Notes on Primitive Optimality Theory (pt 1)

Daniel M. Albro

May 11, 1999

1 Review of last time

Last time was a combination of ideas from (Ellison 1994) and (Eisner 1997a), and introduced most of the computational issues involved in implementing OT with finite-state methods:

- OT consists of GEN — a device which maps an input to an infinite set of candidate outputs, plus CON — an ordered sequence of constraints $c : \Sigma^* \rightarrow \mathcal{N}$.
- The evaluation function/winnowing process works as follows. For each $c_i \in$ CON, the corresponding filter function is $F_i(\text{CANDS}) = \{w_j | \forall w_k \in \text{CANDS}, c_i(w_j) \leq c_i(w_k)\}$.
- Derivation in OT is defined as follows: given CON $= \langle c_1, c_2, \ldots, c_n \rangle$, where c_1 outranks c_2, etc., OUTPUT(INPUT) $= F_n(F_{n-1}(\ldots F_2(F_1(\text{GEN}(\text{INPUT})))\ldots))$.
- GEN(INPUT) can be modeled as an FSM which accepts all outputs that could theoretically be derived from the input. More about this later...
- The constraint function c_i can be modeled as a Weighted Finite State Machine (WFSM) $W(c_i)$ whose domain is Σ^* and which is designed such that the summed weight of any successful path equals the number of constraint violations c_i would assign to the corresponding candidate.
- $F_i(\text{CANDS})$ then equals $BSP(W(c_i) \sqcap M(\text{CANDS}))$, where $W(c_i) \sqcap M(\text{CANDS})$ is a WFSM that accepts just the elements of CANDS and is such that the summed weight of any successful path in it is equal to the number of constraint violations c_i would assign to the candidate. *BSP* is the Best Successful Paths algorithm (from (Dijkstra 1959), modified as in (Albro 1998)).

2 Applications to Phonology

Several questions are left unanswered by the above, including these:

1. How do we represent inputs and candidates as FSMs?

2. How do we make construct the weighted finite state machine $W(c)$ for any particular constraint c that we want to represent?

Primitive Optimality Theory (OTP) (Eisner 1997b) is an attempt to answer these questions. To some extent, so was One-Level Phonology (Bird and Ellison 1994), and we should think about its answers as we look at the answers of OTP.

3 Representations in OTP

Some review of representations...

3.1 Phonemic/Alphabetic Representations

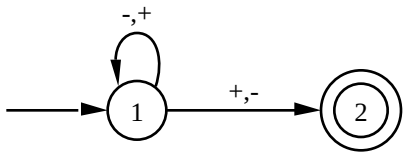
The "standard" representation for phonology in most finite-state accounts is the phoneme (to be charitable, in lots of real finite-state phonology, it's the letter, since most computational phonology uses orthography, I think). There are certain benefits to this approach — the alphabets are not too complex, and it's almost possible to look at the machines and see what they're doing. However, the use of phonemic or phonetic symbols can obscure certain regularities, especially autosegmental-type things, and we still have to worry about prosody. Anyway, it's pretty clear how to translate phonemic representations to FSMs, as long as you don't worry too much about tones and syllables and things like that.

3.2 SPE-Style Feature Matrices

In the SPE paradigm, syllables and things like that are discarded, as much as possible, and words and morphemes are mainly noted by special boundary symbols. Thus, everything is a "segment." Other than the boundary elements, segments are vectors of binary features, *e.g.:*

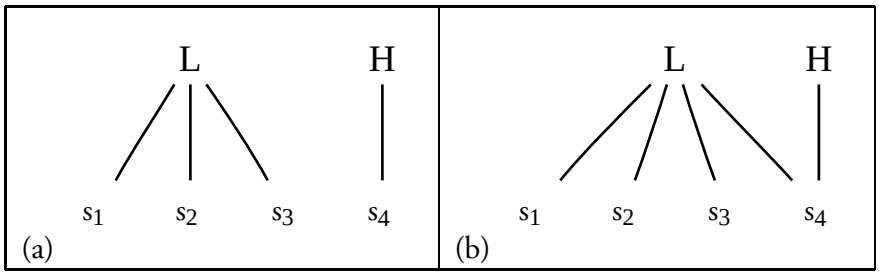
$$\begin{bmatrix} +\text{voice} \\ -\text{tense} \\ -\text{round} \end{bmatrix}$$

A possible (fairly direct) representation for these things in finite state machines is to label each edge of a finite state automaton with an ordered tuple of elements from the alphabet $\{+, -\}$. Thus,



is equivalent to $\begin{bmatrix} -\text{voice} \\ +\text{tense} \end{bmatrix}^* \begin{bmatrix} +\text{voice} \\ -\text{tense} \end{bmatrix}$, assuming just those two features exist.

