University of Waterloo

CS 662: Formal Languages and Parsing

Winter 2020

Graduate Term Project

One-page project proposal due February 12 2020.

Written project due April 3 2020.

To obtain credit for CS 662, graduate students are required to complete a term project that involves exploring some facet of formal language theory or parsing. Some possible topics are listed in the course textbook at the end of each chapter. You can certainly choose a problem not in the the textbook or below, but please check with me first. If your favorite area is not represented, you might try asking me or your advisor for a suitable topic. The one constraint is that you should *not* do the project on a topic that you have already studied. Doing the project on a topic which you will, in the future, do more work on (such as a thesis or essay) is fine.

On February 12 2020 you should hand in (in class) a sheet of paper with your name and a brief description of which project you plan to do. If your project involves something other than individual research (see below), please provide a list of *at least three* references on your subject area that you plan to read.

Part of the goal of this project is for you to learn something about how to do research. The first step of your exploration is to learn what has already been done on the problem. In the textbook I sometimes suggest a paper or two to get started, but you should not be content with just this. Trace back the citations in the listed paper to find out about earlier work. You should use any source of information you can think of. Check your text and the other reference books on the lists. Use the library: the library's online catalogue, references such as Computing Reviews and Math Reviews (MathSciNet), Talk to people. Also try MathSciNet at

`https://mathscinet.ams.org/mathscinet/search.html`

and Citeseer at

`http://citeseer.ist.psu.edu/`

and DBLP at

`https://dblp.uni-trier.de`

.

How much information is enough? If your topic is well-studied and you only bother to look at one source (one paper or book) then you haven't done enough. (Even if the paper *claims* to have the ultimate solution, you should explore that for yourself.) On the other hand, if you collect a list of more than 10 papers, then you should probably narrow the field somehow, either by restricting the scope of the topic or the type of solution, or by focusing on the most recent or the most relevant work.

The second step of your exploration is then to make sense of the available results, judging which are most useful. Your final report (5–15 pages) should summarize the results you have found, and say what your conclusions are.

If you prefer a research-oriented approach, you may concentrate your efforts on trying to solve an open problem. Such an attempt at original work is NOT required, but I would like to encourage it, and I will mark such attempts based on the efforts, not the results. Attempting original work does not excuse you from doing a literature search, but you should spend less time on the search (just enough to be sure you are attempting something new). Some open problems are given in the course textbook.

Yet another possibility is to take one or more Wikipedia pages on topics related to formal languages and parsing and edit them, adding substantial new content. If you choose this option, you should print out the page before you begin editing it and then at the end, highlighting the changes you made. A list of suggested topics to work on is given at the end of this sheet. Be sure to provide a good reference list in your Wikipedia article, using the standard method to cite references in Wikipedia.

Plagiarism is a serious offense and will be dealt with accordingly. You must report on your topic *in your own words*. All use of sources must be cited in the text. Any verbatim quotes from reference works must be clearly indicated (for example, through indentation and citation). Copying entire sections word-for-word from published sources is *not* acceptable. Anything quoted must be *clearly* labeled as such, and must not exceed a paragraph or two. Copying proofs verbatim out of papers is *not* acceptable.

Don't cite secondary sources, such as Wikipedia.

You will be marked on synthesis, organization, and depth of understanding displayed (50%), clarity of presentation (33%), and the presentation itself (spelling, format, etc.) (17%). Be sure to proofread your report. Use a spell checker, such as Unix's `spell` to check your paper. If you are using TEX, you can say something like

```
detex foo.tex | spell | more
```

If you are using LaTeX (highly recommended), then be sure to look at http://recursed.blogspot.ca/2013/05/ten-common-latex-errors.html for a list of common errors.

Suggested Survey Topics

1. Find out about applications of the theory of formal languages to the study of natural languages, such as English. The following paper will get you started: S. M. Shieber, Evidence against the context-freeness of natural language, *Linguistics and Philosophy* **8** (1985), 333–343. Also see Gazdar, Klein, Pullum, and Sag, *Generalized Phrase Structure Grammar*, Harvard University Press, 1985.

2. Find out about L-systems and their applications to computer graphics. The following two books will be of interest:

P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag, 1990.

P. Prusinkiewicz and J. Hanan, *Lindenmayer Systems, Fractals, and Plants*, Vol. 79 of *Lecture Notes in Biomathematics*, Springer-Verlag, 1989.

