Michael Becker, michael@linguist.umass.edu
University of Massachusetts, Amherst
January 5-8 2006
80th LSA Meeting, Albuquerque NM

CCamelOT - an implementation of OT-CC's GEN and EVAL in Perl*

Highlights:

- CCamelOT is an open-source web-based program that takes an input and a constraint ranking, and finds the output using OT-CC.
- OT-CC ("OT with Candidate Chains", McCarthy forthcoming) is a theory of phonology with finite candidate sets, so CCamelOT can produce complete candidate sets and find the outputs in them.
- CCamelOT's interface includes ready-to-use phonological building blocks and constraints, making it a valuable tool for researchers and instructors in OT.

Roadmap:

- The theoretical underpinnings of CCamelOT
- CCamelOT as a transparent model of phonology
- Using CCamelOT in OT research and teaching

1 Theoretical underpinnings of CCamelOT

In Classic OT (Prince & Smolensky 2004), GEN creates an infinite number of candidates, so it would take infinitely long to generate a full candidate set.

If we want to make a model of GEN, we need to find a way to generate only the interesting candidates.

1.1 Harmonic improvement

Moreton (2004): In OT, the winner is either completely faithful to the input, or less marked than the input.

(1) Given an input /A/ and an OT grammar, the output is either [A] or some [B] that is less marked than [A].

(2) [A] is the **the fully faithful candidate**, the most harmonic candidate that incurs no faithfulness violations.

*I am grateful to Kathryn Flack, Shigeto Kawahara, John McCarthy and Matt Wolf for wonderfully helpful discussions. I also got great feedback from Tim Beechey, Peter Jurgec, Mike Key, John Kingston, Joe Pater, Chris Potts, and Anne-Michelle Tessier. There are no remaining errors.

(3) The output is the most harmonic candidate. If the output is different from the fully faithful candidate ⟶ the output is less faithful and less marked than the fully faithful candidate.

(4)

/pat/	*CODA	DEP
a. pat	*	
☞ b. patə		*

1.2 OT-CC, Optimality Theory with Candidate Chains

OT-CC (McCarthy forthcoming) is a theory of phonology that builds on Moreton's "harmonic improvement", and adds the idea that improving the input is done one step at a time.

In this theory, a candidate is not just a surface form, it is a **chain** of forms that starts with the input and derives the output one step at a time.

(5) Given an input /A/ and a surface form [B], the winner is a **candidate chain** such that:

- The first link in the chain is [A]
- The last link in the chain in [B]
- Every link in the chain is more harmonic than the preceding link
- Every link in the chain adds exactly one basic phonological operation = one Localized Unfaithful Mapping (LUM)

(6) Example: given the input /pat/ and the grammar *CODA » DEP, the chain <pat, patə> is the winner, since

- [pat] is the fully faithful candidate
- [patə] is more harmonic than [pat] given the grammar
- [pat] → [patə] adds exactly one LUM: epenthesis of a schwa

(7) Given the input /pat/ and the grammar *CODA » DEP, *VTV

- <pat, patə, padə> is the winner
- *<pat, patə> is a possible chain (but not the winner)
- **<pat, padə> is not (epenthesis and voicing done at once)[1]
- **<pat, pad, padə> is not (not harmonically improving)

The basic phonological operations include epenthesis of one segment, deletion of one segment, and change of one feature. The operations derive the input from the output one step at time.

[1] One star marks a losing chain, two stars mark an ill-formed chain

1.3 Finite candidate sets

OT-CC candidate sets are finite if we are know that:

- Each chain is finitely long
- The number of chains is finite

(8) What are possible ways to make a chain infinitely long?

- Unbounded epenthesis
- Repeating forms in the chain

If these things don't happen, all chains are finitely long.