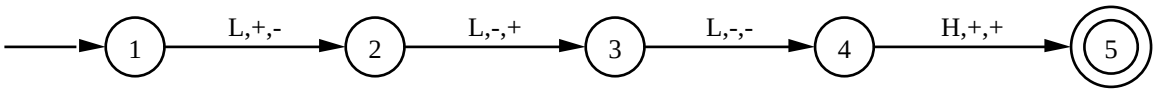
However, there are (at least) two problems with this — what about the existence of suprasegmentals, autosegments, and so forth, which seem to apply to (overlap with) sequences of segments? For example:



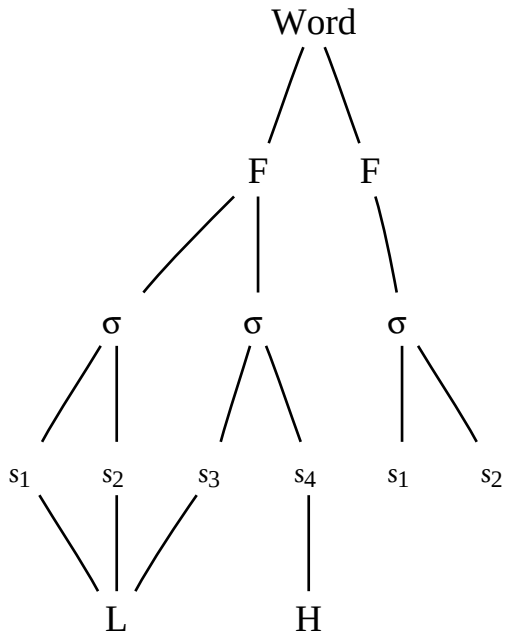
Here let's assume the following:

$$s_1 = \begin{bmatrix} +\text{voice} \\ -\text{tense} \end{bmatrix} \quad s_2 = \begin{bmatrix} -\text{voice} \\ +\text{tense} \end{bmatrix} \quad s_3 = \begin{bmatrix} -\text{voice} \\ -\text{tense} \end{bmatrix} \quad s_4 = \begin{bmatrix} +\text{voice} \\ +\text{tense} \end{bmatrix}$$

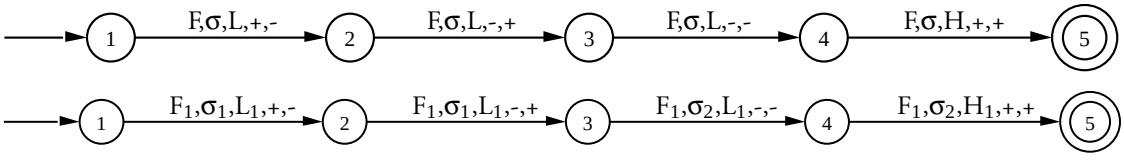
For (a) we could use the following representation, in keeping with the SPE stuff above:



but notice that we have three L's, where we might want to think of only one L actually being there (why?). We also have the problem of prosody:



What representation can cover this case?



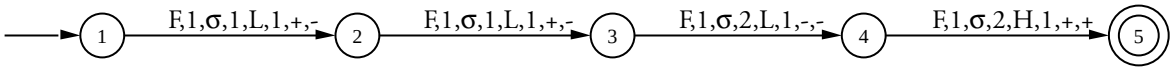
(1)

(2)

Problems with (1): Are there four syllables? How many are there? Is s_1 the same σ as s_3? Note that the representation in (1) is much like the One-Level Phonology representation.

Problems with (2): How do we really represent this with FSMs? Notice it requires a potentially infinite alphabet, and makes it tricky to identify in constraints the fact that σ_1 and σ_2 are both syllables.

