Formal languages II: sub- and super-regular languages

LING83800

1 Introduction

The regular languages were one of the earliest formal language families to be characterized, but they are by no means the only such family. (Indeed, there are in theory an unbounded number of formal language families that one could characterize.)

There are several different ways one may characterize a formal language. The most common characterization is algebraic (or "grammatical"), which is how we defined the regular languages in the previous lecture, and how we will do so today. A second characterization is "automata-theoretic", or characterization in terms of the capacity of abstract computational devices; we provide an automata-theoretic characterization of the regular languages in the following lecture. A third type of characterization—which we won't provide in this class—is "logical"; for instance, every regular language corresponds to a definition in monadic second-order logic (Büchi 1960).

The four best-known formal languages form what's known as the *Chomsky hierarchy* (Chomsky 1956). Formally, this is a *containment hierarchy*: each pair of languages are in a strictly nesting superset/subset relationship such that

$$\text{regular} \subset \text{context-free} \subset \text{context-sensitive} \subset \text{recursively enumerable}.$$

Table 1 gives further characterizations.[1] In practice—and with the benefit of decades of hindsight—research in formal language theory, natural language processing, and linguistics focuses on two additional classes, neither of which yet has an agreed-upon formal characterization:

- The *subregular languages* are a proper subset of the regular languages. The best-characterized languages in this class are *learnable in the limit from positive data* (Gold 1967, Angluin 1980) meaning that one can, under certain generous assumptions, determine the exact language that generated a sample of strings.
- The *mildly context-sensitive languages* (MCSLs; Joshi 1985) are a proper subset of the context-sensitive languages and proper superset of the context-free languages. They are informally characterized by *polynomial parsing*: that is, one can efficiently solve the membership problem, i.e., determine whether or not a given string is a member of the language.

These two families figure heavily into linguists' conjectures about the actual complexity of human languages. The subregular languages were characterized in part in response to Gold's

[1] Note there is no logical reason why formal languages ought to form a proper hierarchy, and indeed, some classes of subregular languages do not nest with respect to other subregular classes.

Grammar	Languages	Recognition automaton
Type 3	regular	finite-state automaton
Type 2	context-free	pushdown automaton
Type 1	context-sensitive	linear-bounded Turing machine
Type 0	recursively enumerable	Turing machine

Table 1: Characterizations of the Chomsky hierarchy.

proof that the regular languages were not in learnable in the limit from positive data. Heinz and colleagues (e.g., Heinz 2010) conjecture that all phonotactic and morphotactic patterns are characterized by subregular languages learnable in the limit from positive data. Similarly, the mildly context-sensitive languages were developed concurrently with Schieber's (1985) demonstration that one human language—Swiss German—is not CFL. Since the Swiss German pattern is MCSL, it is conjectured that human languages are at most MCSL.

$$\ldots \subset \textbf{subregular} \subset \text{regular} \subset \text{context-free} \subset$$
$$\textbf{mildly context-sensitive} \subset \text{context-sensitive} \subset \text{recursively enumerable}.$$

Bibliographic note

The definition of strictly-local languages is based on Jäger and Rogers (2012). The definition of context-free languages is loosely based on Jurafsky and Martin, in preparation, chap. 17, who in turn draw from Hopcroft et al. 2006. Algorithms for parsing context-free grammars are covered in later courses.

Software note

SigmaPie[2] (Aksënova 2020) is a Python toolkit for working with certain families of subregular languages, including the strictly local languages. There are a huge number of libraries for context-free grammar parsing; one well-known one is `pyparsing`.[3]

2 Strictly local languages

A *strictly local* (SL) *language* is a proper subset of the subregular languages. A SL language is parameterized by positive counting number k and we may speak of the set of, e.g., SL_3 languages. It can be grammatically characterized either positively or negatively and these two characterizations are equivalent (i.e., for every positive characterization there is an equivalent negative characterization).

[2] `https://github.com/alenaks/SigmaPie`
[3] `https://github.com/pyparsing/pyparsing/`

2.1 Positive characterization

2.1.1 Definitions

In the positive characterization, an SL_k language is defined by a finite set of k-factors, substrings (i.e., n-grams) of length k. More formally, an SL grammar is a tuple $\Sigma, \text{^}, \$, K$ such that:

- Σ a set of *terminal symbols*
- ^ is the left boundary symbol
- $ is the right boundary symbol
- K is a set of k-factors, each of one of the following three forms:
  - Σ^k
  - $\text{^}\Sigma^{k-1}$
  - $\Sigma^{k-1}\$$

2.1.2 The membership problem

To determine whether a string is a member of an SL language, one simply has to check whether all of its k-factors are members of K. For example, if the string in question is $\beta\gamma\delta$ and one wants to test whether it is a member of an SL_2 grammar, then one must consider each of this string's 2-factors: $\text{^}\beta, \beta\gamma, \gamma\delta, \delta\$$.[4] The string is a member of the corresponding SL language if and only if each of these factors are present in the grammar's set of k-factors (i.e., if $K' \subseteq K$). Similarly, if the grammar is an SL_3 grammar, we would check k-factors of the string $\text{^}\beta\gamma, \ldots$, and so on.

