] D]	BINMAX(φ,ω)	MATCH(XP,φ)	MATCH(φ,XP)
a. →	((A B) (C D))	1	2	1
b.	((((A) B) C) D)	W_2	L_0	L_0

With just candidates (10a, b), the analysis is correct; when BINMAX(φ,ω) dominates both MATCH constraints, the rebracketing candidate beats the matching candidate. However, Ishihara (2014) discovered that the single additional candidate (11c) admitted by GEN leads to harmonic bounding of the desired winner.

While neither (11b) nor (11c) alone individually bounds (11a), the two collectively bound it.

(11) The Recursivity Problem (Ishihara 2014)

	[[[[A] B] C] D]	BinMax(φ,ω)	Match(XP,φ)	Match(φ,XP)
a. →	((A B) (C D))	1	2	1
b.	((((A) B) C) D)	W_2	L_0	L_0
c.	(A B) (C D)	L_0	W_3	e_1

Candidates (11a) and (11c) differ only in that the former wraps the two minimal phrases (AB) and (CD) in a maximal phrase, while the latter does not, which means that (CD) is not downstepped with respect to (AB) (see Ishihara 2014). With the candidate and constraint sets in (11), the Japanese optimum (11a) is harmonically bounded. Although BinMax(φ,ω) favors Japanese (11a) over matching (11b), and Match(XP,φ) favors Japanese (11a) over nonrecursive (11c), there is no ranking of these constraints under which (11a) is optimal. When one of the Match constraints dominates BinMax, the perfectly matching (11b) is optimal, and when BinMax dominates both of the Match constraints, the nonrecursive (11c) is optimal. Returning to the notion of the ERC, we note that no arrangement of the constraint columns in (11) results in there being a W before every L in both rows (11b) and (11c), indicating that no ranking of these constraints produces the desired optima.

For (11a) to win out over nonrecursive (11c) in Japanese, an additional constraint is needed. Ishihara (2014) proposes a new Match constraint, Match(XP_{max},φ_{max}), also called MatchMax, which is violated when a maximal lexical XP in the input does not have a matching maximal φ in the output. In the case at hand, MatchMax is violated when the maximal XP [ABCD] does not have a matching maximal φ (ABCD). Adding MatchMax to the constraint set, and keeping the candidate set the same as in (11), the Japanese candidate is no longer harmonically bounded.

(12) Solution to the Recursivity Problem (Ishihara 2014)

	[[[[A] B] C] D]	MatchMax	BinMax(φ,ω)	Match(XP,φ)	Match(φ,XP)
a. →	((A B) (C D))	0	1	2	1
b.	((((A) B) C) D)	e	W_2	L_0	L_0
c.	(A B) (C D)	W_1	L_0	W_3	e_1

Under the ranking MatchMax ≫ BinMax ≫ Match-XP, Match-φ, the Japanese candidate (12a) is optimal, given this candidate set. It would appear, therefore, that Ishihara (2014) has solved the candidate-omission problem in Selkirk (2011).

However, there are still further additional candidates that must be considered. Using SPOT to generate all possible candidates admitted by Gen, Kalivoda (2018) discovered that MatchMax does not save the Japanese candidate from harmonic bounding when the full set is considered. A single additional candidate, (13d), which Kalivoda (2018) calls the *squishing* candidate, harmonically bounds the desired (13a).

(13) The Squishing Problem (Kalivoda 2018)

	[[[[A] B] C] D]	MatchMax	BinMax(φ,ω)	Match(XP,φ)	Match(φ,XP)
a. →	((A B) (C D))	0	1	2	1
b.	((((A) B) C) D)	e	W_2	L_0	L_0
c.	(A B) (C D)	W_1	L_0	W_3	e_1
d.	((A B) C D)	e	e_1	e_2	L_0

Japanese (13a) and the squishing candidate (13d) differ only in that while the former includes a φ (CD), the latter does not, instead placing these words directly under the maximal φ. They tie on MatchMax, BinMax, and Match-XP, but Match-φ favors (13d) over (13a), since there is no XP [CD] in the input. So with the full candidate set admitted by Gen, it is necessary to either add another constraint to Con that favors (13a) over (13d), or to pursue another constraint set entirely.

Kalivoda (2018) shows that adding one more constraint solves this problem. Since (14a) is perfectly binary-branching, while in (14d) the maximal φ is ternary-branching, the constraint BinMax(φ,branches) does the trick. Unlike BinMax(φ,ω), which is violated by every φ containing more than two prosodic words, BinMax(φ,branches) is violated by every φ that immediately dominates more than two nodes (see Elfner 2012 for an argument that BinMax(φ,branches) is active in Irish).

(14) Solution to the Squishing Problem (Kalivoda 2018)

	[[[[A] B] C] D]	BinMax (φ,branches)	Match Max	BinMax (φ,ω)	Match (XP,φ)	Match (φ,XP)
a. →	((A B) (C D))	0	0	1	2	1
b.	((((A) B) C) D)	e	e	W_2	L_0	L_0
c.	(A B) (C D)	e	W_1	L_0	W_3	e_1
d.	((A B) C D)	W_1	e	e_1	e_2	L_0

Finally, in the full candidate set we also find a candidate just like Japanese [[[[A]B]C]D] → ((AB)(CD)), but where the XP [A] has a matching φ: [[[[A]B]C]D] → (((A)B)(CD)). This candidate outperforms the desired winner on MATCH(XP,φ), incurring one violation for failing to match the XP [ABC], while the Japanese candidate fails to match both [ABC] and [A]. Thus, as a final amendment to CON, we need to include BINMIN(φ,ω), a constraint penalizing one-word phrases (see Selkirk 2011, among others).[5]

(15) Solution to the Unary Problem

	[[[[A] B] C] D]	BINMIN (φ,ω)	BINMAX (φ,branches)	MATCH MAX	BINMAX (φ,ω)	MATCH (XP,φ)	MATCH (φ,XP)
a. →	((A B) (C D))	0	0	0	1	2	1
b.	((((A) B) C) D)	W_1	e_0	e_0	W_2	L_0	L_0
c.	(A B) (C D)	e_0	e_0	W_1	L_0	W_3	e_1
d.	((A B) C D)	e_0	W_1	e_0	e_1	e_2	L_0
e.	(((A) B) (C D))	W_1	e_0	e_0	e_1	L_1	e_1

The above tableau shows that for (15a) to beat (15e), BINMIN(φ,ω) must dominate MATCH(XP,φ). This ranking blocks the faithful mapping of XP [A] to φ (A). The need for BINMIN(φ,ω) was not lost on Selkirk (2011); Ishihara (2014); or Kalivoda (2018), but neither it nor candidate (15e) were included in Selkirk and Ishihara's original discussions of this topic.

To sum up, Selkirk (2011) proposed that Japanese ranks BINMAX(φ,ω) above MATCH(XP,φ) and MATCH(φ,XP), but Ishihara (2014) showed the insufficiency of this analysis by adding a neglected nonrecursive candidate. With an expanded candidate set, Ishihara proposed retaining Selkirk's ranking, by adding a new constraint MATCHMAX, which in Japanese must dominate BINMAX(φ,ω). Kalivoda (2018), considering the full set of 253 four-word candidates generated using SPOT's GEN_{EX}, found that yet another neglected candidate, the squishing candidate, harmonically bounds the desired winner in Ishihara's system. Kalivoda showed that the analysis could be rescued by ranking another constraint, BINMAX(φ,branches), above MATCH(φ,XP) in Japanese. Kalivoda (2018) also noted the need for BINMIN(φ,ω) (or BINMIN(φ,branches)) to solve the Unary Problem left implicit in previous work.

This recap of the progression from Selkirk (2011) to Ishihara (2014) to Kalivoda (2018) serves two purposes here. First, it underscores the importance of generating the full candidate set so that no such problems go unnoticed. This consideration alone is enough to justify using the SPOT application to check syntax–prosody analyses, as we do for every analysis presented in the remainder of the chapter.

[5] Since the terminal nodes of every prosodic output under consideration are prosodic words, and no φ has another φ as its sole child, BINMIN(φ,ω) is in fact equivalent to BINMIN(φ,branches). The choice here is therefore arbitrary.

Second, the analysis of rebracketing in Japanese left-branching syntactic structures shows what constraints we will need to employ in our analysis of the broader range of syntactic structures in (3)–(5). These include both versions of maximal binarity, BinMax(φ,branches) and BinMax(φ,ω), as well as Ishihara's MatchMax. For expository purposes, we will in fact build MatchMax into Gen as an inviolable condition on output representations, allowing us to exclude it from Con. We will also simplify by ignoring unary branching XPs in the syntactic inputs (see footnote 2), but do not remove BinMin(φ,ω) from Con.

Having established the importance of considering all candidates for strictly left-branching cases, we now turn to an analysis of all other syntactic structures containing three and four words.

15.4 Match plus Align

We now present an OT system called S.MspAsp[6] (named for its constraints "Match syntax → prosody" and "Align syntax → prosody") that accounts for the following mappings, which are schematized versions of the data in Section 15.2.[7]

(16) Syntax–prosody mappings in Japanese

a. Three words

i. [[A B] C] → ((A B) C)

ii. [A [B C]] → (A (B C))

b. Four words

i. [[[A B] C] D] → ((A B) (C D))

ii. [[A [B C]] D] → ((A (B C)) D)

iii. [[A B] [C D]] → ((A B) (C D))

iv. [A [[B C] D]] → (A ((B C) D))

v. [A [B [C D]]] → (A (B (C D)))

An OT system S is a pair ⟨Gen.S, Con.S⟩ (Prince and Smolensky 2004; Alber et al. 2016; Merchant and Prince 2021). Gen.S defines the candidate sets (csets) of S, composed of input-output pairs, and Con.S is the constraint set of S. We define S.MspAsp as follows:

[6] The system S.MspAsp is available on the SPOT interface. It is labeled "Japanese ✓: Match SP, Align SP" in the dropdown menu of the section "Built-in systems."

