SPOT App

Syntax-Prosody in OT

Jenny Bellik & Nick Kalivoda, UC Santa Cruz

October 7, 2018 @ the Annual Meeting on Phonology

SPOT interface:
https://people.ucsc.edu/~jbellik/research/spot/interface1.html

Thanks to

Ozan Bellik
(Collaborator)

Junko Ito
(Mentor)

Armin Mester
(Mentor)

Welcome to the SPOT tutorial

To follow along, please visit the links posted on the SPOT tutorial page on the AMP2018 website: http://phonology.ucsd.edu/program/sunday/spot-tutorial/

- Interface: people.ucsc.edu/~jbellik/research/spot/interface1.html
- Codebase: github.com/syntax-prosody-ot

Why a syntax-prosody app?

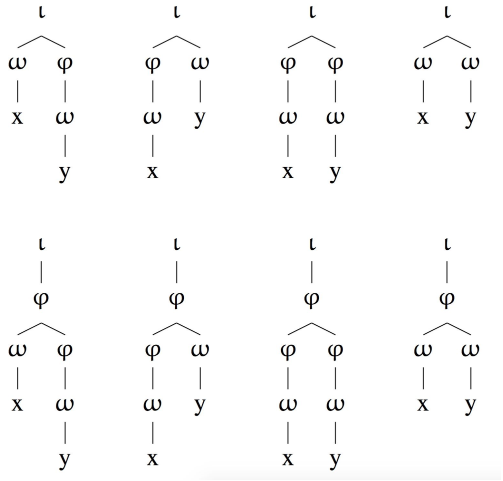
In Optimality Theory (Prince & Smolensky 1993), the winning output is supposed to be optimal

- amongst **all** the outputs of GEN
- as evaluated by CON.

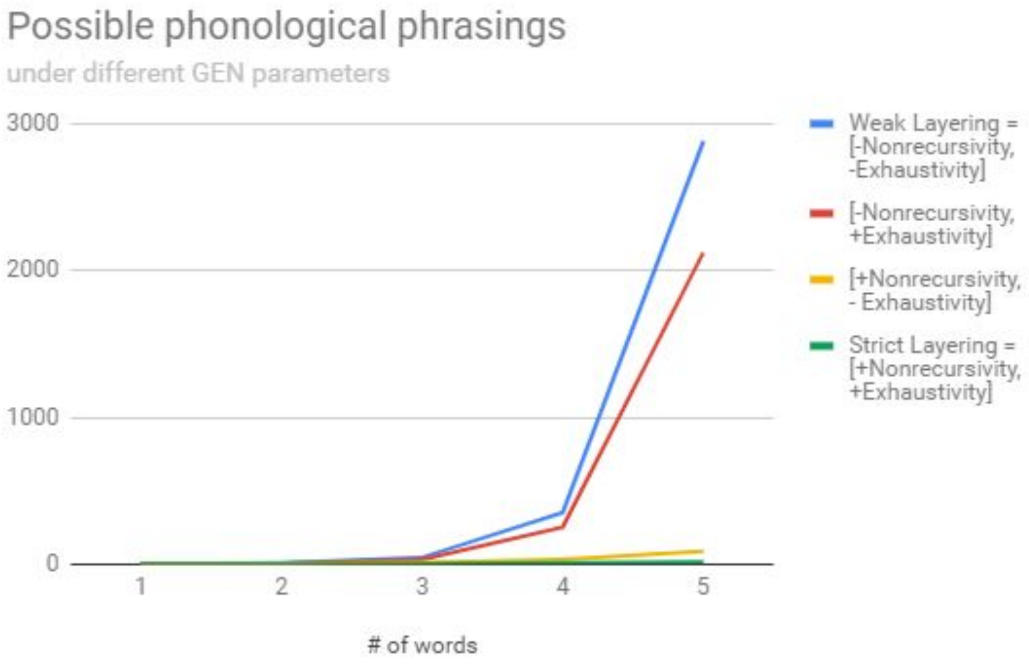
Why a syntax-prosody app?

In syntax-prosody mapping:

- Both inputs and outputs are tree structures.
- Outputs proliferate!



Why a syntax-prosody app?



Why a syntax-prosody app?

- Automatically {generate, evaluate} the thousands of output candidates
- Existing automatic candidate generation/evaluation software cannot handle tree structures of arbitrary depth
- That’s why we built SPOT!