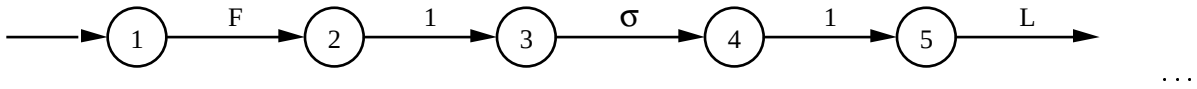
3.2.1 Potential fix:



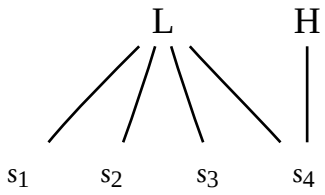
(3)

Problems with (3): We now have a finite alphabet, but the number of tuples in any given label is now unbounded and writing a constraint that says "in the same syllable" or "in a different syllable" will be tricky.

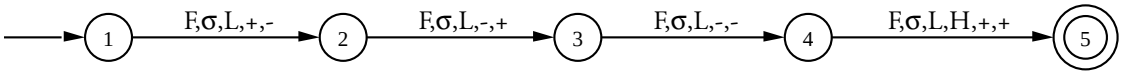
3.2.2 Alternative formulation of (3):



This fixes some of the problems, but others remain. Now back to the tones. What about the second representation?

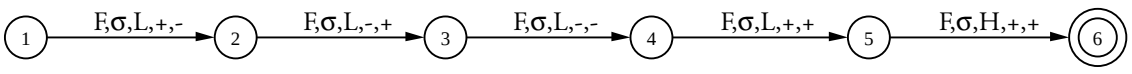


A suggestion:



(4)

This could work, but notice it means we have to have different length tuples (unbounded) and the tuples have to be interpreted as both representing simultaneity (+,+) and temporal precedence (L,H). Another possibility:

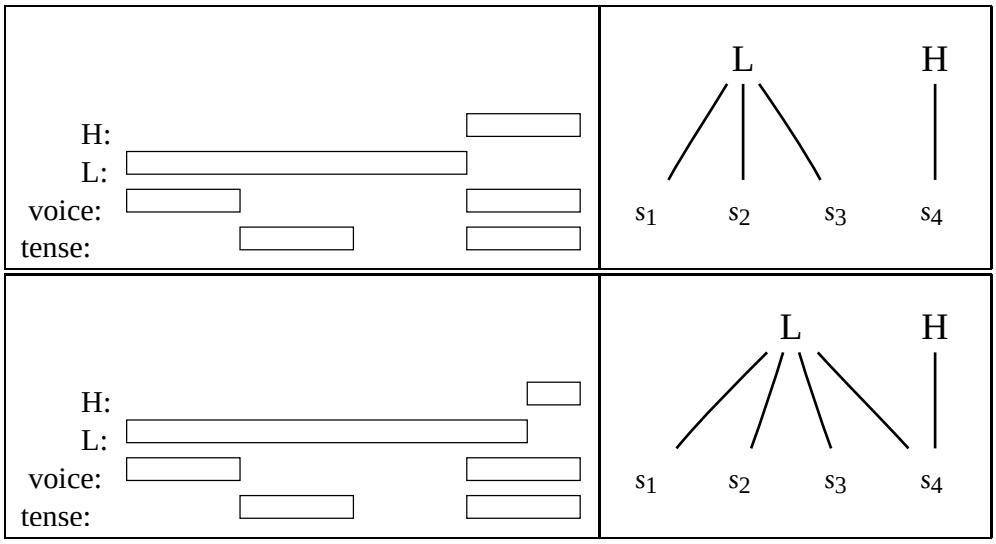


(5)

