Finite-state Scripting
and the Computational Curriculum

Michael Hammond

U. of Arizona

Jan. 8, 2012
Outline

Why computational literacy generally

What is finite-state scripting

Toolkits

Examples

Why finite-state scripting is useful
The field has changed

Why should linguists be computationally literate?

- Lots of large data sets available electronically.
- Inexpensive powerful personal computers.
- Mathematically intensive theories gaining in popularity.
What we do at Arizona

- We have theoretical and practical computational courses at all levels of our curriculum.
- We have lots of computationally oriented faculty.
- We have a specific additional MS degree in Human Language Technology (HLT).
- Students in our PhD program can earn either an MA or an HLT MS degree “on the way” to their PhD in Linguistics.
- Computational skills are also introduced in traditional linguistics courses.
A specific example

- We have a course “Formal foundations of linguistic theory”
- It covers formal language theory, logic, set theory, etc.
- I’ve started using a finite-state toolkit to teach some of these concepts.
What is finite-state scripting?

- Finite-state scripting implements *regular languages* and *regular relations*.
- A regular language (or regular expression) can only be defined in terms of a finite alphabet, concatenation, disjunction, and repetition. For example:
  - $ab = \{ab\}$,
  - $a|b = \{a, b\}$,
  - $a^* = \{\epsilon, a, aa, aaa, \ldots\}$,
  - $[a^*|b]b = \{b, bb, ab, aab, aaab, \ldots\}$.

It’s been argued that the set of words in a natural language can be defined with just this much computational power.
A mathematically imprecise characterization is that these are rules or mappings.

Any two regular expressions can be paired to make a regular relation.

All the operations available for regular languages can be used in regular relations.

For example:
- $a : b$ maps $a$ to $b$.
- $[ab] : c$ maps $ab$ to $c$.
- $[a : b][a : c]$ maps $a$ to $b$ or maps $a$ to $c$.

It’s been argued that all of phonology and all of morphology can be treated with just this much computational power.
Free implementations

There are a number of implementations of finite state technology:

- xfst [http://www.cis.upenn.edu/~cis639/docs/xfst.html](http://www.cis.upenn.edu/~cis639/docs/xfst.html)
- fsp [http://www.u.arizona.edu/~hammond/flbi1.02.tar.gz](http://www.u.arizona.edu/~hammond/flbi1.02.tar.gz)
- fsa [http://odur.let.rug.nl/~vannoord/Fsa/](http://odur.let.rug.nl/~vannoord/Fsa/)
- etc.
Why I use foma:

- free;
- cross-platform;
- fast, efficient, complete;
- written by an Arizona alumnus: Mans Hulden.
Example: Welsh syllable structure

- define Stops [p|t|c|b|d|g];
- define ...;
- define Onset [(Stops|Frics) (w) (Nasals) (w) (Liquids) (i)] [(Nasals|r) h (Liquids)];
- define Nuc [(High) [High|Mid] (High)][(High) [Mid|Low] ([High|e|o]));
- define Coda (Liquids) (Nasals) [Stops|Frics]^<5 (w|y|r|l|n) ([r|s]);
- define Syllable [Onset Nuc Coda];

(Parentheses indicate an optional element; ^<5 means less than five instances of the preceding element.)
Why use finite state scripting in this course?

- To better understand formal language theory
- To better understand logic and set theory too
- To better understand linguistic theory: phonology, morphology, syntax
- As a practical tool for language documentation
Desiderata

- It helps turn some of the drier material of formal foundations into something practical/useful.
- Its different modality makes formal foundations more accessible.
- Regardless of a student’s interest in formal language theory, it gives them a practical tool for language documentation.
Conclusion

- Computational literacy is more and more critical to linguists.
- Precisely what kind of literacy depends on a student’s interests and goals.
- We can connect computational literacy to difficult pedagogical domains and to practical concerns.
- As faculty, we need to be bold in trying out and exposing our students to new computational tools.