This workshop

- Introduce the SPOT application
- Demonstrate how to use the GUI
- Work through examples that use JavaScript directly
    - Phrasing in Kinyambo, Japanese, Italian

SPOT application

SPOT is an open-source JavaScript application in development since 2014 that automates all three components of an OT system (GEN, EVAL, CON).

- Online GUI
- Source code available on Github

Using the GUI

1. Navigate to https://people.ucsc.edu/~jbellik/research/spot/interface1.html
2. Choose constraints by clicking on checkboxes
3. Build your input syntactic tree
4. Select parameters for GEN
5. Click “Submit” and scroll down for a tableau

Go ahead, try it out!

Kinyambo

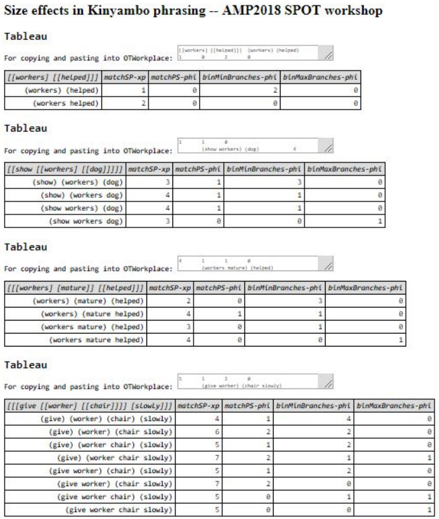
In Kinyambo, High Tone Deletion (1) diagnoses the φ-boundaries in (2) (Bickmore 1989, 1990).

(1) $H \rightarrow \varnothing \;/\; (_{\varphi} \ldots (_{\omega} \ldots \underline{\quad} \ldots) \; (_{\omega} \ldots H \ldots) \ldots)$

(2) a. $(_{\varphi}$ abak$\underline{\text{o}}$zi b$\underline{\text{á}}$kaj$\underline{\text{ú}}$na)
‘the workers helped’

b. $(_{\varphi}$ abak$\underline{\text{o}}$zi bak$\underline{\text{ú}}$ru) $(_{\varphi}$ b$\underline{\text{á}}$kaj$\underline{\text{ú}}$na)
‘the mature workers helped’

Kinyambo



Size effects in Kinyambo phrasing -- AMP2018 SPOT workshop

Tableau

For copying and pasting into OTWorkplace:

```
[[workers] [[helped]]]  (workers) (helped)
1	0	2	0
```

[[workers] [[helped]]]	*matchSP-xp*	*matchPS-phi*	*binMinBranches-phi*	*binMaxBranches-phi*
(workers) (helped)	1	0	2	0
(workers helped)	2	0	0	0

Tableau

For copying and pasting into OTWorkplace:

```
1	1	0
	(show workers) (dog)		4
```

[[show [[workers] [[dog]]]]]	*matchSP-xp*	*matchPS-phi*	*binMinBranches-phi*	*binMaxBranches-phi*
(show) (workers) (dog)	3	1	3	0
(show) (workers dog)	4	1	1	0
(show workers) (dog)	4	1	1	0
(show workers dog)	3	0	0	1

Tableau

For copying and pasting into OTWorkplace:

```
4	1	1	0
	(workers mature) (helped)
```

[[[workers] [mature]] [[helped]]]	*matchSP-xp*	*matchPS-phi*	*binMinBranches-phi*	*binMaxBranches-phi*
(workers) (mature) (helped)	2	0	3	0
(workers) (mature helped)	4	1	1	0
(workers mature) (helped)	3	0	1	0
(workers mature helped)	4	0	0	1

Tableau

For copying and pasting into OTWorkplace:

