A Workbench for Logically Definable Stringsets

Dakotah Lambert, Andrew Dai, and James Rogers
Computer Science, Earlham College

Abstract

Our group has been developing a suite of computational tools that provide a workbench to analyze and manipulate axiomatic descriptions of phonotactic patterns. We have established a catalog of primitive logical constraints of known cognitive complexity that serve as the basis for these axiomatic descriptions. Given a set of axioms, our workbench is able to aid in determining whether or not the axioms are consistent, which of these axioms are logical consequences of the others and thus not strictly necessary, and what the overall complexity of the language is.
We model the phonological words in a language as strings of symbols representing syllables and their stress. Our alphabet contains a single symbol for each combination of weight and stress. This abstraction of syllables allows the comparison of languages with vastly different phonemic inventories. Further, logical formulae are divided into two categories: local and piecewise. Allowing conjunctions of both types of formulae, we derive axiomatic descriptions of languages that require fewer cognitive resources than using only local or only piecewise formulae. In fact, all but 4 of the 402 stress patterns in our corpus are testable or lower in complexity. Additionally, the four patterns of higher complexity share a common constraint: that syllables be of even length.

Logical reasoning about languages

Throughout our prior work, we have been using finite-state automata to represent descriptions of natural languages. This allows us to present theorems that are provable in a computational sense regarding these languages. In creating this workbench, we have created a set of tools to further this work, and to allow others to continue the path we have started. The operations implemented include the Boolean operations on automata (intersection, union, complement) the Kleene star, and concatenation. In addition to these basic operations, the workbench allows tests for acceptance of a given string, for isomorphism of two automata, and (with a bit of work) logical independence of primitive constraints.
A library of primitive constraints has been compiled in order to allow the construction of any as of yet unknown language by merely reusing primitive constraints found in already-known languages. Having developed a description of a language, a linguist will be able to use this workbench to determine which, if any, of the constraints listed are truly necessary to describe the language. The least complex set of logically independent primitive constraints that fully describes the language determines the complexity of the whole.

Terminology

When discussing languages as being built up of primitive constraints, the combinations are said to be conjuctive (*a* **and** *b*), or disjunctive (*a* **or** *b*). However, when discussing automata, the terms used are *intersection* and *union*, respectively. These may be used interchangably to some degree. A primitive constraint is a constraint on a pattern that cannot be further reduced to a conjuction of other constraints.

An Example — Cambodian

Primary stress falls on the final syllable

No final unstressed syllables

$$\neg\, \sigma \ltimes$$
```
noUssEnd = complement ((star xss) uss)
```

No final secondary-stressed syllables

$$\neg\, \grave{\sigma} \ltimes$$
```
noSssEnd = complement ((star xss) sss)
```

Nothing follows primary-stressed syllables

$$\neg\, \acute{\sigma}\, \overset{*}{\sigma}$$
```
noPssXss = complement (pss <> xss)
```

No empty words

$$\neg \rtimes \ltimes$$
```
noEmpty = contains xss
```

Secondary stress falls on all heavy syllables

No unstressed heavy syllables

$$\neg\, H$$
```
noUsH = complement (ush)
```

Light syllables only occur immediately following heavy syllables

No two consecutive light syllables

$$\neg\, \overset{*}{L}\, \overset{*}{L}$$
```
noXsLXsL = complement (xsl <> xsl)
```

No intial light syllables

$$\neg \rtimes \overset{*}{L}$$
```
noStartXsL = complement (xsl <> (star xss)
```

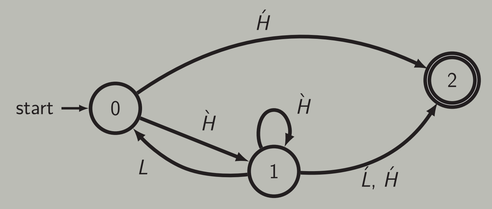
The language is described by the conjunction of these

