Context-sensitive languages

Informatics 2A: Lecture 28

John Longley

School of Informatics
University of Edinburgh
jrl@inf.ed.ac.uk

26 November 2015

Recap: context-sensitivity in natural language

An example of context sensitivity in natural language was presented in Lecture 25:

- Crossing dependencies in Swiss German (and Dutch).

There are other phenomena that are most naturally described in a 'context-sensitive' way (e.g. choice between the determiners *a* and *an*).

Such phenomena take natural languages outside the context-free level of the Chomsky hierarchy.

It is believed that natural languages naturally live (comfortably) within the context-sensitive level of the Chomsky hierarchy.

In today's lecture . . .

. . . we look at what lies beyond context-free languages from a formal language viewpoint.

- How to show that a language is not context free.
- Defining the notion of context-sensitive language using context-sensitive grammars.
- An alternative characterisation of context-sensitive languages using noncontracting grammars.
- The notion of unrestricted grammar, and the associated recursively-enumerable languages.

Non-context-free languages

We saw in Lecture 8 that the pumping lemma can be used to show a language isn't regular.

There's also a context-free version of this lemma, which can be used to show that a language isn't even context-free:

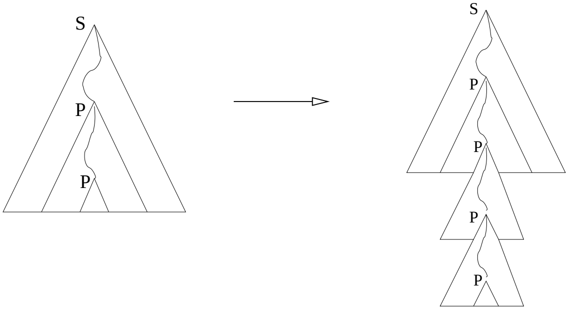
Pumping Lemma for context-free languages. Suppose L is a context-free language. Then L has the following property.

> *(P) There exists $k \geq 0$ such that every $z \in L$ with $|z| \geq k$ can be broken up into five substrings, $z = uvwxy$, such that $|vx| \geq 1$, $|vwx| \leq k$ and $uv^iwx^iy \in L$ for all $i \geq 0$.*

Context-free pumping lemma: the idea

In the regular case, the key point is that any sufficiently long string will visit the same state twice.

In the context-free case, we note that any sufficiently large syntax tree will have a downward path that visits the same non-terminal twice. We can then 'pump in' extra copies of the relevant subtree and remain within the language:



Context-free pumping lemma: continued

More precisely, suppose L has a CFG in CNF with m non-terminals.

Then take k so large that every syntax tree for a string of length $\geq k$ contains a path of length $> m + 1$.

Such a path (even with the root node removed, which means the remaining path has length $> m$) is guaranteed to visit the same nonterminal twice.

To show that a language L is not context free, we just need to prove that it satisfies the negation $(\neg\text{P})$ of the property (P):

> $(\neg P)$ *For every* $k \geq 0$, *there exists* $z \in L$ *with* $|z| \geq k$ *such that, for every decomposition* $z = uvwxy$ *with* $|vx| \geq 1$ *and* $|vwx| \leq k$, *there exists* $i \geq 0$ *such that* $uv^iwx^iy \notin L$.

Standard example 1

The language $L = \{a^n b^n c^n \mid n \geq 0\}$ isn't context-free!

We prove that $(\neg\mathrm{P})$ holds for L:

Standard example 1

The language $L = \{a^n b^n c^n \mid n \geq 0\}$ isn't context-free!

We prove that $(\neg\text{P})$ holds for L:

Suppose $k \geq 0$.

Standard example 1

The language $L = \{a^n b^n c^n \mid n \geq 0\}$ isn't context-free!

We prove that $(\neg\mathrm{P})$ holds for L:

Suppose $k \geq 0$.

We choose $z = a^k b^k c^k$. Then indeed $z \in L$ and $|z| \geq k$.

Standard example 1

The language $L = \{a^n b^n c^n \mid n \geq 0\}$ isn't context-free!

We prove that $(\neg\mathrm{P})$ holds for L:

Suppose $k \geq 0$.

We choose $z = a^k b^k c^k$. Then indeed $z \in L$ and $|z| \geq k$.

Suppose we have a decomposition $z = uvwxy$ with $|vx| \geq 1$ and $|vwx| \leq k$.

Standard example 1

The language $L = \{a^n b^n c^n \mid n \geq 0\}$ isn't context-free!

We prove that $(\neg\text{P})$ holds for L:

Suppose $k \geq 0$.

We choose $z = a^k b^k c^k$. Then indeed $z \in L$ and $|z| \geq k$.

Suppose we have a decomposition $z = uvwxy$ with $|vx| \geq 1$ and $|vwx| \leq k$.

Since $|vwx| \leq k$, the string vwx contains at most two different letters. So there must be some letter $d \in \{a, b, c\}$ that does not occur in vwx.

Standard example 1

The language $L = \{a^n b^n c^n \mid n \geq 0\}$ isn't context-free!

We prove that $(\neg\mathrm{P})$ holds for L:

Suppose $k \geq 0$.

We choose $z = a^k b^k c^k$. Then indeed $z \in L$ and $|z| \geq k$.

Suppose we have a decomposition $z = uvwxy$ with $|vx| \geq 1$ and $|vwx| \leq k$.

Since $|vwx| \leq k$, the string vwx contains at most two different letters. So there must be some letter $d \in \{a, b, c\}$ that does not occur in vwx.

But then $uwy \notin L$ because at least one character different from d now occurs $< k$ times, whereas d still occurs k times.

Standard example 1

The language $L = \{a^n b^n c^n \mid n \geq 0\}$ isn't context-free!

We prove that $(\neg\mathrm{P})$ holds for L:

Suppose $k \geq 0$.

We choose $z = a^k b^k c^k$. Then indeed $z \in L$ and $|z| \geq k$.

Suppose we have a decomposition $z = uvwxy$ with $|vx| \geq 1$ and $|vwx| \leq k$.

Since $|vwx| \leq k$, the string vwx contains at most two different letters. So there must be some letter $d \in \{a, b, c\}$ that does not occur in vwx.

But then $uwy \notin L$ because at least one character different from d now occurs $< k$ times, whereas d still occurs k times.

We have shown that $(\neg\mathrm{P})$ holds with $i = 0$.

Standard example 2

The language $L = \{ss \mid s \in \{a, b\}^*\}$ isn't context-free!

We prove that $(\neg P)$ holds for L:

Suppose $k \geq 0$.

We choose $z = a^k b\, a^k b\, a^k b\, a^k b$. Then indeed $z \in L$ and $|z| \geq k$.

Suppose we have a decomposition $z = uvwxy$ with $|vx| \geq 1$ and $|vwx| \leq k$. Since $|vwx| \leq k$, the string vwx contains at most one b.

There are two main cases:

- vx contains b, in which case uwy contains exactly 3 b's.
- Otherwise uwy has the form $z = a^g b\, a^h b\, a^i b\, a^j b$ where either:
  - exactly two adjacent numbers from g, h, i, j are $< k$ (this happens if w contains b and $|v| \geq 1 \leq |x|$), or
  - exactly one of g, h, i, j is $< k$ (this happens if w contains b and one of v, x is empty, or if vwx does not contain b).

In each case, we have $uwy \notin L$. So $(\neg P)$ holds with $i = 0$.

Complementation

Consider the language L' defined by:

$$\{a,b\}^* - \{ss \mid s \in \{a,b\}^*\}$$

This is context free.
Idea: If $t = t_1 \ldots t_{2n} \in L'$, there's some $i \leq n$ such that $t_i \neq t_{n+i}$. This means that t has the form *waxybz* or *wbxyaz*, where $|w| = |x|$ and $|y| = |z|$. Not hard to give a CFG that generates all such strings. (See Kozen p. 155).

The complement of L' is

$$\{a,b\}^* - L' \quad = \quad \{ss \mid s \in \{a,b\}^*\}$$

which, as we've seen, is *not* context-free.

So context-free languages are not closed under complementation.

Context sensitive grammars

A Context Sensitive Grammar has productions of the form

$$\alpha X \gamma \ \rightarrow \ \alpha \beta \gamma$$

where X is a nonterminal, and α, β, γ are sequences of terminals and nonterminals (i.e., $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$) with the requirement that β is nonempty.

So the rules for expanding X can be sensitive to the context in which the X occurs (contrasts with context free).

Minor wrinkle: The nonempty restriction on β disallows rules with right-hand side ϵ. To remedy this, we also permit the special rule

$$S \rightarrow \epsilon$$

where S is the start symbol, and with the restriction that this rule is only allowed to occur if the nonterminal S does not appear on the right-hand-side of any productions.