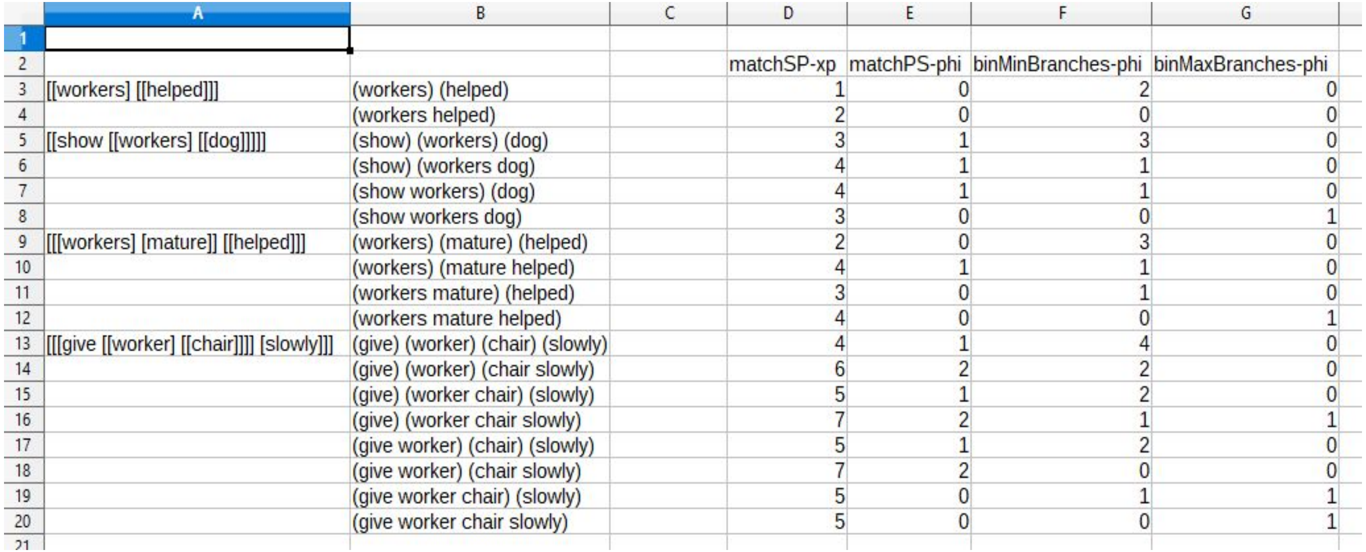
```
5	1	2	0
	(give worker) (chair slowly)
```

[[[give [[worker] [[chair]]]] [slowly]]]	*matchSP-xp*	*matchPS-phi*	*binMinBranches-phi*	*binMaxBranches-phi*
(give) (worker) (chair) (slowly)	4	1	4	0
(give) (worker) (chair slowly)	6	2	2	0
(give) (worker chair) (slowly)	5	1	2	0
(give) (worker chair slowly)	7	2	1	1
(give worker) (chair) (slowly)	5	1	2	0
(give worker) (chair slowly)	7	2	0	0
(give worker chair) (slowly)	5	0	1	1
(give worker chair slowly)	5	0	0	1

- Suppose I want to try to capture these phrasings in Match Theory (cf. Bellik & Kalivoda 2015), with CON = {MatchSP-XP, MatchPS-φ, BinMin, BinMax}
- To generate the four tableaux at left, I could manually create them one by one in the GUI

Kinyambo

Then I can copy/paste the results into one big .csv tableau:

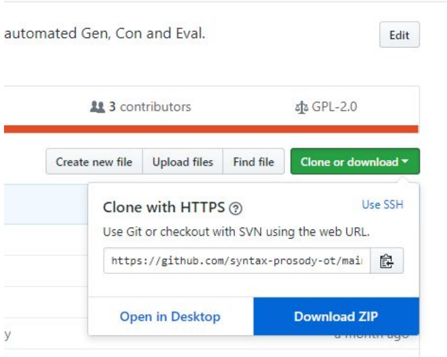


	A	B	C	D	E	F	G
1							
2				matchSP-xp	matchPS-phi	binMinBranches-phi	binMaxBranches-phi
3	[[workers] [[helped]]]	(workers) (helped)		1	0	2	0
4		(workers helped)		2	0	0	0
5	[[show [[workers] [[dog]]]]]	(show) (workers) (dog)		3	1	3	0
6		(show) (workers dog)		4	1	1	0
7		(show workers) (dog)		4	1	1	0
8		(show workers dog)		3	0	0	1
9	[[[workers] [mature]] [[helped]]]	(workers) (mature) (helped)		2	0	3	0
10		(workers) (mature helped)		4	1	1	0
11		(workers mature) (helped)		3	0	1	0
12		(workers mature helped)		4	0	0	1
13	[[[give [[worker] [[chair]]]] [slowly]]]	(give) (worker) (chair) (slowly)		4	1	4	0
14		(give) (worker) (chair slowly)		6	2	2	0
15		(give) (worker chair) (slowly)		5	1	2	0
16		(give) (worker chair slowly)		7	2	1	1
17		(give worker) (chair) (slowly)		5	1	2	0
18		(give worker) (chair slowly)		7	2	0	0
19		(give worker chair) (slowly)		5	0	1	1
20		(give worker chair slowly)		5	0	0	1
21							