Problem Given a (positive) SL grammar where $\Sigma = \{\text{a}, \text{b}, \text{c}\}$ and $K = \{\text{^a}, \text{aa}, \text{ab}, \text{bb}, \text{b\$}\}$, give one string of length 5 which is a member of the corresponding language, and one string of length 6 which is not.

2.2 Negative characterization

2.2.1 Definitions

In the negative characterization, K is simply a set of **disallowed** k-factors, of the same three forms as in the positive characterization.

Problem Let $\Sigma = \{\text{C}, \text{V}\}$ where these stand for consonants and vowels, respectively. Give a negative characterization of the so-called "strict CV" languages, those where every syllable is of the shape CV (e.g., CV, CVCV, etc.). State the minimal k and give all negative k-factors.

[4] Note that one "hallucinates" an initial ^ and a final $ for each candidate string. We will see a similar hallucination of boundary symbols when we formalize rewrite rules in a few weeks.

2.2.2 The membership problem

With this negative characterization, a string is the member of the corresponding SL language if and only if **none** of the k-factors is present in the string.

2.3 Non-equivalence

It is possible to conceive of regular languages which are not SL for any finite k. For instance, a language of the form $\Sigma^*\beta\Sigma^*$, a language which requires that β occurs at least once, is not SL, because this condition cannot be positively or negatively characterized in terms of finite k-factors.

2.4 Applications

The SL languages have been used to characterize many linguistic patterns, including stress (Rogers et al. 2013), morphotactics (Aksënova and De Santo 2018), and elements of syntax and semantics (e.g., Graf 2022).

3 Context-free languages

A *context-free language* (CFL) is a language characterized by a *context-free grammars* (CFGs), a formalization of the phrase structure grammars of traditional grammar. The formalism is originally due to Chomsky (1956) though it has been independently discovered several times. The insight underlying CFGs is the notion of *constituency*: that strings are generated by nested phrases.

3.1 Definitions

A CFG is a four-tuple N, Σ, R, S such that:

- N is a set of *non-terminal* symbols, corresponding to phrase markers in a syntactic tree.
- Σ is a set of *terminal* symbols, corresponding to words (i.e., X^0s) in a syntactic tree.
- R is a set of *production rules*. These rules are of the form $A \rightarrow \beta$ where $A \in N$ and $\beta \in (\Sigma \cup N)^*$. Thus A is a phrase label and β is a sequence of zero or more terminals and/or non-terminals.
- $S \in N$ is a designated start symbol (i.e., the highest projection in a sentence).

For simplicity, we assume N and Σ are disjoint. As is standard, we use Roman uppercase characters to represent non-terminals and Greek lowercase characters to represent terminals.

3.2 The membership problem

To determine whether or not a string is a member of a context-free language, one has to determine whether that string can be derived—in a sense to be made precise below—by that language's grammar. *Direct derivation* describes the relationship between the input to a single grammar rule in R and the resulting output. If there is a rule $A \to \beta$ in R, and α, γ are strings in $(\Sigma \cup N)^*$, then $\alpha A \gamma$ directly derives $\alpha\beta\gamma$:

$$\alpha A \gamma \Rightarrow \alpha\beta\gamma.$$

Derivation is a generalization of direct derivation which allows one to iteratively apply rules to strings. Given strings $\alpha_1, \alpha_2, \alpha_m \in (\Sigma \cup N)^*$ such that $\alpha_1 \Rightarrow \alpha_2$, and $\alpha_2 \Rightarrow \alpha_3$, ..., $\alpha_{m-1} \Rightarrow \alpha_m$, then

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

i.e., α_1 *derives* α_m (and α_1 also derives α_2, α_3, etc.). A string is a member of the CFL L_G if it can be derived by G, starting from the start symbol S, and, the (possibly infinite) language L_G is the set of strings that can be so derived.

Problem Enumerate the language generated by a CFG with the following rules:

$S \to NP\ VP$

$VP \to V\ NP$

$VP \to V$

$NP \to DT\ NN$

$NP \to$ `Kyle`

$DT \to$ `a` | `the`

$NN \to$ `cat` | `dog`

$V \to$ `barks` | `bites`

3.3 Non-equivalence

Speaking informally, regular languages are able to "count" up to a certain counting number. For instance, a regular language can require that any string in the language contain exactly n a's, no more than m b's, or between m and n c's. But as Jäger and Rogers put is, "no regular grammar is able to count two sets of symbols and compare their size if their size is potentially unlimited" (p. 1959).[5] However, context-free languages can do exactly this sort of unbounded comparison, and this is one example of how they are a proper superset of the regular languages. One intuitive

[5]One can formally prove that a language (like $\beta^n\gamma^n$) is non-regular using a technique known as the *pumping lemma*; see Hopcroft et al. 2006:§4.1 for a demonstration.

explanation for this fact is that derivation rules in a regular grammar must be *left-linear* or *right-linear*. That is, they are all of the form $A \rightarrow B\ \Sigma^*$ (a left-linear rule) or $A \rightarrow \Sigma^*\ B$ (a right-linear rule). But CFGs allow additional types of rules, such as *center-embedding* rules of the form $A \rightarrow \beta\ A\ \gamma$. Imagine this rule is part of the following CFG:

$$S \rightarrow A$$

$$A \rightarrow \beta\ A\ \gamma$$

$$A \rightarrow \epsilon$$

Intuitively this grammar derives the language $\beta^n\gamma^n$ (where n is some non-negative integer). However, regular languages can only approximate this language (e.g., with $\beta^*\gamma^*$), but cannot "insist" that the number of βs and γs match.

3.4 Chomsky-normal form

Syntacticians have long had a preference for *binary branching* syntactic structures, meaning that each non-terminal node has at most two children. As it happens, this assumption greatly simplifies parsing algorithms as well. One way this is enforced by converting grammars or treebanks to a format known as *Chomsky normal form* (CNF; Chomsky 1963). In Chomsky normal form, the elements of R, the set of production rules, are constrained to have one of two forms:

- $A \rightarrow B\ C$ where $A, B, C \in N$.
- $A \rightarrow \beta$ where $A \in N$ and $\beta \in \Sigma$.

In other words, the right-hand side of every rule either consists of two non-terminals or one terminal. There exists for every CFG grammar a *weakly equivalent* CNF grammar, meaning that there exists a CNF which generates the same language (though it does not necessarily assign exactly the same phrase structure). For instance, given the rule $A \rightarrow B\ C\ D$, we can convert this to two CNF rules, namely $A \rightarrow B\ X$ and $X \rightarrow C\ D$.

Problem Given the CFG rule $M \rightarrow X\ \lambda\ \rho\ Y$, where X, Y are non-terminals and λ, ρ are terminals, convert the rule into a series of CNF rules.

3.5 Applications

There exist relatively-efficient cubic-time recognition and parsing algorithms for CFGs. Nearly all programming languages are described by—and parsed using—a CFG,[6] and CFG grammars of human languages are widely used as a model of syntactic structure in natural language processing and understanding tasks.

[6]Python programs are even described by a CFG (`https://docs.python.org/3/reference/grammar.html`). When you execute a Python script, it is parsed using this grammar specification.

References

Aksënova, Alëna. 2020. Tool-assisted induction of subregular languages and mappings. Doctoral dissertation, State University of New York at Stony Brook.

Aksënova, Alëna, and Aniello De Santo. 2018. Strict locality in morphological derivations. In *Proceedings of the Fifty-third Annual Meeting of the Chicago Linguistic Society*, 1–12.

Angluin, Dana. 1980. Inductive inference of formal languages from positive data. *Information and Control* 45:117–135.

Büchi, Julius Richard. 1960. Weak second order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 6:66–92.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 3:113–124.

Chomsky, Noam. 1963. Formal properties of grammars. In *Handbook of Mathematical Psychology*, ed. R. Duncan Luce, Robert R. Bush, and Eugene Galanter, 323–418. John Wiley & Sons.

Gold, E. Mark. 1967. Language identification in the limit. *Information and Control* 10:447–474.

Graf, Thomas. 2022. Subregular linguistics: bridging theoretical linguistics and formal grammar. *Theoretical Linguistics* 48:145–184.

Heinz, Jeffrey. 2010. Learning long-distance phonotactics. *Linguistic Inquiry* 41:623–661.

Hopcroft, John E., Rajeev Motwani, and Jeffrey D. Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation.* Pearson, 3rd edition.

Joshi, Aravind K. 1985. Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions? In *Natural Language Parsing*, ed. David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, 206–250. Cambridge University Press.

Jurafsky, Dan, and James H. Martin. In preparation. *Speech and Language Processing*. 3rd edition.

Jäger, Gerhard, and James Rogers. 2012. Formal language theory: refining the Chomsky hierarchy. *Philosophical Transactions of the Royal Society B* 367:1956–1970.

Rogers, James, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In *International Conference on Formal Grammar*, 90–108.

Schieber, Stuart M. 1985. Evidence against the context-freeness of natural languages. *Linguistics and Philosophy* 8:333–343.