(9) Chains can't have unbounded epenthesis: **<A, AA, AAA, AAAA, ...> thanks to the nature of markedness and faithfulness[2]:

- Markedness constraints can't cause unbounded epenthesis, because they only look at the output. They can only demand epenthesis up to a certain size (e.g. minimal word).
- Faithfulness constraints demand input-output *identity*, so they can't call for epenthesis. [3]

(10) Forms can't repeat in a chain: **<A, B, A, B, A, B, ...>

If A follows B in a chain, then A is more harmonic than B
If B follows A in a chain, then B is more harmonic than A
It's impossible for A to be more and less harmonic than B

(11) The number of chains is finite because the number of operations is finite.

Starting with the trivial one-link chain, a finite number of two-link chains will be created. From those, a finite number of three-link chains will be created, etc., until chains can't get any longer.

2 Finite derivations with CCamelOT

To find CCamelOT, just Google "CCamelOT", or go here:
http://wwwx.oit.umass.edu/~linguist/CCamelOT/
Start with the *guided tour* to get an idea of how CCamelOT works.

[2] In terms of Moreton (2004), the grammar is "eventually idempotent".

[3] Anti-faithfulness (Alderete 2001) constraints can call for epenthesis, but they are always satisfied by a single operation. They can demand epenthesis of no more than one phonological unit relative to the input.

(12) CCamelOT uses OT-CC principles to run an input through a grammar and find the output.

(13) The derivation takes the input and applies phonological operations to it one at a time, to find all forms more harmonic than the input. The number of these forms is guaranteed to be finite, and in practice it is often very small.

Try the input /pat/ with this grammar:
MAX » *APPENDIX » *C/NUC » *CODA » DEP
You will find this grammar in the "Ranking" page, under the "Open" tab.

(14) The derivation starts with a trivial one-link chain, which contains the fully faithful candidate (= the input as syllabified by the grammar).

(15) The fully faithful candidate goes through one round of application of the phonological operations: deletion of a single segment, epenthesis of a single segment (ə or ʔ), change of one feature (+ to – or vice versa).
Whenever the output of an operation is more harmonic than the fully faithful candidate, a two-link chain is created.

Add *VTV to your grammar to see inter-vocalic voicing.

(16) The final links in the two-links chains are passed through one round of phonological operations, to produce three-link chains. And so on, until no more chains can be created.

(17)

/pat/	MAX	*CODA	DEP	*VTV
a. <pat>		*		
b. <pat, patə>			*	*
☞ c. <pat, patə, padə>			*	
d. **<pat, pa>	*			

3 CCamelOT phonological representations

CCamelOT has the kind of representations that we normally assume, so it can be a useful tool for testing hypotheses about the ways phonological structure works.

(18) CCamelOT's linguistic forms are represented using segmental, prosodic and morphological structure.

- Segments have indices used for computing Correspondence relations (McCarthy & Prince 1995)

- Segments are associated with moras and syllables
- Phonemes are bundles of features
- Forms have morphological structure (root vs. affix).

This lays the foundations for phonological representations that are essentially identical to representations typically assumed in generative phonology.

To customize the linguistic representations, download CCamelOT's source: http://wwwx.oit.umass.edu/~linguist/CCamelOT/needCON.cgi

- A table of phonemes and their features is kept in a separate file, so they are easy to change (see the Help page).
- The infrastructure is there for extending the existing prosodic structures to include feet, prosodic words, etc.
- Implementing autosegmental representations or finer morphological structure would demand more thinking.

4 CCamelOT constraints

CCamelOT provides a transparent model of OT constraints, so CCamelOT constraints can teach us something about the OT constraints that we work with.

Like OT, CCamelOT has two kinds of constraints:

- Markedness: a function from outputs to integers (number of violations).
- Faithfulness: a function from <input, output> pairs to integers.

(19) Markedness: ONSET

Assign one violation mark for every syllable whose first segment is a vowel

In pseudo-code:

```
1  violations = 0
2  for each (i in output.syllables) {
3     if (i.phonemes[0].consonantal = "V") {
4        violations++
5     }
6  }
7  return violations
```

1. Start with zero violations
2. Go over the syllables of the output
3. Look at the consonantal feature of the first phoneme in that syllable - is it a vowel?
4. If so, add one to the count of violations
7. Return the number of ONSET violations found

(20) Faithfulness: IDENT(voice)

Assign one violation mark for every segment of the output **i** that has an input correspondent **j**, and **i**'s value for [voice] is different from **j**'s.