[7] A violation tableau and factorial typology for this and every other system mentioned in the chapter is provided in the supplementary OTWorkplace file hosted at <https://spot.sites.ucsc.edu/oup-supplementary-material-bellik-ito-kalivoda-and-mester/>

(17) The system S.MspAsp [S(ystem).M(atch) s(yntax)p(rosody)A(lign)s(yntax) p(rosody)]

a. GEN.MspAsp[8]

i. *Inputs*: Syntactic trees with three or four terminal nodes, where every non-terminal node is a binary-branching XP and every terminal node is an X^0, as shown in (16).

ii. *Outputs*: For a syntactic input S, every prosodic tree P with non-terminal nodes of category φ and terminal nodes of category ω, such that the terminal nodes in S stand in a one-to-one correspondence relation with the terminal nodes in P, with linear order preserved.

b. CON.MspAsp

i. Syntax-to-prosody mapping constraints

- MATCH(XP,φ): Assign one violation for every node of category XP in the syntactic tree such that there is no node of category φ in the prosodic tree that dominates all and only the same terminal nodes as XP.
- ALIGNL(XP,φ): Assign one violation for every node of category XP in the syntactic tree whose left edge is not aligned with the left edge of a node of category φ in the prosodic tree.
- ALIGNR(XP,φ): Assign one violation for every node of category XP in the syntactic tree whose right edge is not aligned with the right edge of a node of category φ in the prosodic tree.

ii. Markedness constraints on surface prosody

- BINMIN(φ,ω): Assign one violation for every node of category φ in the prosodic tree that contains fewer than two nodes of category ω.
- BINMAX(φ,ω): Assign one violation for every node of category φ in the prosodic tree that dominates more than two nodes of category ω.
- BINMAX(φ,branches): Assign one violation for every node of category φ in the prosodic tree that has more than two children.

[8] The output generator from (17a.ii) is SPOT's $\text{GEN}_{\text{RECROOT}}$, which is stricter than GEN_{EX}. Both versions of GEN enforce EXHAUSTIVITY, but while GEN_{EX} allows multiple maximal φ's to be dominated by the root ι, $\text{GEN}_{\text{RECROOT}}$ labels the root of every candidate with the same category as the intermediate nodes—here a φ, rather than an ι. This is essentially equivalent to maintaining an ι root but requiring every prosodic tree to contain a single maximal φ immediately dominated by the root. Although there are fewer candidates than in GEN_{EX}, the candidate set still grows as a function of the number of terminals in the input: 1, 4, 24, 176, 1440, 12608, 115584, …

Ishihara's (2014) MATCHMAX is not included in (17b), because GEN.MspAsp is defined such that MATCHMAX is never violated by any candidate. As discussed in Section 15.3, MATCHMAX is crucial when it is not built into GEN.[9]

With S.MspAsp defined as in (17), we can now examine its factorial typology. We have calculated it by generating candidates and evaluating constraints using SPOT, and importing the resulting violation tableau into OTWorkplace (Prince et al. 2020), which calculated the typology. OTWorkplace is an Excel-based software that takes OT tableaux and computes rankings and factorial typologies.[10]Although there are 6! = 720 linear orders of the constraints in (17b), the typology contains only fourteen languages. This is possible because languages are defined extensionally as sets of optimal candidates (input-output mappings), and multiple total orderings of the constraint set give rise to identical sets of optima (Alber et al. 2016).

Certain candidate sets in S.MspAsp contain only one possible optimum, i.e., a single candidate that wins under every ranking. These are the three-word [A[BC]] and [[AB]C] as well as the four-word [[AB][CD]]. These three candidates are always mapped to the perfectly isomorphic (A(BC)), ((AB)C) and ((AB)(CD)), respectively.[11] The following table presents the factorial typology of S.MspAsp with these three candidate sets omitted for space reasons.

(18) Factorial typology of S.MspAsp

	[[[A B] C] D]	**[[A [B C]] D]**	**[A [[B C] D]]**	**[A [B [C D]]]**
L.1	((A B) (C D))	((A B) (C D))	((A B) (C D))	((A B) (C D))
L.2	((A B) ((C) D))	((A B) ((C) D))	((A B) ((C) D))	((A B) (C D))
L.3	(((A B) C) D)	((A (B C)) D)	(A ((B C) D))	((A B) (C D))
L.4	(((A B) C) D)	(A (B C) D)	(A (B C) D)	((A B) (C D))
L.5	((A B) (C D))	(A (B C) D)	(A (B C) D)	((A B) (C D))
L.6	((A B) ((C) D))	(A (B C) D)	(A (B C) D)	((A B) (C D))
L.7	((A B) (C D))	((A (B)) (C D))	((A (B)) (C D))	((A (B)) (C D))
L.8	((A B) ((C) D))	((A (B)) ((C) D))	((A (B)) ((C) D))	((A (B)) (C D))
L.9	((A B) (C D))	(A (B C) D)	(A (B C) D)	((A (B)) (C D))
L.10	((A B) ((C) D))	(A (B C) D)	(A (B C) D)	((A (B)) (C D))
L.11	(((A B) C) D)	((A (B C)) D)	(A ((B C) D))	(A (B (C D)))
L.12	**((A B) (C D))**	**((A (B C)) D)**	**(A ((B C) D))**	**(A (B (C D)))**
L.13	(((A B) C) D)	(A (B C) D)	(A (B C) D)	(A (B (C D)))
L.14	((A B) (C D))	(A (B C) D)	(A (B C) D)	(A (B (C D)))

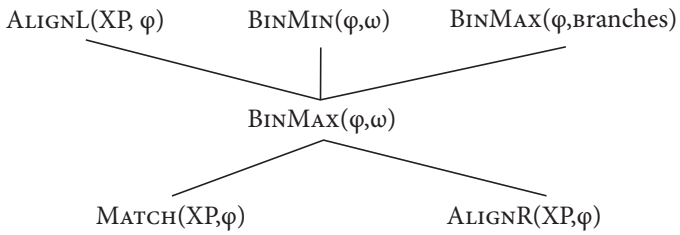
[9] MATCH(φ,XP), included in Section 15.3, is also excluded from CON.MspAsp. A related system exactly like s.MspAsp but with MATCH(φ,XP) added to CON, shown in (29) in Section 15.7, results in a factorial typology with six more languages than in the factorial typology of S.MspAsp. Both typologies contain a language with the Japanese pattern laid out in (16).

[10] It can also assign constraint violations in a tableau, but we carried out this function with SPOT before importing the resulting violation tableau into OTWorkplace.

[11] Whether these inputs should map to the same outputs in every natural language is a question we leave for future research. While these mappings are universal within S.MspAsp, they do not hold in closely related systems with other commonly assumed constraints and representational conditions.

It goes beyond the scope of this chapter to comment on the properties of all these grammars. Language 12 in (18) shows the same mappings as Japanese, laid out in (16). It matches everything but the strictly left-branching [[[AB]C]D], which it maps to ((AB)(CD)). The grammar of this language is given in the following Hasse diagram.

(19)



All of the information in (19) can be ascertained from examining the *Support* for L.12, provided by OTWorkplace. A support is a comparative tableau (Prince 2002) that contains ERCs specifying all domination relationships among constraints in a given grammar.

(20) Support for L.12

	Input	Winner ~ Loser	AlignL (XP,φ)	BinMin (φ,ω)	BinMax (φ,branches)	BinMax (φ,ω)	Match (XP,φ)	AlignR (XP, φ)
a.	[A[B[CD]]]	(A(B(CD))) ~ ((AB)(CD))	**W**			**L**	W	
b.	[A[B[CD]]]	(A(B(CD))) ~((A(B))(CD))		**W**		**L**	W	
c.	[A[[BC]D]]	(A((BC)D)) ~ (A(BC)D)			**W**	**L**	W	
d.	[[[AB]C]D]	((AB)(CD)) ~ (((AB)C)D)				**W**	**L**	**L**

Each row in (20) is an ERC. For the support to be valid for L.12, each row must contain a W-cell to the left of all L-cells. Working our way from bottom to top, we learn from (20d) that BinMax(φ,ω) must dominate Match(XP,φ) and AlignR(XP,φ). The winner, [[[AB]C]D] → ((AB)(CD)), violates BinMax(φ,ω) once, since the maximal φ contains four words, while the loser, [[[AB]C]D] → (((AB)C)D), violates it twice: once for the four-word φ (ABCD), and once for the three-word φ (ABC). The "W" in the BinMax(φ,ω) column of row (20d) expresses this preference.

The L-cells in row (20d) come from the fact that the winning mismatching candidate [[[AB]C]D] → ((AB)(CD)) is less well matched, and has less right-edge alignment, than the perfectly matching loser. The winner violates

MATCH(XP,φ) once, since it fails to match the XP containing A, B, C. It also violates ALIGNR(XP,φ) once, since the word C is at the right edge of an XP in the input, but not at the right edge of a φ in the output. The loser has no violations for these mapping constraints.

Moving up a row, (20c) reveals that *either* BINMAX(φ,branches) *or* MATCH (XP,φ) must dominate BINMAX(φ,ω) to derive the mapping [A[[BC]D]] → (A((BC)D)). Unlike row (20d), this ERC contains a disjunction. The intended winner (A((BC)D)) will prevail as long as either of the two W-cells precedes the lone L-cell. However, row (20d) has already established the ranking BINMAX(φ,ω) ≫ MATCH(XP,φ). So taken together, rows (20c, d) establish the ranking BINMAX(φ,branches) ≫ BINMAX(φ,ω). The "W" in the BINMAX(φ,branches) cell of row (20c) comes from the fact that the winner, with output structure (A((BC)D)), is perfectly binary-branching, while the loser, with output structure (A(BC)D), contains a ternary-branching φ, namely (AφD), and thereby incurs one violation of branch-counting binarity.

Like row (20c), row (20b) contains two W-cells and one L-cell; BINMIN(φ,ω) and MATCH(XP,φ) favor the winner [A[B[CD]]] → (A(B(CD))), an isomorphic mapping, while BINMAX(φ,ω) favors the loser ((A(B))(CD)). The winner violates neither BINMIN(φ,ω) nor MATCH(XP,φ), while the loser incurs a violation of BINMIN(φ,ω) for the phrase ($_{\varphi}$ B), and a violation of MATCH(XP,φ) for failing to match the XP containing B, C, D. BINMAX(φ,ω), by contrast, favors the loser in (20b) for exactly the same reason that it favored the winner in (20d): the rebracketed output contains one φ dominating more than two ωs, while the matching output contains two that do this. Although this ERC is disjunctive, together with the ERC in row (20d) it shows that BINMIN(φ,ω) must dominate BINMAX(φ,ω), by the same logic discussed for rows (20c, d). That is, MATCH(XP,φ) cannot be the reason for choosing the perfectly matching winner in (20b), because the rebracketing in (20d) shows that BINMAX(φ,ω) dominates MATCH(XP,φ).

Finally, row (20a) shows that ALIGNL(XP,φ) must dominate BINMAX(φ,ω) to derive the mapping [A[B[CD]]] → (A(B(CD))). The fully right-branching input has three non-coinciding left XP-edges to align to left φ-edges, and although the loser's rebracketing is preferred by BINMAX(φ,ω), it fails to align the XP left edge that precedes B.