The conjuction

$$\left(\neg\, \sigma \ltimes\right) \wedge \left(\neg\, \grave{\sigma} \ltimes\right) \wedge \left(\neg\, \acute{\sigma}\, \overset{*}{\sigma}\right) \wedge \left(\neg \rtimes \ltimes\right) \wedge \left(\neg\, H\right) \wedge \left(\neg\, \overset{*}{L}\, \overset{*}{L}\right) \wedge \left(\neg \rtimes \overset{*}{L}\right)$$
```
cambodian = noUssEnd /\\ noSssEnd /\\ noPssXss /\\ noEmpty /\\ noUsH /\\ noXsLXsL /\\ noStartXsL
```

The result is an automaton

A graphical depiction of the result



Testing the description

Using the workbench, we can check whether or not a given string is accepted by our description of the language. The string "*H̀HLH̀L*" should be in the language. On the other hand, "*H̀HĹĹ*" should not be. The workbench confirms this:

```
>> FSAt.accepts cambodian (symbolString "H` H` L H` L´")
True
>> FSAt.accepts cambodian (symbolString "H` H` L L´")
False
```

More interestingly, we can determine necessity of constraints. A constraint is a logical consequence of the description of the language if the conjunction of the description with the negation of the constraint is null. Two constraints that are appear in every known language are that every word contains some syllable of primary stress, and that no word contains two such syllables. Using the workbench:

`>> FSAt.null $ (not (contains pss) /\\ cambodian` and
`>> FSAt.null $ ((contains pss) <> (contains pss)) /\\ cambodian`
both return `True`. Both are consequences of our description.

References

- G. Bailey, M. Edlefsen, M. Visscher, D. Wellcome, and S. Wibel. Deciding strictly piecewise stringsets. In *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics (MCURCSM'09)*, 2009.
- J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.
- M. Edlefsen, D. Leeman, N. Myers, N. Smith, M. Visscher, and D. Wellcome. Deciding strictly local (SL) languages. In J. Breitenbucher, editor, *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics (MCURCSM'08)*, pages 66–73, 2008.
- C. C. Elgot. Decision problems of finite automata and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–51, 1961.
- M. Fero, D. Lambert, S. Wibel, and J. Rogers. Abstract categories of phonotactic constraints. In *National Conference on Undergraduate Research*, 2014.
- J. Heinz. Ud phonology lab stress pattern database. http://phonology.cogsci.udel.edu/dbs/stress/, 2012.
- R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press, Cambridge, MA, 1971.
- Y. T. Medvedev. On the class of events representable in a finite automaton. In E. F. Moore, editor, *Sequential Machines—Selected Papers*, pages 215–227. Addison-Wesley, 1964. Originally in Russian in Avtomaty (1956), pp. 385–401.
- J. Rogers. wMSO theories as grammar formalisms. *Theoretical Computer Science*, 293:291–320, 2003.
- J. Rogers, J. Heinz, G. Bailey, M. Edlefsen, M. Visscher, D. Wellcome, and S. Wibel. On languages piecewise testable in the strict sense. In C. Ebert, G. Jäger, and J. Michaelis, editors, *The Mathematics of Language: Revised Selected Papers from the 10th and 11th Biennial Conference on the Mathematics of Language*, volume 6149 of *LNCS/LNAI*, pages 255–265. FoLLI/Springer, 2010.
- J. Rogers, J. Heinz, M. Fero, J. Hurst, D. Lambert, and S. Wibel. Cognitive and sub-regular complexity. In G. Morrill and M.-J. Nederhof, editors, *Formal Grammar 2012*, volume 8036 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2012.
- W. Thomas. Classifying regular events in symbolic logic. *Journal of Computer and Systems Sciences*, 25:360–376, 1982.
- S. Wibel, M. Fero, J. Hurst, D. Lambert, and J. Rogers. Classifying relative complexity of factored stress patterns. In J. Heinz, H. van der Hulst, and R. Goedemans, editors, *Univ. of Delaware Conference on Stress and Accent*, 2012.

Anchors: Left ($\rtimes$), Right ($\ltimes$) Weights: Light (L), Heavy (H), Arbitrary (σ) Stress: None (σ), Secondary ($\grave{\sigma}$), Primary ($\acute{\sigma}$), Arbitrary ($\overset{*}{\sigma}$) Logic: Negation ($\neg$), Conjunction ($\wedge$)