3. Read more about words without repetitions. The following book

R. K. Guy, *Unsolved Problems in Number Theory*, Springer-Verlag, 1994, second edition

has an extensive bibliography in section E 21. But there has been a huge amount of work since then.

4. Find out more about applications of automata theory to game theory and economics ("bounded rationality"). You could start with the following papers: Ariel Rubinstein, Finite automata play the repeated prisoner's dilemma, *J. Economic Theory* **39** (1986), 83–96; B. G. Linster, Evolutionary stability in the infinitely repeated prisoners' dilemma played by two-state Moore machines, *Southern Economic J.* **58** (1992), 880–903. Also see Robert M. Axelrod, *The Evolution of Cooperation*, Basic Books, 1984.

5. Problems 3.16–3.18 in Hopcroft and Ullman deal with "regularity-preserving" transformations of regular sets. A good start is the paper of Seiferas and McNaughton, Regularity preserving relations, *Theor. Comput. Sci.* **2** (1976), 147–154; Berstel et al., *Theor. Comput. Sci.* **354** (2006), 405–420.

6. Explore some of the connections between automata theory and number theory. See the paper of J.-P. Allouche, Automates finis en théorie des nombres, *Expositiones Math.* **5** (1987), 239–266. (This paper is in French–I have a copy of a preliminary version written in English.) Or see the book of Allouche and Shallit, *Automatic Sequences*, Cambridge Universty Press, 2003.

7. Explore *efficient algorithms* for determining whether or not a given word contains repetitions of various flavours. You could start with the following articles: A. J. Kfoury, A linear-time algorithm to decide whether a binary word contains an overlap, *RAIRO Informatique Théorique et Applications*, **22** (1988), 135–145; M. Crochemore, An optimal algorithm for computing the repetitions in a word, *Info. Proc. Letters* **12** (1981), 244–250. There has been a *lot* of work since then.

8. Find out more about Burnside's problem, which asks, *Is every finitely generated group of finite exponent finite?* and how the Thue-Morse word $\texttt{0110100110010110}\cdots$ played a role in its solution. You can start with the book of Adian, *The Burnside Problem and Identities in Groups* and the survey paper of Gupta, On groups in which every element has finite order, *Amer. Math. Monthly* **96** (1989), 297–308.

9. The "star-height" problem: given a regular set L, we may ask for the regular expression for L that has the fewest nested instances of Kleene closure. Is the star height computable? If we also allow intersection and complement in our regular expression, are there languages of "extended" star height ≥ 2? Look for the name "Hashiguchi".

10. Find out more about the class of languages that can be expressed as the intersection of a finite number of context-free languages. What are their closure properties? Try looking at Liu and Weiner, *Math. Systems Theory* **7** (1972), 185–192.

11. Investigate ω-languages, that is, those languages consisting of *infinite* words. A finite automaton accepts an ω-word if, for example, the path labeled by the word passes through an accepting state infinitely often; such automata are often called Büchi automata. Start with the survey by W. Thomas, "Automata on Infinite Objects", in J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, Volume B: Formal Models and Semantics. There is also the great book of Jean Berstel and Dominique Perrin, *Infinite Words: Automata, Semigroups, Logic and Games*, Elsevier.

12. Look into how regular languages can be efficiently learned. The paper of Angluin, Learning regular sets from queries and counterexamples, *Info. & Control* **75** (1987), 87–106, is a good place to start. Also see Ibarra & Jiang, Learning regular languages from counterexamples, *J. Comput. System Sci.* **43** (1991), 299–316.

13. Read about "rational series", a generalization of automata to formal power series. Look at the books of Salomaa & Soittola, *Automata-Theoretic Aspects of Formal Power Series*, or Berstel & Reutenauer, *Rational Series and Their Languages*. The latest edition of that book has been retitled to *Noncommutative Rational Series with Applications*.

14. Look into probabilistic languages and automata. Start with C. S. Wetherell, Probabilistic languages: a review and some open questions, *Computing Surveys* **12** (1980), 361–379; Michael O. Rabin, Probabilistic automata, *Inform. Control* **6** (1963), 230–245; A. Paz, Some aspects of probabilistic automata *Inform. Control* **9** (1966), 26–60.