15.5 Pure MATCH

It was shown in Section 15.2 that Match Theory, appropriately formulated, gives rise to the prosodic rebracketing seen in Japanese four-word left-branching phrases. However, the constraint ranking that does so incorrectly predicts that right-branching four-word phrases, i.e., [A[B[CD]]], will have the prosodic structure ((AB)(CD)). To see why, consider the system S.Msp.Mps, which

only contains MATCH constraints (Match syntax→prosody, Match prosody→syntax).[12]

(21) The system S.MspMps
- a. GEN.MspMps = GEN.MspAsp, defined in (17a)
- b. CON.MspMsp
  - i. Mapping constraints
    - MATCH(XP,φ), defined in (17b.i)
    - MATCH(φ,XP): Assign one violation for every node of category φ in the prosodic tree such that there is no node of category XP in the syntactic tree that dominates all and only the same terminal nodes as φ.
  - ii. Markedness constraints on surface prosody, defined in (17b.ii)
    - BINMIN(φ,ω)
    - BINMAX(φ,ω)
    - BINMAX(φ,branches)

The candidate space in S.MspMps is identical to that in S.MspAps. The constraint set differs in that the prosody-to-syntax mapping constraint MATCH(φ,XP) replaces the two alignment constraints. MATCH(XP,φ) and the three binarity constraints remain.

The factorial typology of S.MspMps contains four languages. In two of these, left-branching [[[AB]C]D] maps to ((AB)(CD)) as in Japanese. This occurs when BINMAX(φ,branches) ≫ MATCH(φ,XP) and BINMAX(φ,ω) ≫ MATCH(XP,φ), MATCH(φ,XP). But in these same two languages, the four-word right-branching syntax maps to the same prosody. There is no ranking under which [[[AB]C]D] → ((AB)(CD)) is optimal and [A[B[CD]]] → ((AB)(CD)) is nonoptimal, or vice versa. This problem—call it the Asymmetry Problem—is revealed by the following two elementary ranking conditions:

(22) The Asymmetry Problem

	Input	Winner ~ Loser	MATCH (XP,φ)	MATCH (φ,XP)	BINMAX (φ,ω)	BINMIN (φ,ω)	BINMAX (φ,branches)
a.	[[[AB]C]D]	((AB)(CD)) ~ (((AB)C)D)	L	L	W	e	e
b.	[A[B[CD]]]	(A(B(CD))) ~ ((AB)(CD))	W	W	L	e	e

In (22a), BINMAX(φ,ω) favors the mismatching winner, which violates it once, while the perfectly matching loser violates it twice. In (22b), MATCH(XP,φ) and MATCH(φ,XP) both prefer the perfectly matching winner over the mismatching

[12] The system S.MspMps is available on the SPOT interface. It is labeled "Japanese ×: MATCH only" in the section "Built-in systems."

loser, while BinMax(φ,ω) prefers the loser. Since all of the prosodic structures in (22) are perfectly binary-branching, the other two binarity constraints have no preferences. No ranking of the constraints in (22) yields the desired optima in both (22a) and (22b).

The Asymmetry Problem is deeply intractable in pure Match Theory (MT). No constraint from the MT literature, as far as we know, can resolve the contradiction. The problem is rooted in the symmetric nature of MT constraints. Constraints of the form Match(α,β) are direction-insensitive, and by definition they cannot detect the difference between a mapping α→β and a mapping α'→β', where α is the mirror image of α' and β the mirror image of β'.[13]

Of course, MT does not just consist of mapping constraints, but also has markedness constraints. In the markedness arena, there is more leeway to define a constraint assigning different violation counts to a prosodic tree α and its mirror image α'. Some markedness constraints, like EqualSisters (Myrberg 2013), are like Match in that they cannot distinguish α and α'. But others are asymmetrical, notably StrongStart.

(23) StrongStart (Selkirk 2011; Elfner 2012):
Assign one violation for every node in the prosodic tree whose leftmost daughter is lower on the prosodic hierarchy than its sister immediately to its right.

Concretely, StrongStart is violated when a prosodic constituent π begins with ($_{\pi}$ ω φ …). The initial ω in π is "weak" compared to the φ to its right, and StrongStart demands that the initial node of π be comparatively "strong," as in ($_{\pi}$ φ φ …), ($_{\pi}$ φ ω …), or ($_{\pi}$ ω ω …). Thus, the strictly right-branching (A(B(CD))) violates StrongStart twice, once for (A φ) and once for (B φ), while strictly left-branching (((AB)C)D) violates it zero times.

But although StrongStart is asymmetry-sensitive, it does not solve the Asymmetry Problem. Consider the pair of candidates from (22), but with StrongStart added to the constraint set.[14]

[13] By "mirror image," we mean "mirror image in terms of bracketing." The labels of the terminal nodes are irrelevant.

[14] The branch-binarity constraints are omitted since they were shown in (22) not to prefer the winner or loser on either line.

(24) The Asymmetry Problem and StrongStart

	Input	Winner ~ Loser	StrongStart	Match (XP,φ)	Match (φ,XP)	BinMax (φ,ω)
a.	[[[AB]C]D]	((AB)(CD)) ~ (((AB)C)D)	e	L	L	W
b.	[A[B[CD]]]	(A(B(CD))) ~ ((AB)(CD))	L	W	W	L

StrongStart has no preference regarding the choice in (24a). Both ((AB)(CD)) and (((AB)C)D) satisfy it perfectly, since neither contains a phrase ($_{\varphi}$ ω φ …). Thus, the ERC in (24a) asserts that BinMax(φ,ω) must dominate the two Match constraints. In (24b), StrongStart favors the loser. The winner, (A(B(CD))), violates StrongStart twice: once for (A φ), and once for (B φ). The ERC in (24b) therefore states that either Match(XP,φ) or Match(φ,XP) must dominate both StrongStart and BinMax(φ,ω). The ERCs are still contradictory, as in (22), so the Asymmetry Problem remains.