In pseudo-code:

```
1  violations = 0
2  for each (i in output.indices) {
3     for each (j in input.indices) {
4        if (i = j) {
5           if (i.phoneme.voice != j.phoneme.voice) {
6              violations++
7           }
8        }
9     }
10 }
11 return violations
```

1. Start with zero violations
2. Go over the output's segment indices
3. For each output segment, go over the input's segment indices
4. Is there is a corresponding input-output pair of segments?
5. If so, is [voice] in the input different from [voice] in the output?
6. If so, add one to the count of violations
11. Return the number of IDENT(voice) violations found

Want to add your own constraints to CCamelOT?
To download CCamelOT's source, go to:
http://wwwx.oit.umass.edu/~linguist/CCamelOT/needCON.cgi

5 Using CCamelOT in research

(21) CCamelOT can find candidates you haven't thought of, and help you make your analyses more explicit. E.g.:

- We tend to think that the ranking of ONSET and *CODA has no effect on the analysis.
- CCamelOT shows that given a low ranking *C/NUC, the ranking of ONSET and *CODA is crucial.

Try the input /ta:n$_{\mu}$/ with this grammar:
*APPENDIX » MAX » DEP » *CODA » ONSET » *3μ » IDENT(length) » *C/NUC
Then promote ONSET over *CODA.
You will find this grammar in the "Ranking" page, under the "Open" tab.

(22) When the number of constraints gets too big for human processing, CCamelOT can help you find crucial rankings.

(23) If you download CCamelOT and build constraints that you need, that can help you think more carefully about the logic of your constraints.

- How do ALIGN constraints evaluate forms? How should deleted segments and epenthetic segments affect alignment?

Bruce Hayes on the pros and cons of using computer simulations in OT: http://www.linguistics.ucla.edu/people/hayes/otsoft/why.htm

6 Using CCamelOT in teaching

CCamelOT can be useful not only in teaching OT-CC, but also in teaching Classic OT. Here are some Classic OT concepts that your students can learn with CCamelOT:

(24) Ranking of universal violable constraints

- Factorial typology
- Crucial vs. non-crucial rankings

(25) General/specific relations (stringency relations) between constraints

- *CODA / *COMPLEXCODA
- IDENT(voice) / IDENT$_{\text{ROOT}}$(voice)

(26) Harmonic improvement

- Learning to think in terms of relative harmony
- Harmonic bounding

For students with basic computer skills, CCamelOT offers a ready-to-use interface. Students with some background in programming can download the source and add new constraints.

7 Summary

- CCamelOT is a program that runs an input through a grammar and finds the output, based on principles of OT-CC.
- CCamelOT models the linguistic representations and constraints that OT phonologists assume. This makes CCamelOT a useful tool for testing hypotheses.
- CCamelOT's interface includes ready-to-use phonological building blocks and constraints, making it a valuable tool for researchers and instructors in OT.

  [Try the guided tour!]

- CCamelOT's open code is an invitation to further our ability to model phonological theory computationally.

References

Alderete, John (2001). Dominance effects as trans-derivational anti-faithfulness. *Phonology* **18**. 201–253.

McCarthy, John J. (forthcoming). *Hidden Generalizations: Phonological Opacity in Optimality Theory*. London: Equinox Publishing Company.

McCarthy, John J. & Alan Prince (1995). Faithfulness and reduplicative identity. In Jill N. Beckman, Laura Walsh & Suzanne Urbanczyk (eds.) *Papers in Optimality Theory*, University of Massachusetts Occasional Papers 18, University of Massachusetts, Amherst: GLSA. 249–384.

Moreton, Elliott (2004). Non-computable functions in optimality theory. In John J. McCarthy (ed.) *Optimality Theory in Phonology*, Blackwell. 141–163.

Prince, Alan & Paul Smolensky (2004). *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell.