Informatics 2A: Language Complexity and the Chomsky Hierarchy

Slides by Bonnie Webber (modified by Stuart Anderson)

September 28, 2010

Starter 1

Is there a finite state machine that recognises all those strings $s$ from the alphabet \{a, b\} where the difference between the number of $a$s and number of $b$s is less than $k$ for some constant $k$?

▶ True or
▶ False?

Starter 2

Is there a finite state machine that recognises all those strings $s$ from the alphabet \{a, b\} where the difference between the number of $a$s and number of $b$s is less than $k$ for some constant $k$ in every prefix of $s$?

A prefix of any string $s$ is a string $p$ such that there is a string $q$ such that $s = pq$. Note that it is possible that $q = \varepsilon$.

▶ True or
▶ False?
Languages: Collection and Generation

A formal language is the possibly infinite set of strings over a finite set of symbols (called a vocabulary or lexicon).

Such strings are also called sentences of the language.

Where do the sentences come from?
- from a (finite) list — useful, but not very interesting (maybe more interesting when we have collections of really large samples of speech or text).
- from a grammar — abstract characterisation of the strings belonging to a language. Grammars are a generative mechanism, they give rules for generating potentially infinite collection of finite strings.

Different kinds of Language

Programming language: Programmers are given an explicit grammar for the syntactically valid strings of the language that they must adhere to.

Human language: Children hear/see sentences of a language (their “mother tongue” or other languages used at home or in their community) and are sometimes (but not always!) corrected if a string they generate isn’t in the language.

Without being given an explicit grammar, how do children learn a grammar(s) for the infinite number of sentences that belong to the language(s) they speak and understand?

Structure and Meaning

Small red androids sleep quietly. ✓
Colorless green ideas sleep furiously. ✓
Sleep green furiously ideas colorless. ✫
Mary persuaded John to wash himself with lavender soap. ✓
Mary persuaded John to wash herself with lavender soap. ✫
Mary persuaded John to wash her with lavender soap. ✓
Mary promised John to wash herself with lavender soap. ✫
Mary promised John to wash him with lavender soap. ✓

- Characterising child language acquisition is one goal of Linguistics.
- Characterising language learnability (grammar induction) is one goal of Informatics.
More broadly, the goals of Linguistics are to characterise:

- individual languages: figuring out and specifying their sound systems, grammars, and semantics;
- how children learn language and what allows them to do so;
- the social systems of language use;
- how individual languages change over time, and how new languages arise.

Work on formal languages in Informatics contributes to achieving these goals through

- clear computational methods of characterising the complexity of languages;
- clear computational methods for processing languages;
- clear computational theories of language learnability.

Noam Chomsky

- Credited with the creation of the theory of generative grammar
- Significant contributions to the field of theoretical linguistics
- Sparked the cognitive revolution in psychology through his review of B.F. Skinner’s Verbal Behavior
- Credited with the establishment of the Chomsky-Schutzenberger hierarchy, a classification of formal languages in terms of their generative power

Questions

We heard from Lecture 2 that grammars differ in their complexity.

- What is complex about a complex grammar?
- How does adding a data structure to an automaton allow its corresponding grammar to be more complex?
- How does removing limits on how the store on an automaton is accessed allow its corresponding grammar to be more complex?
- Is there any relationship between language complexity and how hard a language is to learn?

Chomsky’s desire to find a “simple and revealing” grammar that generates exactly the sentences of English led him to the discovery that some models of language were more powerful than others.


Linguistic theory attempts to explain the ability of a speaker to produce and understand new sentences, and to reject as ungrammatical other new sequences, on the basis of his limited linguistic experience. [Chomsky 1956, p. 113]

The adequacy of a linguistic theory can be tested by looking at a grammar for a language constructed according to the theory and seeing if it makes predictions that accord with what’s found in a large corpus of sentences of that language.

What about what is not found in a large corpus of sentences?

Chomsky’s paper explores the sort of linguistic theory that is “required as a basis for an English grammar what will describe the set of English sentences in an interesting and satisfying manner”.