Context sensitive languages

A language is context sensitive if it can be generated by a context sensitive grammar.

The non-context-free languages:

$$\{a^n b^n c^n \mid n \geq 0\}$$
$$\{ss \mid s \in \{a, b\}^*\}$$

are both context sensitive.

In practice, it can be quite an effort to produce context sensitive grammars, according to the definition above.

It is often more convenient to work with a more liberal notion of grammar for generating context-sensitive languages.

General and noncontracting grammars

In a general or unrestricted grammar, we allow productions of the form

$$\alpha \;\rightarrow\; \beta$$

where α, β are sequences of terminals and nonterminals, i.e., $\alpha, \beta \in (N \cup \Sigma)^*$, with α containing at least one nonterminal.

In a noncontracting grammar, we restrict productions to the form

$$\alpha \;\rightarrow\; \beta$$

with α, β as above, subject to the additional requirement that $|\alpha| \leq |\beta|$ (i.e., the sequence β is at least as long as α).
In a noncontracting grammar also permit the special production

$$S \rightarrow \epsilon$$

where S is the start symbol, as long as S does not appear on the right-hand-side of any productions.

Example noncontracting grammar

Consider the noncontracting grammar with start symbol S:

$$\begin{array}{rcl} S & \longrightarrow & abc \\ S & \longrightarrow & aSBc \\ cB & \longrightarrow & Bc \\ bB & \longrightarrow & bb \end{array}$$

Example derivation (underlining the sequence to be expanded):

$$\underline{S} \;\Rightarrow\; a\underline{S}Bc \;\Rightarrow\; aab\underline{cB}c \;\Rightarrow\; aa\underline{bB}cc \;\Rightarrow\; aabbcc$$

Exercise: Convince yourself that this grammar generates exactly the strings $a^n b^n c^n$ where $n > 0$.

(N.B. With noncontracting grammars and CSGs, need to think in terms of derivations, not syntax trees.)

Noncontracting = Context sensitive

Theorem. A language is context sensitive if and only if it can be generated by a noncontracting grammar.

That every context-sensitive language can be generated by a noncontracting grammar is immediate, since context-sensitive grammars are, by definition, noncontracting.

The proof that every noncontracting grammar can be turned into a context sensitive one is intricate, and beyond the scope of the course.

Sometimes (e.g., in Kozen) noncontracting grammars are called context sensitive grammars; but this terminology is not faithful to Chomsky's original definition.

The Chomsky Hierarchy

At this point, we have a fairly complete understanding of the machinery associated with the different levels of the Chomsky hierarchy.

- Regular languages: DFAs, NFAs, regular expressions, regular grammars.
- Context-free languages: context-free grammars, nondeterministic pushdown automata.
- Context-sensitive languages: context-sensitive grammars, noncontracting grammars.
- Recursively enumerable languages: unrestricted grammars.

Context-sensitivity in programming languages

Some aspects of typical programming languages can't be captured by context-free grammars, e.g.

- Typing rules
- Scoping rules (e.g. variables can only be used in contexts where they have been 'declared')
- Access constraints (e.g. use of `public` vs. `private` methods in Java).

The usual approach is to give a CFG that's a bit 'too generous', and then separately describe these additional rules.
(E.g. typechecking done as a separate stage after parsing.)

In principle, though, all the above features fall within what can be captured by context-sensitive grammars. In fact, no programming language known to humankind contains anything that can't.

Scoping constraints aren't context-free

Consider the simple language L_1 given by

$$S \;\rightarrow\; \epsilon \;\mid\; \texttt{declare}\ v; S \;\mid\; \texttt{use}\ v; S$$

where v stands for a lexical class of variables. Let L_2 be the language consisting of strings of L_1 in which variables must be declared before use.

Assuming there are infinitely many possible variables, it can be shown that L_2 is not context-free, but is context-sensitive.

(If there are just n possible variables, we could in theory give a CFG for L_2 with around 2^n nonterminals — but that's obviously silly. . .)

Summary

- Context-sensitive languages are a big step up from context-free languages in terms of their power and generality.
- Natural languages have features that can't be captured conveniently (or at all) by context-free grammars. However, it appears that NLs are only mildly context-sensitive — they only exploit the low end of the power offered by CSGs.
- Programming languages contain non-context-free features (typing, scoping etc.), but all these fall comfortably within the realm of context-sensitive languages.
- Next time: what kinds of machines are needed to recognize context-sensitive languages?