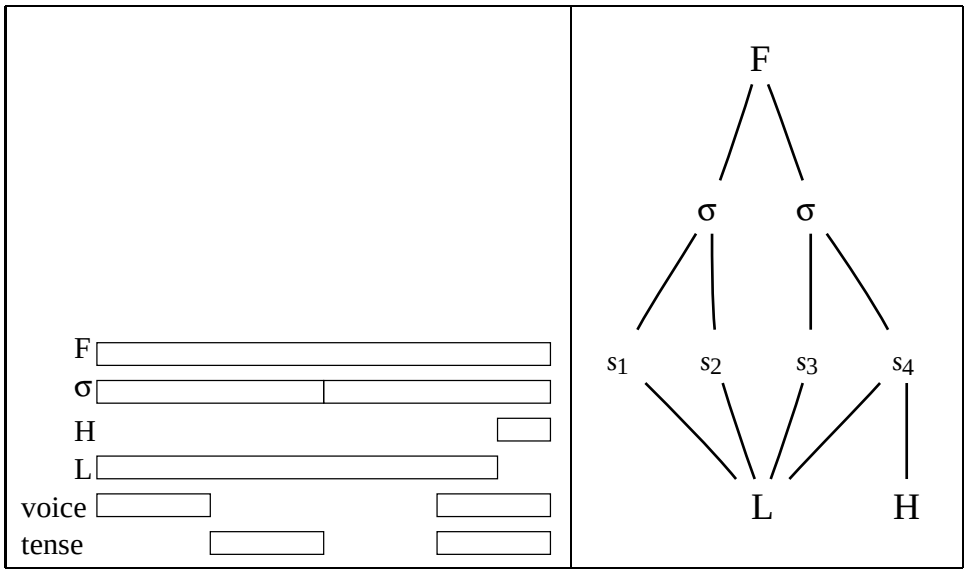
Here we have the old problem. Is +,+ two segments or just one?

3.3 Gestural Scores

Proposed solution — something like a gestural score, so:



What's the relationship between the two representations? The autosegmental-type diagrams show two relations: precedence by order on tiers, and temporal overlap by association lines. The gestural score shows precedence by order on tiers, as before, and temporal overlap by, well, overlap, that is, mutual presence at some particular point in time. What about the prosody stuff, though? Well, prosody adds a new relation — constituency (containment). That is, a foot is made up of (contains) two syllables, etc. The proposal here is to conflate constituency with temporal overlap (more specifically, temporal containment, in this case). We also have to be able to note directly abutting intervals:

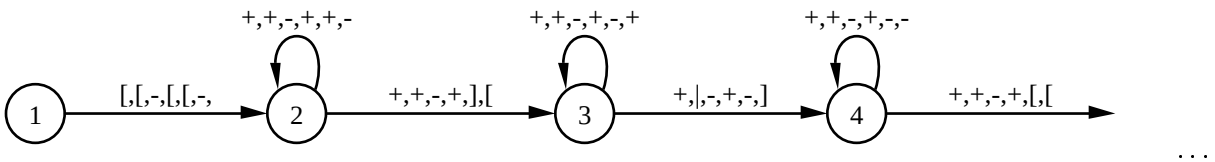


Note that the use of gestural scores enforces the anti-line-crossing convention, unlike autosegmental diagrams or indices.

Now, how do we encode this in an FSM? We want to be able to distinguish the following things: interiors, exteriors, constituent edges, and abutting boundaries. In this regard, let's look at One-Level Phonology again. Here we have things like $(+\text{voice} \cap -\text{tense})^+$— these have the interior/exterior distinction (sort of), but not edges. Note also a possible convention — if something is repeated, it's still the same thing. We use that here. The OTP representation looks at the gestural score as being a timeline and separates it into (Isaac Newton's term, I think, or maybe Leibnitz's...) infinitessimals. That is, dts — infinitely small slices of time. Each of these is represented in the FSM:

```
          *   *   *   *     *
    F [ + + + + + + + + + ]
    σ [ + + + | + + + + + ]
    H - - - - - - - - [ + ]
    L [ + + + + + + + ] - -
voice [ + ] - - - [ + + + ]
tense - - [ + ] - [ + + + ]
```

so here we have a representation that notes the boundaries, interiors, and exteriors. In an FSM, each dt can be represented as a tuple:



Note that the alphabet for this FSA is thus $\{-, +, [,], |\}$, if viewed as a multi-level transducer, or the set of 6-tuples of elements of the set, if viewed as a simple acceptor.