Slides by Bonnie Webber (modified by Stuart Anderson)
Three Models for the Description of Language

For that description to be “interesting and satisfying”, Chomsky felt that a grammar had to be

- finite
- “revealing”, in allowing strings to be associated with meaning (semantics) in a systematic way

The three models he considered were:

2. Phrase structure grammars reflecting pedagogical ideas of “sentence diagramming”
3. Transformational grammars

Dependencies don’t need to be binary

- R.D. Laing took this to extremes in Knots his play on sanity in everyday language.
- “There must be something the matter with him because he would not be acting as he does unless there was therefore he is acting as he is because there is something the matter with him”

Dependency and Complexity

Much of Chomsky’s argument in 3MDL is based on the notion of dependency:

Suppose $s = a_1 a_2 \ldots a_n$ is a sentence of language $L$. We say that $S$ has an $i$-$j$ dependency if when symbol $a_i$ is replaced with symbol $b_i$, the string is no longer a sentence of $L$ and when symbol $a_j$ is then replaced by some new symbol $b_j$, the resulting string is a sentence of $L$.

We’ve already seen such a dependency in English: Mary persuaded John to wash himself with lavender soap. John ⇒ Sue himself ⇒ herself

Mary persuaded Sue to wash herself with lavender soap.

Dependency Sets

- If we restrict ourselves to binary dependencies, then for any sentence $s$ we can construct a dependency set $D = \{(i_1, j_1), \ldots, (i_k, j_k)\}$ where each pair is a dependency in $S$.
- For example: If Mary has persuaded John to wash himself with lavender soap, then he is clean. (dep_set size = 4)
- Sentences in the language generated by a regular grammar can have dependencies.
- Consider the regular language described by a regular expression:
  
  $$L_0 = (b^* + (ab^*c))^*$$

  I.e. where every $a$ is eventually followed by a $c$ and only $bs$ may intervene
An example: $L_0$

- $bbbabbcbbbacbbbbbb \in L_0$ is a typical sentence in the language.
- $\{(4,7),(11,13)\}$ is the dependency set for the sentence.
- If we use the convention that we colour the pair of symbols in the dependency set the same colour and we can reuse colours for parts of the string after the later symbol in the dependency pair has appeared. How many colours do we need to colour the symbols in sentences in $L_0$?
- $bbbabbcbbbabcbbbb$ uses just one colour.

Limits to Dependencies

- The number of colours we need to colour the dependency set of a sentence gives us a measure of the amount that has to be remembered about earlier symbols to get the dependencies right. If we need $k$ colours then we need to remember $k$ symbols at most at any one time.
- For any regular language $R$ there must exist a constant $k_R$ such that the dependency set for any sentence in the language can be coloured with at most $k_R$ colours.
- What do you make of this claim?

Example 1

- $L_1$ consists of all (and only) sentences over $\{a,b\}$ containing $n$ as followed by $n$ bs: e.g., $ab$, $aabb$, $aaabbb$, ...
- Suggest a dependency set for $aaaaaabbbbbb$.
- How many colours does it take to colour the dependencies?
- How many colours does it take to colour the dependencies for $a^n b^n$?
- Is this a good example? What would you need to add to improve it?

Example 2

- $L_2$ consists of all (and only) sentences over $\{a,b\}$ containing a string of $a$s and $b$s followed by its reverse $\{aa^R \mid a \in \{a,b\}^*\}$: e.g., $aa$, $bb$, $abba$, $baab$, $abaaaaabbbbbbbaaabbaabaabba$, ...
- What is the dependency set for $aaaaaaa$?
- How many colours are required to colour this dependency set?
- How many colours does it take to colour the dependency set for $a^{2n}$?
Example 3

- $L_3$ consists of all (and only) sentences over \{a, b\} containing a string of as and bs followed by the same string over again, \(\{\alpha \alpha | \alpha \in \{a, b\}^*\}\): e.g. aa, bb, abab, baba, ababb, abaaba, . . .
- What is the dependency set for aaaaaaaa?
- How many colours does it take to colour the dependencies?
- How many colours does it take to colour the dependency set for $a^{2n}$?

The simplest languages – ones that can be described by a regular grammar – need at most a finite number of colours to colour any dependency set in the language.

They are at the lowest rung of the Chomsky Hierarchy.

Are all languages with arbitrarily many dependencies equally complex?
A phrase structure grammar consists of
- a finite vocabulary \( V \)
- a finite set \( \Sigma \) of initial strings over \( V \)
- a finite set of rules of the form \( X \rightarrow Y \) where
  1. \( X \) and \( Y \) are strings over \( V \)
  2. \( Y \) is formed from \( X \) by replacing one symbol of \( X \) with a string over \( V \)
  3. Neither the replaced symbol nor the replacing string is empty (\( \epsilon \)).

Rules of the simplest PS Grammars contain only a single symbol on their left-hand side – e.g.,
\[
\begin{align*}
\Sigma &: \{S\} \\
S &\rightarrow NP \ VP \\
VP &\rightarrow \text{verb} \ NP \\
NP &\rightarrow \text{the man} \\
NP &\rightarrow \text{the book} \\
\text{verb} &\rightarrow \text{took}
\end{align*}
\]
These are called Context-free PSGs or, for short, Context-free Grammars (CFGs).

Some dependencies that are beyond the capability of a regular grammar can be captured by a context-free grammar. Such dependencies are ones that can be generated locally.

Recall \( L_1 \): all (and only) sentences over \{a, b\} containing \( n \) a’s followed by \( n \) b’s.
Here, the presence of a b on the right of the string depends on there being a comparable a on the left.

Simple PSG for generating \( L_1 \):
\[
\begin{align*}
V &= \{a, b, S\} \\
\Sigma &= S \\
PS\ rules: &\quad S \rightarrow aSb \\
&\quad S \rightarrow ab
\end{align*}
\]
Derivation

Sample derivation:

\[ S \Rightarrow aSB \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaaabbbb \]

Dependency and Complexity Revisited

Are all dependencies local?
Are there dependencies that cannot be captured in a CFG?

- context-free grammars
- regular grammars

The dependency in \( L_3 = \{XX\} \) where \( X \) is a string over \( \{a, b\} \) cannot be captured by a CFG, nor can the dependency in \( L_4 \), consisting of all (and only) sentences over \( \{a, b, c\} \) containing a string of \( n \) \( a \)'s, then \( n \) \( b \)'s followed by \( n \) \( c \)'s – e.g., abcaabbcc, aabbbccc, etc.

Context-sensitive PSGs

Phrase structure grammars with rules whose LHS contain \( \geq 1 \) symbol are called context-sensitive phrase structure grammars or simply, context-sensitive grammars.

Simple context-sensitive grammar for generating \( L_4 \):

- \( V = \{a, b, c, S, B\} \)
- \( \Sigma = S \)
- \( PS \) rules:
  - \( S \rightarrow abc \mid aSBc\)
  - \( cB \rightarrow Bc \)
  - \( bB \rightarrow bb \)

Sample derivation:

\[ S \Rightarrow aSBc \Rightarrow aaSBcBc \Rightarrow aaabcBcBc \Rightarrow aaabBccBc \Rightarrow aaabbcccBc \Rightarrow aaabbbccc \]

Context on the LHS allows for more dependencies and hence more complexity.
Top of the Chomsky Hierarchy

Arbitrary re-write systems that can take account of any amount of context on the LHS and re-write any number of symbols, called **Type 0 grammars**.

<table>
<thead>
<tr>
<th>Type 0 grammars</th>
</tr>
</thead>
<tbody>
<tr>
<td>context-sensitive grammars</td>
</tr>
<tr>
<td>context-free grammars</td>
</tr>
<tr>
<td>regular grammars</td>
</tr>
</tbody>
</table>

This is what is normally called the **Chomsky hierarchy**.