Since StrongStart favors rebracketing the strictly right-branching [A[B[CD]]] instead of left-branching [[[AB]C]D], we might wonder whether a reverse constraint StrongEnd would not solve the problem.

(25) StrongEnd (reverse of Elfner's 2012 StrongStart; to be rejected)
Assign one violation for every node in the prosodic tree whose rightmost daughter is lower on the prosodic hierarchy than its sister immediately to its left.

But adding this constraint to Con.MspMps does not solve the Asymmetry Problem either, as shown in (26). (As in (24), BinMax(φ,branches) is omitted because each winner and loser under consideration satisfies it perfectly.)

(26) The Asymmetry Problem and StrongEnd

	Input	Winner ~ Loser	StrongEnd	Match (XP,φ)	Match (φ,XP)	BinMax (φ,ω)
a.	[[[AB]C]D]	((AB)(CD)) ~ (((AB)C)D)	W	L	L	W
b.	[[A[BC]]D]	((A(BC))D) ~ ((AB)(CD))	L	W	W	L

While StrongEnd favors rebracketing in (26a), it also incorrectly does so in the mixed-branching (26b). For rebracketing to occur in (26a), either StrongEnd or BinMax(φ,ω) must dominate both Match(XP,φ) and Match(φ,XP). But for matching to occur in (26b), either Match(XP,φ) or Match(φ,XP) must dominate both StrongEnd and BinMax(φ,ω). StrongEnd does not yield the Japanese pattern. The failure of StrongEnd is a welcome result, since, as pointed out in

Ito and Mester (2019a), a constraint that directly strengthens the end of prosodic units runs afoul of the evidence from phonetics and psycholinguistics that has accumulated over the years since Beckman (1997), Smith (2005b), etc.

15.6 Pure ALIGN

Since the Asymmetry Problem cannot be solved using constraints from the literature on Match Theory, and MATCH(XP,φ) does not dominate any constraints in L.12 of S.MspAsp, one might attempt to dispense with MATCH altogether. However, this turns out not to work either. Consider a final system, S.AspAps, defined as follows:

(27) The system S.AspAps

a. GEN.AspAps = Gen.MspAsp, defined in (17a)

b. CON.AspAps

i. Syntax-to-prosody mapping constraints
- ALIGNL(XP,φ), defined in (17b.i)
- ALIGNR(XP,φ), defined in (17b.i)

ii. Prosody-to-syntax mapping constraints
- ALIGNL(φ,XP): Assign one violation for every node of category φ in the prosodic tree whose left edge is not aligned with the left edge of a node of category XP in the syntactic tree.
- ALIGNR(φ,XP): Assign one violation for every node of category φ in the prosodic tree whose right edge is not aligned with the right edge of a node of category XP in the syntactic tree.

iii. Markedness constraints on surface prosody
- BINMIN(φ,ω), defined in (17b.ii)
- BINMAX(φ,branches), defined in (17b.ii)
- BINMAX(φ,ω), defined in (17b.ii)

GEN.AspAps is the same as in the systems already discussed. The constraint set contains four alignment constraints: the two syntax–prosody alignment constraints from S.MspAsp, and two prosody-syntax alignment constraints.[15] The three binarity constraints from the previous systems are also included.

The factorial typology of S.AspAps contains thirty-four languages, but none of these exhibit the Japanese pattern in (16). However, this system's failure differs from that of the pure MATCH system. With left- and right-alignment constraints, there is no longer an Asymmetry Problem. Instead, a new problem arises, which we call the Ambivalence Problem, illustrated in (28).

[15] The availability of prosody-syntax alignment alongside syntax–prosody alignment follows from the theory of Generalized Alignment (McCarthy and Prince 1993). For analyses involving prosody-syntax alignment, see Zerbian (2007) and Cheng and Downing (2016).

(28) The Ambivalence Problem

	Input	Winner ~ Loser	AlignL (XP,φ)	AlignR (XP,φ)	AlignL (φ,XP)	AlignR (φ,XP)	Bin Max (φ,ω)	Bin Min (φ,ω)	Bin Max (φ,branches)
a.	[A[[BC]D]]	(A((BC)D)) ~ ((A(BC))D)	e	e	e	e	e	e	e
b.	[[A[BC]]D]	((A(BC))D) ~ (A((BC)D))	e	e	e	e	e	e	e

The Ambivalence Problem is the inability of this system to adjudicate between matching and mismatching candidates when the input contains a medial two-word XP, i.e., [$_{XP}$ BC]. Recall from (5b, c), schematized in (16), that in Japanese these syntactic inputs are mapped to perfectly matching prosodic outputs. In S.AspAps, no constraint distinguishes a mapping […[BC]…] → (A((BC)D)), where the last three words form a constituent, from a mapping […[BC]…] → ((A(BC))D), where the first three words form a constituent, as indicated by the "e" in every cell of the ERCs in (28).

The binarity constraints do not distinguish (A((BC)D)) and ((A(BC))D), because these are mirror images of each other, and binarity constraints are not sensitive to asymmetry. The forms in (28) both fully satisfy BinMax(φ,branches) and BinMin(φ,ω), and both violate BinMax(φ,ω) twice: once for the four-word φ, and once for the three-word φ.

The four alignment constraints, which can distinguish between left- and right-branching in trees with only one level of embedding, are fully satisfied by all four mappings: input→winner and input→loser in (28a), and input→winner and input→loser in (28b). This is because in each input, words A and B are XP-initial and XP-nonfinal, while words C and D are XP-final and XP-noninitial. Similarly, in each input, A and B are φ-initial and φ-nonfinal, while C and D are φ-final and φ-noninitial. Thus, every alignment constraint is perfectly satisfied. Every XP-boundary is aligned with an appropriate φ-boundary, and vice versa. But what of other constraints not included in Con.AspAps? Additional markedness constraints do not help. Just as in the purely Match-Theoretic system S.MspMps, adding StrongStart or its hypothetical opposite StrongEnd to Con would not solve the Ambivalence Problem either. Since the problem involves symmetric structures, i.e., (28a) and (28b), asymmetric constraints cannot solve the problem.

Alignment-based theories of syntax–prosody mapping often also include a constraint Wrap(XP), which states that every XP must be contained in a φ (Truckenbrodt 1995, 1999). Wrap(XP) differs from Match(XP,φ) in that it does not demand that every XP have a perfectly *matching* φ, but only that each XP be entirely contained within a φ. For instance, while the mapping […A…[$_{XP}$ BC]…D…] → ($_{φ}$ ABCD) violates Match(XP,φ), since XP has no matching φ, it satisfies Wrap(XP), since all of XP is contained in a single φ. In a mapping like […A…[$_{XP}$ BC]…D…] → ($_{φ}$ AB)($_{φ}$ CD), both Match(XP,φ) and Wrap(XP) are violated. While Wrap(XP) is not an alignment constraint, the fact that it is often

used alongside alignment constraints raises the question of whether adding it to CON.AspAps would solve the Ambivalence Problem. The answer is that WRAP(XP) would have no effect at all. GEN.AspAps is defined such that every candidate satisfies WRAP(XP). Since every output candidate has a root node of category φ, every input XP is wrapped in this maximal φ. So WRAP(XP) is unable to favor [A[[BC]D]] → (A((BC)D)) over [A[[BC]D]] → *((A(BC))D) or [[A[BC]]D] → ((A(BC))D) over [[A[BC]]D] → *(A((BC)D)).

Thus, the pure alignment system S.AspAps does not produce the Japanese pattern, nor do we know of any existing constraints that could be added (other than MATCH constraints) to solve the Ambivalence Problem. It is of course not unthinkable that a completely new type of constraint could be invented, but we leave this to potential pure alignment theorists.

15.7 Conclusion

We have shown that both MATCH and ALIGN constraints are needed to account for the rebracketing asymmetry in Japanese phonological phrasing. Systems with MATCH but not ALIGN run into an intractable Asymmetry Problem: it is not possible to map syntactic input structures to outputs that are not mirror images of each other, as needed for Japanese four-word left-branching and right-branching inputs. On the other hand, systems with ALIGN but not MATCH face the Ambivalence Problem, whereby certain deeply embedded syntactic constituents cannot unambiguously map to isomorphic output constituents. ALIGN constraints solve MATCH's Asymmetry Problem, since they distinguish between left and right edges, and MATCH constraints solve ALIGN's Ambivalence Problem, since they favor strict input-output isomorphism. Thus, a full theory of syntax–prosody mapping in Optimality Theory requires both MATCH and ALIGN. This discovery is notable because MATCH and ALIGN constraints are typically not used together in the same analysis. The original conception of Match Theory in Selkirk (2011) was that it would supplant Align Theory. However, this case study of Japanese phrasing reveals that the full range of syntax–prosody mappings can only be captured if both constituent-oriented MATCH constraints and edge-oriented ALIGN constraints work together within the same system.

Why should MATCH and ALIGN both be necessary? We suggest that the two are in fact both motivated. Conceptually, syntax cares about constituents, and therefore MATCH best reflects the demands of syntax-to-prosody mapping. Phonology, on the other hand, cares about edges a great deal. Alignment constraints are not specific to the syntax–prosody interface, but are regularly used in the analysis of footing, stress placement, reduplication, and other phonological or morphophonological phenomena. We can understand constituent-based MATCH constraints as representing the demands of syntax, and edge-based ALIGN constraints as representing a phonological requirement. On this view, perhaps it is

no longer surprising that both MATCH and ALIGN would co-occur and interact to represent the full range of syntax–prosody mappings.

The working system presented here included three mapping constraints, MATCH(XP,φ), ALIGNL(XP,φ), and ALIGNR(XP,φ). All three of them enforce syntax-to-prosody mappings. That is, the number of violations they assign is determined by the number of *syntactic* objects that lack prosodic correspondents. We presented this particular system for expositional clarity, since the syntax-to-prosody mapping constraints are the most familiar and widely used. However, we also tested ten additional systems that used the same markedness constraints but varied in their mapping constraints. So many combinations were possible because we included all possible combinations of the mapping constraints, and included prosody-to-syntax mapping constraints in addition to the syntax-to-prosody mapping constraints. Prosody-to-syntax mapping constraints assign violations based on the number of *prosodic* objects that lack syntactic correspondents, and have been a part of Match Theory from its inception (Selkirk 2011). Here, the relevant prosody-to-syntax constraint is MATCH(φ,XP), which is violated when the output contains a phonological phrase that does not correspond to any XP in the input—that is, when a φ has been "inserted" relative to the perfectly matching candidate. This gives us four types of mapping constraints to consider: MATCH vs. ALIGN, and syntax-to-prosody (SP) vs. prosody-to-syntax (PS). Taking every possible combination of two or more of these types of mapping constraint produces the combinations in (29a–k). In every system that included at least one MATCH constraint and at least one ALIGN constraint, the typology included a language with the Japanese phrasing pattern. In contrast, as we have described, neither the pure MATCH nor the pure ALIGN system was able to generate the attested pattern of Japanese phrasing.[16]