3.4 Input-Output Correspondence and Representations

Okay, so far so good. Now what about Input-Output things? Here we basically use the transducer idea, and here we get into some trouble. Maybe someone will be able to come up with a better way to deal with input and output than what I'm going to present here... As a digression, one way to deal with inputs and outputs would be to represent the surface form *only* (like in One-Level Phonology) and just start out with something that said what all the possible surface forms might be, but this would put the responsibility of faithfulness constraints into GEN, basically. In OTP, it's done like this (assume Kleene stars over the interiors):

```
voice: [ + ] - -
tense: - - [ + ]
voice: [ + | + ]
tense: [ + ] - -
```

So this is like a transducer with input-output relationships, or just an acceptor of output forms that contain their input. Note that it respects the Same-Length property, a fact that we're going to try to hold onto. At this point we can see the major flaw of the OTP representation, a flaw for which I don't currently have any good remedy — in addition to conflating constituency and overlap relations, it also conflates these relations with correspondence relations. In constituency and overlap, a strong case may be made that crossing dependencies aren't possible, but with correspondence relations the case isn't so strong. The existence of fairly long-distance metathesis shows that some crossing dependencies seem to be necessary. These cases seem quite rare, though.

Homework (not really...): Think of a decent representation that can encode these sorts of things in finite-state machines, but still allows you to manipulate and represent crossing dependencies for input-output.

3.4.1 Insertion and Deletion

This discussion of representations and the Same-Length property now brings up the question of how to deal with insertion and deletion. Eisner's proposal is as follows:

- Epenthesis: CC $\Rightarrow$ CVC:

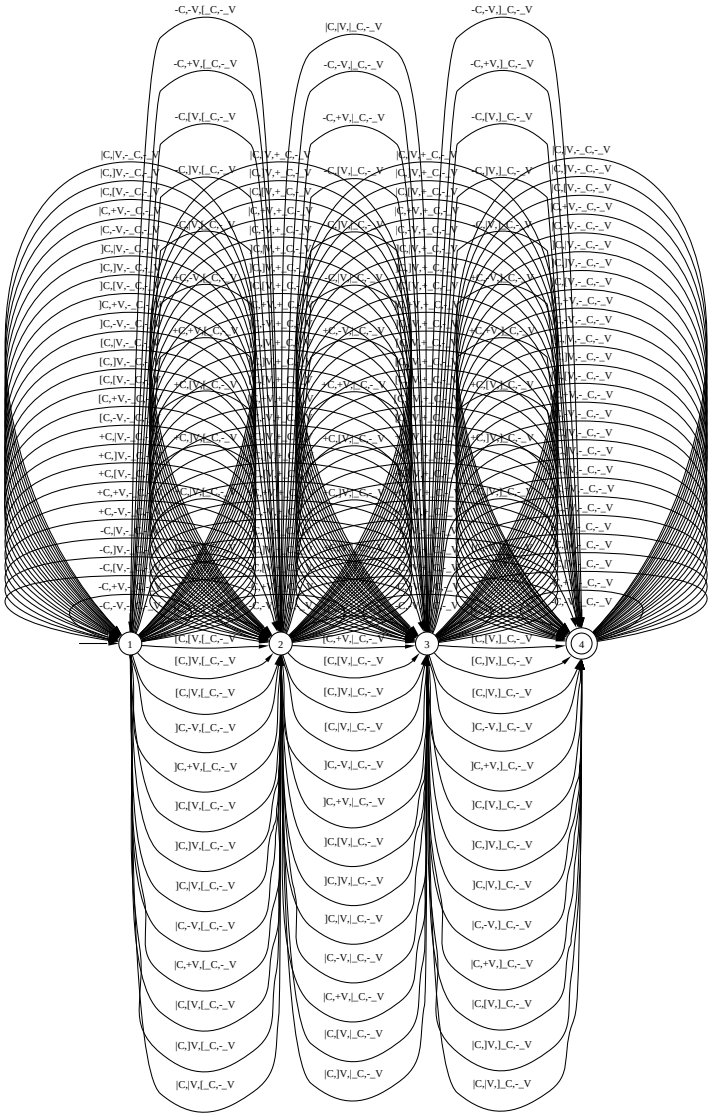
```
V: - - [ + ] - -
C: [ + ] - [ + ]
C: [ + ] - [ + ]
V: - - - - - - -
```

- Syncope: <u>CVC</u> ⇒ CC:

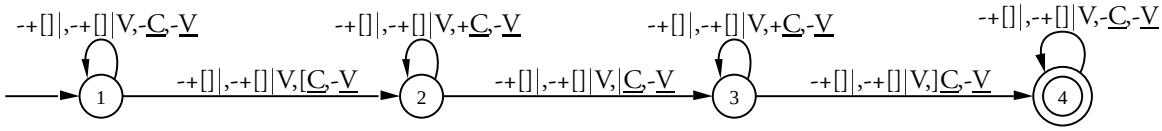
```
V: - - - - - -
C: [ + | + + ]
C: [ + | + + ]
V: - - [ ] - -
```

We will discuss the flaws in this proposal next time, but for now let's note its virtue: it gets around the non-SL problem. What does an input have to look like to generate this?