Kinyambo

- But can’t we automate that kind of repetitious work?!
- We can, and we did! https://github.com/syntax-prosody-ot/main/blob/master/tutorials/AMP2018_kinyambo_example.html
- But such automation requires going beyond the GUI.

Beyond the GUI



automated Gen, Con and Eval.

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

`https://github.com/syntax-prosody-ot/mai`

- To use SPOT outside the GUI, you'll need to download the codebase.
- Go to the SPOT Github website: https://github.com/syntax-prosody-ot/main
- Click on the green "Clone or Download" button, then on "Download ZIP"
  - Git users can also clone the repository, make their own branch, etc.
  - Email us if you are interested in collaborating via Github to work on SPOT!

Beyond the GUI

- After downloading, unzip the file and open “main-master”
- Now you have a local copy of the entire SPOT codebase on your computer!
- No further installation is necessary—every browser already has JavaScript and html.

Kinyambo

To automatically generate the Kinyambo master tableau, open: **main/tutorials/AMP2018_kinyambo_example.html**

- Right-click and open with a text editor (ex. Notepad, TextEdit) to view and edit the code
- Double-click to open in the browser and see it execute

More options: Recursion

- The actual prosodic structures in Kinyambo do not involve any recursion – they conform to Strict Layering
- But recursive prosodic structures are also possible (Ito & Mester 2003)
- These can be generated in SPOT by turning off the GEN option `obeysNonRecursivity`
- Let’s see an example from Japanese: **main/tutorials/AMP2018_japanese_example.html**

Japanese

based on Ito & Mester (2013, 2017)

Left-branching syntax: [[[α]β]γ]:

(3) [[[U]U]U] → ((U U) U)
(4) [[[U]A]A] → ((U A) (A))
(5) [[[U]A]U] → ((U A) (U))
(6) [[[U]U]A] → ((U U) (A))
(7) [[[A]A]A] → (((A) (A)) (A))
(8) [[[A]A]U] → (((A) (A)) (U))
(9) [[[A]U]A] → (((A) (U)) (A))
(10) [[[A]U]U] → (((A) (U)) (U))

Japanese

based on Ito & Mester (2013, 2017)

To summarize:

(11) [[[U]U]U] → ((ω ω) ω)

(12) [[[U]A]A] → ((ω ω) (ω))
[[[U]A]U]
[[[U]U]A]

(13) [[[A]X]Y] → (((ω) (ω)) (ω))

Japanese

based on Ito & Mester (2013, 2017)

- CON = {MatchSP-XP$_{Max}$, MatchSP-φ, MatchSP-XP$_{Non\text{-}Unary}$, MatchPS-φ$_{NonUnary}$, MatchPS-φ, BinMinBranches-φ, BinMax$_{Branches}$-φ, BinMax$_{2Words}$-φ, EqualSisters$_{Adj}$-φ, EqualSisters$_{Adj}$2-φ, AccentAsHead-φ, NoLapseL}

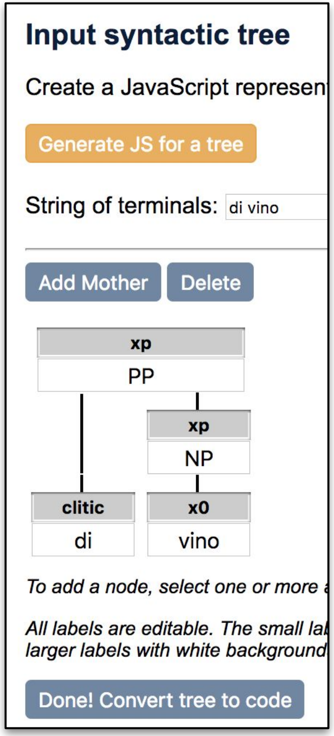
main/tutorials/AMP2018_japanese_example.html

