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?

Possible phonological phrasings
under different GEN parameters

- Weak Layering = [-Nonrecursivity, -Exhaustivity]
- [-Nonrecursivity, +Exhaustivity]
- [+Nonrecursivity, -Exhaustivity]
- Strict Layering = [+Nonrecursivity, +Exhaustivity]

# of words
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!
In Kinyambo, High Tone Deletion (1) diagnoses the $\varphi$-boundaries in (2) (Bickmore 1989, 1990).

\[
H \rightarrow \emptyset / (\varphi \ldots (\omega \ldots \_ \ldots \ldots ) (\omega \ldots H \ldots ) \ldots )
\]

(2)  
\begin{enumerate}
\item a. \((\varphi \text{ abakozi bákajúna})\)  
\hspace{1cm} \text{‘the workers helped’}
\item b. \((\varphi \text{ abakozi bakúru} \varphi \text{ bákajúna})\)  
\hspace{1cm} \text{‘the mature workers helped’}
\end{enumerate}
Kinyambo

- Suppose I want to try to capture these phrasings in Match Theory (cf. Bellik & Kalivoda 2015), with $\text{CON} = \{\text{MatchSP-XP}, \text{MatchPS-}\phi, \text{BinMin}, \text{BinMax}\}$
- To generate the four tableaux at left, I could manually create them one by one in the GUI
Then I can copy/paste the results into one big .csv tableau:
Kinyambo

- But can’t we automate that kind of repetitious work?!
- We can, and we did!
- But such automation requires going beyond the GUI.
Beyond the GUI

- 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: $[[[\alpha]\beta]\gamma]$:

(3) $[[[U]U]U] \rightarrow ((U U) U)$
(5) $[[[U]A]U] \rightarrow ((U A) (U))$
(6) $[[[U]U]A] \rightarrow ((U U) (A))$
(8) $[[[A]A]U] \rightarrow (((A) (A)) (U))$
(9) $[[[A]U]A] \rightarrow (((A) (U)) (A))$
(10) $[[[A]U]U] \rightarrow (((A) (U)) (U))$
To summarize:

(11) \[ [[[U]U]U] \rightarrow ((\omega \omega) \omega) \]

(12) \[ [[[U]A]A] \rightarrow ((\omega \omega) (\omega)) \]

(13) \[ [[[A]X]Y] \rightarrow (((\omega) (\omega)) (\omega)) \]
Japanese

based on Ito & Mester (2013, 2017)

\[ \text{CON} = \{ \text{MatchSP-XP}_{\text{Max}}, \text{MatchSP-}\varphi, \text{MatchSP-XP}_{\text{Non-Unary}}, \text{MatchPS-}\varphi_{\text{NonUnary}}, \text{MatchPS-}\varphi, \text{BinMinBranches-}\varphi, \text{BinMax}_{\text{Branches}}-\varphi, \text{BinMax}_{\text{2Words}}-\varphi, \text{EqualSisters}_{\text{Adj}}-\varphi, \text{EqualSisters}_{\text{Adj}}^{2}-\varphi, \text{AccentAsHead}-\varphi, \text{NoLapseL} \} \]
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:

\[(14) \text{amerika-}\text{no} \quad \text{tomodachi-}\text{no} \quad \text{pasokon} \]

\[
\text{America-GEN} \quad \text{friend-GEN} \quad \text{computer}
\]

'my American friend's computer'
More options: Clitics

- **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 $\omega$
  - 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 $\omega$ or $\varphi$.
- *Troncamento*: deletion of word-final unstressed mid vowels after sonorants; sensitive to $\varphi$-boundaries

(15)

\[ \begin{align*}
    \sigma & \varphi \\
    \text{per} & \omega \omega \\
    \text{potere} & \text{capire}
\end{align*} \]

- How can we represent this in SPOT?  
[main/tutorials/AMP2018_italian_clitic_example.html](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

- 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

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