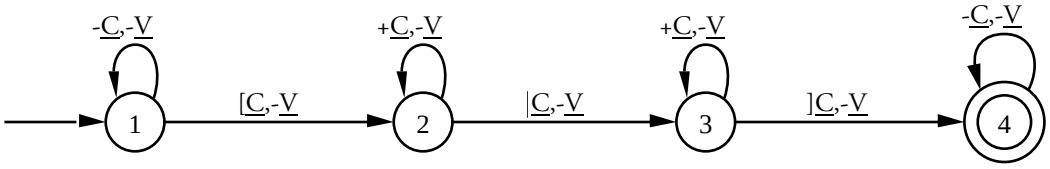
First, we have to think about what an input FSM might look like *period*. What does it do? an input FSM says "I don't (much) care about what the output is, but the input looks like this." Thus, for input <u>CC</u>, (not allowing for syncope or epenthesis for now), we have:


1
2
3
4

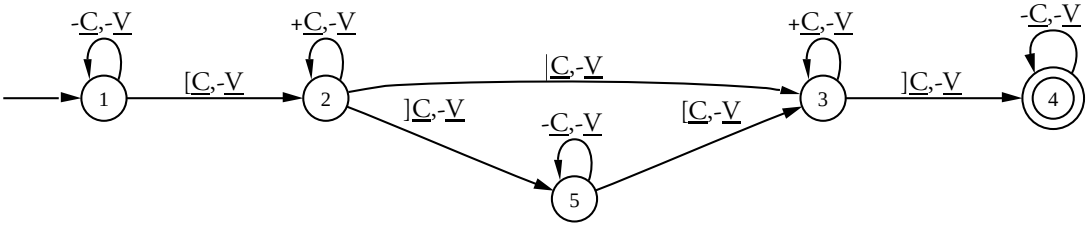
This is a mess. As an abbreviatory convention, we write "-+[]|" for "-, +, [,], or |", so an edge labeled "-+[]|C,..." is equivalent to five edges labeled "-C,...", "+C,...", "[C,...", "]C,...", "|C,..." Now we get:



This is still sort of a mess. A further abbreviatory convention (not in the implementation, just in print) is that "-+[]|" tiers aren't written:



This representation allows anything on the output tiers and arbitrary insertions at the outsides. It also allows the two C's to be squeezed into oblivion (the syncope picture). The only thing it doesn't allow is epenthesis. Thus, we have to explicitly allow the "|" edge to be broken up:



This will allow epenthesis. As mentioned before, it has shortcomings, but we will get to that later.

4 Constraints

It's too much to ask, perhaps, that constraints be written directly as WFSMs. Instead, something like the two-level rule notation or the (rather amorphous) one-level phonology notation is needed. Eisner claims that most reasonable constraints have the formulation $\forall x, \exists y$ or $\forall x, \nexists y$ or just $\nexists y$. I think this is based mainly on Generalized Alignment and Correspondence Theory. In GA, we have the family Align(thing1, Edge1, thing2, Edge2), which says something like for every Edge1 of a thing1, there

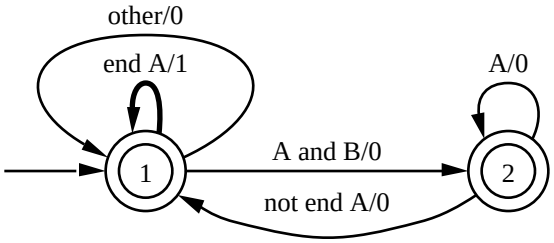
must exist an Edge2 of thing2 such that Edge1 is aligned with Edge2, and in Correspondence Theory constraints say "for every so and so in such and such level of representation, there must exist a corresponding thingee." For these sorts of things, OTP provides a family of constraints called "implication," which say, basically, "for every x, there exists a y such that x and y overlap on the gestural score. Here x is a conjunction with one or more elements, of which the elements can be interiors, left edges, or right edges, thus "C", "[C", or "]C". Here "C" refers to any part or the entirety of the interior of any "C" constituent, "[C" refers to the left edge (thus it matches "[" and "|"), and "]C" refers to the right edge. Similarly, y is a disjunction with one or more elements, of which the elements can be interiors, left edge, or right edges, as well. See (Eisner 1997b) for lots of examples of this. The other basic kind of constraint is the "*" constraint, where we say something isn't permitted. In OTP this is called a "clash" constraint, for some reason, and is noted by the symbol $\perp$, don't ask me why... These basically take a single conjunction of the element types given above, and forbid mutual coöccurrence.