More options: Clitics

- So far we’ve been assuming that every syntactic word maps to a prosodic word
- What about clitics?
- In Japanese, we had the clitic -*no*, which we ignored by placing inside the ω of its host:

(14) amerika-**no** tomodachi-**no** pasokon
America-GEN friend-GEN computer
'my American friend's computer'

More options: Clitics



Input syntactic tree

Create a JavaScript represen

String of terminals: di vino

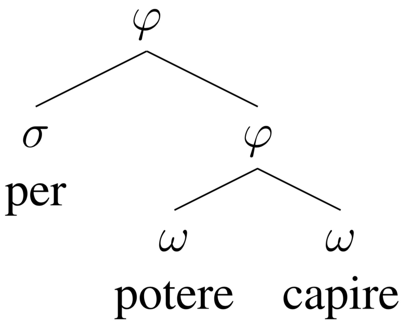
To add a node, select one or more

All labels are editable. The small la
larger labels with white background

- **Problem**: What if you want to generate prosodic trees in which some syntactic terminals are *not* mapped to prosodic words?
- **Solution**: When building the syntactic tree, set the category of a desired clitic to “clitic” rather than “x0”
  - The clitic will receive the prosodic category “syll” (syllable).
  - It won’t count for prosodic constraints that look for ω
  - It also won’t count for mapping constraints that look for X^0
- Let’s illustrate with Italian.

Italian

based on Van Handel (2018)



- Function words in Italian, like P^0 *per* in (15), are generally clitics: lower on the PH than ω or φ.
- *Troncamento*: deletion of word-final unstressed mid vowels after sonorants; sensitive to φ-boundaries

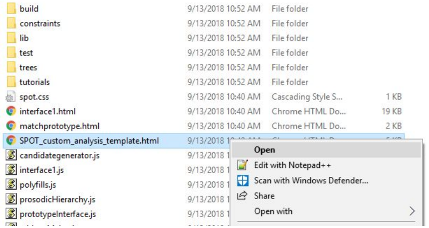
(15) a. per poter_ capire
b. ? per poter**e** capire
‘in order to be able to understand’

- How can we represent this in SPOT? **main/tutorials/AMP2018_italian_clitic_example.html**

Beyond the GUI

- Any of the html files in the tutorials folder can be adapted for your own analysis. (Just change the trees & constraints appropriately.)
- What if you also want to compare constraint sets?
    - Ex. You want to see what typologies a Match system vs an Align system generate for a set of several syntactic inputs.
- We have a template for this -- you don’t need to write your own JavaScript!

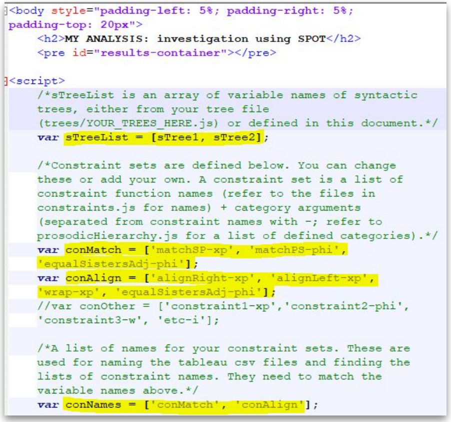
Customizing the template



build	9/13/2018 10:52 AM	File folder	
constraints	9/13/2018 10:52 AM	File folder	
lib	9/13/2018 10:52 AM	File folder	
test	9/13/2018 10:52 AM	File folder	
trees	9/13/2018 10:52 AM	File folder	
tutorials	9/13/2018 10:52 AM	File folder	
spot.css	9/13/2018 10:40 AM	Cascading Style S...	1 KB
interface1.html	9/13/2018 10:40 AM	Chrome HTML Do...	19 KB
matchprototype.html	9/13/2018 10:40 AM	Chrome HTML Do...	2 KB
SPOT_custom_analysis_template.html	9/13/2018 1		
candidategenerator.js	9/13/2018 1		
interface1.js	9/13/2018 1		
polyfills.js	9/13/2018 1		
prosodicHierarchy.js	9/13/2018 1		
prototypeInterface.js	9/13/2018 1		