[16] A reviewer asks whether similar results obtain if CON excludes right-alignment constraints of the form ALIGNR(α,β), and allows only left-alignment constraints ALIGNL(α,β), pointing out that our analysis of Japanese in S.MspAsp requires ALIGNL(XP,φ) but not ALIGNR(XP,φ), and that there are precedents for proposing such a left-right asymmetry in other areas of phonology (Nelson 2003; Alber 2005). Every system described in this table that involves ALIGNL(XP,φ) and at least one MATCH constraint, i.e., (29a, b, d, e, f, h) would still account for the Japanese pattern if all right-alignment constraints were removed from CON. But systems that include prosody-to-syntax alignment constraints, but not syntax-to-prosody alignment constraints, actually exhibit the reverse behavior; in systems (29c, g, i), the alignment constraint needed for Japanese is ALIGNR(φ,XP), not ALIGNL(φ,XP). So an analysis of Japanese needs either ALIGNL(XP,φ) or ALIGNR(φ,XP), plus a MATCH constraint.

(29) Summary of the different systems examined

	MATCH (XP,φ)	MATCH (φ,XP)	ALIGN L/R (XP,φ)	ALIGN L/R (φ,XP)	Languages in Factorial Typology	Typology contains the Japanese pattern?
a.	✓	✓	✓	✓	34	Yes
b.	✓	✓	✓		20	Yes
c.	✓	✓		✓	8	Yes
d.	✓		✓	✓	34	Yes
e.		✓	✓	✓	34	Yes
f.	✓		✓		14	Yes
g.	✓			✓	8	Yes
h.		✓	✓		20	Yes
i.		✓		✓	7	Yes
j.	✓	✓			4	No
k.			✓	✓	34	No

Given that ten out of the twelve systems here all succeed in representing the Japanese phrasing pattern, a question arises as to which combination of MATCH and ALIGN is the most desirable. Including all four types of mapping constraints in the same system works, but increases the size of the typology. On the other hand, it is also problematic to exclude the prosody-to-syntax mapping constraints altogether. This is particularly clear when we consider recent proposals to subsume Match Theory under General Correspondence Theory in Optimality Theory (as Syntax-Prosody MAX/DEP), as in Ito and Mester (2019a), cf. also the related proposal in Selkirk's (2017, 2019) serialist (two-stage) conception of prosodic structure formation. From this perspective, leaving out prosody-to-syntax mapping constraints (as we did in Msp.Asp, Section 15.4) is as problematic as leaving MAX or DEP out of an analysis of syllable structure (see Gouskova 2007 for discussion). Without it, there is nothing to restrict insertion. We leave the question open, noting only that many combinations of MATCH and ALIGN are possible. A detailed investigation of the different predictions of these combinations is a topic for future research.

Our analysis of the Japanese data predicts that similar asymmetries would occur in other languages. That is, we predict a language in which a four-word right-branching syntactic structure [A[B[CD]]] should undergo rebracketing to ((AB)(CD)), while all other three- and four-word syntactic structures, including strictly left-branching structures, should map to perfectly matching prosodic outputs (L.3 in (18)). We are not currently aware of a case exactly like this, and if none were ever found, this could suggest that the typology presented here is symmetric in ways it should not be. However, it is at least possible for four-word right-branching structures to rebracket in this way. In Irish, a sentence consisting of a verb, a one-word subject, and a two-word direct object, [$_{\Sigma P}$ V [$_{TP}$ [$_{NP}$ S] [$_{VP}$ [$_{NP}$ [$_{NP}$ N] [$_{AP}$ A]]]]], maps to a balanced structure ($_{\varphi}$ ($_{\varphi}$ V S) ($_{\varphi}$ N A)) (Elfner

2012: 108). While this is suggestive, we leave for future research the question of whether there are languages with the mirror image of the Japanese pattern, which could inform the validity of the MATCH+ALIGN analysis.

More broadly, we conclude that Optimality-Theoretic work on the syntax-prosody interface (and other domains) requires consideration of *all* candidates and *all* constraints as defined within a system. Leaving candidates out can result in incorrect claims about typology and ranking, as has been the case for the Japanese phrasing pattern at hand. Only with a fully defined OT system can we be certain that our results in ranking and typology are sound, and the SPOT program has been indispensable in exploring the properties of the system.

Acknowledgments

This work was funded by National Science Foundation Grant #1749368 to Junko Ito and Armin Mester. We are grateful to the audiences at the SPOT2 2018 workshop (UC Santa Cruz), ICPP 2019 (NINJAL, Tokyo), and SOTA IV 2020 (Eckerd College, Florida) for helpful comments and questions. We would also like to thank SPOT co-developer Ozan Bellik for his programming expertise. All mistakes are our own.