15. For each of the usual operations $\otimes$ on regular languages, one can define the "state complexity" of the operation as the maximum possible number of states in a DFA accepting $L_1 \otimes L_2$, in terms of the number of states of the DFA's for L_1 and L_2. The point is to find good bounds on this state complexity. See, for example, S. Yu, Q. Zhuang, and K. Salomaa, The state complexities of some basic operations on regular languages, *Theoret. Comput. Sci.* **125** (1994), 315–328.

16. Look into good algorithms for minimizing finite automata. For example, the following article discusses a linear-time algorithm for minimizing certain kinds of automata: D. Revuz, Minimisation of acyclic deterministic automata in linear time, *Theoret. Comput. Sci.* **92** (1992), 181–189. Also, Bruce Watson has a survey paper in this area.

17. Look into graph grammars, a generative way of specifying a class of graphs. There are several conference proceedings in this area.

18. Look into packages that manipulate finite automata and/or regular expressions, such as "Grail" or "REGPACK" (E. Leiss, University of Waterloo technical report CS-77-32, October 1977). The Grail home page is `http://www.csit.upei.ca/~ccampeanu/Grail/`.

19. Look into generalizations of the pumping lemma and Ogden's lemma for CFL's. For example, start with Bader and Moura, A generalization of Ogden's lemma, *J. ACM* **29** (1982), 404–407.

20. Look into efficient methods for transforming grammars to Greibach normal form. For example, see Koch and Blum, "Greibach normal form transformation, revisited", in *STACS 97*, Lecture Notes in Computer Science # 1200, Springer, 1997, 47–54.

21. Look into undecidability and NP-completeness questions involving regular languages. Start with, for example, Halava and Harju, Undecidability of the equivalence of finite substitutions on regular language, *Theoret. Informatics Appl.* **33** (1999), 117–124; Ehrenfeucht and Rozenberg, Finding a homomorphism between two words is NP-complete, *Info. Proc. Letters* **9** (1979), 86–88.

22. Look into the relationship between cellular automata and regular languages. Start with Stephen Wolfram, Computation theory of cellular automata, *Commun. Math. Physics* **96** (1984), 15–57. This can also be found in Wolfram's book of collected papers, *Cellular Automata and Complexity.* Wolfram's new book, *A New Kind of Science*, also has some information, but at a much less technical level.

23. Look into quantum finite automata. You can start with this article `http://arxiv.org/pdf/1406.4048v1.pdf` of Cem Say and Yakaryilmaz.

24. Look into the connection between automata and formal logic. Some possibilities include the book of Straubing, *Finite Automata, Formal Logic, and Circuit Complexity.*

25. Read about the genus of a regular language: `https://arxiv.org/abs/1511.09405`.

Suggested Research Questions

You can find more questions under "Open Problems" on the course home page.

1. Consider the homomorphism defined by $\varphi(1) = 121$; $\varphi(2) = 12221$. This homomorphism has a infinite fixed point $r = r_0 r_1 r_2 \cdots$, which you obtain by iterating φ, starting with 1.

The sequence r has some very strange properties; for example, $r(16n+1) = r(64n+1)$ for $n = 0, 1, \ldots, 1864134$, but not for $n = 1864135$. Explain this.

2. Suppose we define a function on two integer sequences of equal (finite) length, as follows:

$$R(a,b) = (\overbrace{b_1, b_1, \ldots, b_1}^{a_1 \text{ times}}, \overbrace{b_2, b_2, \ldots, b_2}^{a_2 \text{ times}}, \ldots, \overbrace{b_n, b_n, \ldots, b_n}^{a_n \text{ times}}).$$

Here $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_n)$. Note that $R(a,b)$ is a sequence of length $\sum_{1 \le i \le n} a_i$.

Now let $X_1 = (2)$, and for $i \geq 1$

$$X_{i+1} = R(X_i,\ (\overbrace{1, 2, 1, 2, \ldots, 1, 2}^{n \text{ terms}})),$$

where $n = |X_i|$, the length of the sequence X_i. Thus we find, for example, $X_2 = (1, 1)$, $X_3 = (1, 2)$, $X_4 = (1, 2, 2)$, $X_5 = (1, 2, 2, 1, 1)$, etc.

Note that X_i is a prefix of X_{i+1} for $i \geq 3$, so we may consider an infinite sequence X of which all the X_i are prefixes:

$$X = (1, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 2, 1, 1, 2, \ldots).$$

This sequence is called the *Kolakoski* sequence in the literature.