- To make your own custom analysis with JavaScript, find **SPOT_custom_analysis_template.html** in the main SPOT directory.
- Right-click and open it in the text or html editor of your choice
  - *not* with the browser -- that is for displaying, not editing
  - Some basic text editors: Notepad, Notepad++ (Windows); Xcode, TextEdit, TextMate (Mac)

Customizing the template



```
<body style="padding-left: 5%; padding-right: 5%;
padding-top: 20px">
    <h2>MY ANALYSIS: investigation using SPOT</h2>
    <pre id="results-container"></pre>

<script>
    /*sTreeList is an array of variable names of syntactic
    trees, either from your tree file
    (trees/YOUR_TREES_HERE.js) or defined in this document.*/
    var sTreeList = [sTree1, sTree2];

    /*Constraint sets are defined below. You can change
    these or add your own. A constraint set is a list of
    constraint function names (refer to the files in
    constraints.js for names) + category arguments
    (separated from constraint names with -; refer to
    prosodicHierarchy.js for a list of defined categories).*/
    var conMatch = ['matchSP-xp', 'matchPS-phi',
    'equalSistersAdj-phi'];
    var conAlign = ['alignRight-xp', 'alignLeft-xp',
    'wrap-xp', 'equalSistersAdj-phi'];
    //var conOther = ['constraint1-xp','constraint2-phi',
    'constraint3-w', 'etc-i'];

    /*A list of names for your constraint sets. These are
    used for naming the tableau csv files and finding the
    lists of constraint names. They need to match the
    variable names above.*/
    var conNames = ['conMatch', 'conAlign'];
```

1. Edit `YOUR_TREES_HERE.js` or make your own tree file
2. Adjust the names of the trees in `sTreeList` (line 37)
3. Put in the constraint set(s) you want to use
   a. Examples are `conMatch` (line 40) and `conAlign` (line 41)
   b. All constraint set names must be listed under `conNames` (line 45)
4. If desired:
   a. Adjust GEN options on line 48.
   b. To display tableaux in the browser, delete the `//` before lines 70-72.
   c. Create a custom constraint by filling in line 17 & include its name in one or more of your constraint sets.

Customizing the template

- When finished, double-click on the html file to open it in the browser and view or download results.
- If it’s not working right, open the JS console to show errors.
  - Ignore the “No spot form” message.
- If you revise the html file in your text editor, reload the page in the browser to update.

Questions?

Thank you!

Please email us if any issues come up when you’re using SPOT, or if you think of a feature you’d like to use!
Jenny: jbellik@ucsc.edu, Nick: nkalivod@ucsc.edu

References

Bellik, Jennifer & Nick Kalivoda. 2016. Adjunction and Branching Effects in Syntax-Prosody Mapping. Hansson, G.O., Farris-Trimble, A., McMullin, K., & D. Pulleyblank. 2015. *Supplemental Proceedings of the 2015 Annual Meeting on Phonology*.

Bickmore, Lee. 1989. *Kinyambo Prosody*. Ph.D. thesis, UCLA.

— 1990. Branching Nodes and Prosodic Categories. In S. Inkelas & D. Zec (eds.) *The Phonology-Syntax Connection.*

Ito, Junko & Armin Mester. 2003. Weak Layering and Word Binarity. In Honma, T., M. Okazaki, T. Tabata, & S. Tanaka (eds.), *A New Century of Phonology and Phonological Theory: A Festschrift for Professor Shosuke Haraguchi on the occasion of his sixtieth birthday*, pp. 26-65. Kaitakusha, Tokyo.

— 2013. Prosodic subcategories in Japanese. *Lingua* 124, 20-40.

Meinschaefer, Judith. (2005). The prosodic domain of Italian troncamento is not the clitic group.

Meinschaefer, Judith. (2009) Lexical exceptionality in Florentine Italian troncamento. In C. Féry, F. Kügler & R. van de Vijver (eds.), *Variation and Gradience in Phonetics and Phonology* Vol. 14 (pp. 223-252). Walter de Gruyter.

Prince, Alan & Paul Smolensky. 1993/2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishing.

Truckenbrodt, Hubert. 1995. Phonological phrases: Their relation to syntax, focus, and prominence. MIT dissertation.

— 1999. On the relation between syntactic phrases and phonological phrases. Linguistic Inquiry 30(2). 219-255.

Van Handel, Nick. 2018. Italian troncamento in Match Theory. Handout.