4.1 Implementation

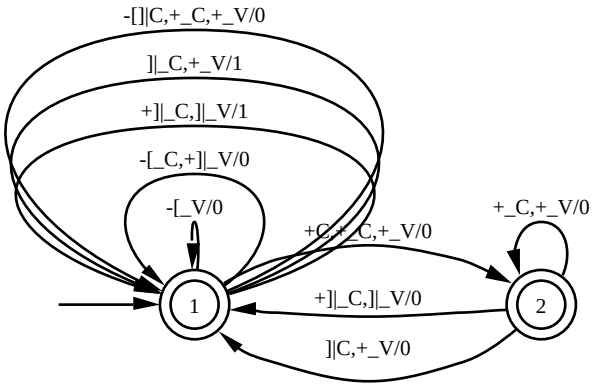
Notation: implication is $A \rightarrow B$, where A and B are lists of tier descriptors (interior, left edge, or right edge, of some constituent type), A is conceived as a conjunction, B is conceived as a disjunction. Clash is $A_1 \perp A_2$ where A_1 and A_2 are just parts of a single conjunction (might as well say $\perp A$, the infix notation is for convenience). The different element types are implemented as +x, [|x, or]|x.

The cases are as follows (it will get more complicated later):

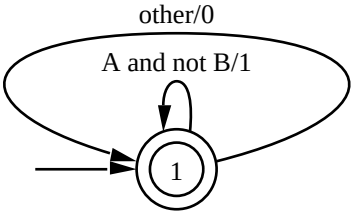
Implication where A is made up entirely of interiors Here we want to say that for every instance of an interval in which all the elements of A are present at the same time, there must be some member of B present during that interval, else a weighted edge must be traversed. Thus:



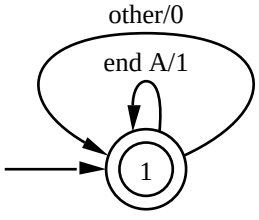
An example is "C and V $\rightarrow$ C":



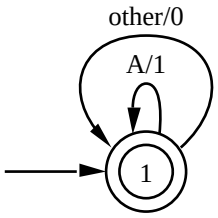
Implication where *A* has edges in it Here we want to say that any edge on which all of *A* appears, but not any of *B*, should have a weight:



Clash where *A* is all intervals Here we want to say that every instance of an interval in which all elements of *A* are simultaneously present should go through a single weighted edge:



Clash where *A* has edges Here we want to say that every edge where all of *A* is present should be weighted:



5 Assignment

Read (Eisner 1997b) and the UCLA OTP user's manual (optional), which can be found in the "doc" directory of OTP (I'll put a recent version on the ftp site, there's an old one on my web page). Write a constraint system using OTP constraints for a language in which there's just one tier — "S" (in input and output variants) — and the output is completely faithful to the input. Extra credit: run the thing under OTP and make sure you were right. More extra credit: write the corresponding WFSMs.

References

ALBRO, DANIEL M., 1998. Evaluation, implementation, and extension of Primitive Optimality Theory. Master's thesis, UCLA.

BIRD, STEVEN, and T. MARK ELLISON. 1994. One-level phonolgy: Autosegmental representations and rules as finite automata. *Computational Linguistics* 20.55–90.

DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1.269–271.

EISNER, JASON. 1997a. Efficient generation in primitive Optimality Theory. In *Proceedings of the ACL*.

——, 1997b. What constraints should OT allow? Handout for talk at LSA, Chicago.

ELLISON, T. MARK. 1994. Phonological derivation in Optimality Theory. In *Coling*, volume II, 1007–1013, Kyoto, Japan.