Prove or disprove: in the limit, the infinite sequence X contains the same number of 1's as 2's. (More precisely, the ratio of the number of 1's to the number of 2's in a prefix of length n converges to 1.)

3. Show that, for each $b \geq 2$, the set of composite numbers expressed in base-b, is not a context-free language. (The case $b = 10$ is of particular interest.) Or do the same thing for the set of squares. (I have a mansucript by Horvath which claims to have solved the problem for powers in general, but I have not been able to follow it. More recently, there is a claimed proof that has been posted here: `https://tinyurl.com/yc7tmb5b`, but I don't understand it. You could try to check these proofs, and if correct, write up these proofs in your own words.)

4. A nonempty word w is said to be *primitive* if it cannot be expressed as $w = x^k$, where $k > 1$. In other words, a primitive word is a non-power. Prove (or disprove) that the set of all primitive words is not a context-free language. This is a big open problem.

5. The separating words problem: suppose you are given two distinct words u, v with $|u|, |v| \leq n$. What is the size of the smallest DFA which accepts u but rejects v, or vice versa? If u and v are of different lengths then a simple argument gives a $O(\log n)$ upper bound. How about if u and v are of the same length? Robson [*Info. Proc. Letters* **30** (1989), 209–214] showed that in this case a machine of size $O(n^{2/5}(\log n)^{3/5})$ exists. Can this be improved?

6. Here is a problem due to Bucher (*Bull. EATCS* **10** (Jan. 1980), p. 53): Given context-free languages L_1, L_2 with $L_1 \subset L_2$ and $L_2 - L_1$ infinite, need there be a context-free language L_3 with $L_1 \subset L_3 \subset L_2$ such that both $L_2 - L_3$ and $L_3 - L_1$ are infinite?

7. Can you find an explicit family of unary NFA's M with n states, such that the shortest regular expression for $L(M)$ has $\Omega(n^2)$ states?

8. Define $x_0 = 01$ and $x_{n+1} = x_n p$, where p is the shortest prefix of x_n that appears exactly once in x_n. Thus, for example, $x_1 = 010$, $x_2 = 01001$, and so forth. Let $\mathbf{x}$ denote the limit of the x_i.

(a) Is $\mathbf{x}$ ultimately periodic?

(b) Is $\mathbf{x}$ k-automatic for any k?

(c) What is the sequence of lengths $|x_i|$?

(d) What is the sequence of first differences of lengths $|x_{i+1}| - |x_i|$?

(e) What is the density of 1's in $\mathbf{x}$? Does it have a limit? What is the limit?

(f) What is the sequence (s_i) of the positions of 1's in $\mathbf{x}$? Is $s_{i+1} - s_i$ unbounded?

(g) What is the subword complexity of $\mathbf{x}$?

Similar questions could be asked for other starting points than $x_0 = 01$.

9. Is the following problem decidable? Given a finite automaton M, does M accept the base-2 representation of at least one perfect square?

10. Consider the regular language L_k over a k-letter alphabet that consists of all words having no two consecutive identical letters. Find good upper and lower bounds on the size of the shortest regular expression for L_k. (Size can be measured pretty much in any reasonable way.) An upper bound of the form $O(k^{2.176})$ is known, but no nontrivial lower bound is known. Can you improve the upper bound, or prove any nontrivial lower bound?

11. Read this paper: Orna Kupferman and Jonathan Mosheiff, Prime languages, *Information and Computation* **240** (2015) 90–107. The paper discusses those languages accepted by n-state DFAs that can be written as the intersection of some number of regular languages with smaller state complexity. There are many open questions, of which one obvious one is: what percentage, asymptotically, of all n-state DFA languages can be so expressed? Another one relates to width: the width of an n-state DFA language L is the minimum number of intersections of languages of smaller state complexity needed to represent it. If the width is finite, how large can it be? Currently here are examples known of width 3, but not larger.

12. Find good upper and lower bounds on the state complexity of L_n, the language of all strings that contain, as (scattered) subsequences, all permutations of $\{1, 2, 3, \ldots, n\}$. An upper bound of $2^{n \cdot 2^{n-1}}$ is known.

Suggested Wikipedia Articles

1. Equations in words (you will have to create it).

2. Combinatorics on words.

3. Squarefree word.

4. Dejean's conjecture.
5. DFA Minimization.
6. 2DPDA's (you will have to create it).
7. Bounded languages (you will have to create it).
8. Axel Thue.
9. Thue-Morse sequence.