Lecture A/2: Grammars

Dr Janka Chlebikova

School of Computing
University of Portsmouth

Oct 3, 2024

Introduction to Grammars

Remember: A language over an alphabet Σ is a set of strings from Σ.

We can define a language:

- giving a set of strings or combining from the existing languages using operations such as products, unions, . . . (Lecture 1)
- using a grammar (this Lecture)
- . . .

A grammar is a set of rules used to define a language – the structure of the strings in the language.

This lecture: "how to generate a language from a grammar" and "how to describe a grammar of a language"

Two alphabets of grammar

To describe a grammar for a language – two collections of alphabets (symbols) are necessary.

- Terminals are those symbols from which all strings in the language are made – symbols of a 'given' alphabet for a generated language.
  (Usually lower case letters.)
- Non-terminal are 'temporary' symbols (disjoint from terminals) used to define the grammar replacement rules (in the production rules). These must all be replaced by terminals before the production can successfully make a valid string of the language.
  (Usually upper case letters.)

Productions

Furthermore, a grammar for a language L (over an alphabet Σ) consists of a set of grammar rules (productions) of the following form:

$$\alpha \to \beta,$$

where α, β are strings of symbols taken from the set of terminals (Σ) and non-terminals.

A grammar rule $\alpha \to \beta$ can be read in any of several ways:

"replace α by β",
"α produces β",
"α rewrites to β" ,
"α reduces to β".

Formal definition of grammar

Example: If $\Sigma = \{a, b\}$ and S is a non-terminal symbol then the rules $S \to aS$, $S \to \Lambda$ are examples of productions for a grammar L.

Now we are ready for the formal definition of a grammar:

1. An alphabet T of symbols called **terminals**. (Identical to the alphabet of the resulting language.)
2. An alphabet N of grammar symbols called **non-terminals**. (Used in the production rules.)
3. A specific non-terminal called the **start symbol**. (Usually S.)
4. **A finite set of productions** of the form $\alpha \to \beta$, where α and β are strings over the alphabet $N \cup T$.

Example 1

Let a grammar G be defined by:

- the set of terminals $T = \{a, b\}$,
- the only non-terminal start symbol S,
- the set of production rules:

$$S \to \Lambda, \quad S \to aSb$$

or in shorthand:

$$S \to \Lambda \,|\, aSb$$

Which strings belong to the language generated by this grammar? Is there any difference with the previous grammar?

Given a grammar, which strings belong to the language generated by the grammar?

How to generate a language using grammar

- Every grammar has a special non-terminal symbol called a start symbol, and there must be at least one production with left side consisting of only the start symbol.
- Starting from the production rules with the start symbol, we can step by step generate all strings belonging to the language described by a given grammar.

Back to Example 1. The grammar G contains the productions $S \to \Lambda \,|\, aSb$ with the non-terminal start symbol S. This means in the first step we get Λ and aSb.

Derivations

A string made up of terminal (grammar) symbols and non-terminal symbols is called a sentential form.

Back to Example 1. aSb is a sentential form for the terminals $\{a, b\}$ and non-terminal S.

To carry on with generation of strings, we introduce derivation.

Definition (derivation)

If x and y are sentential forms and $\alpha \to \beta$ is a production, then the replacement of α by β in $x\alpha y$ is called a derivation, and we denote it by writing

$$x\alpha y \Rightarrow x\beta y.$$

Language generated by grammar

Back to Example 1. The grammar contains the production $S \to aSb$, so that $aaSbb$ could be derived from aSb, that means $aSb \Rightarrow aaSbb$.

As we can use the production rules again and again, we can also get $S \Rightarrow a\underline{S}b \Rightarrow aa\underline{S}bb \Rightarrow aaaSbbb \ldots$

The following three symbols with their associated meanings are used quite often in discussing derivations:

$\Rightarrow$ derives in one step,

$\Rightarrow^+$ derives in one or more steps,

$\Rightarrow^*$ derives in zero or more steps.

Back to Example 1. Which strings can we derive from the start symbol? $S \Rightarrow \Lambda$, $S \Rightarrow a\underline{S}b \Rightarrow ab$ hence $S \Rightarrow^* ab$, $S \Rightarrow^* aaaSbbb$,

...

Formal definition of $L(G)$

The set of all strings (over terminal symbols) which can be derived from the start symbol is the language generated by the grammar G.

Back to Example 1. $L(G) = \{\Lambda, ab, aabb, aaabbb, \ldots\}$

Definition

If G is a grammar with start symbol S and set of terminals T, then the language generated by G is the following set:

$$L(G) = \{s \mid s \in T^* \text{ and } S \Rightarrow^+ s\}.$$

That is, it's the set of all strings containing only terminal symbols which can be derived from the start symbol using one or more steps.

Example 2

Let $\Sigma = \{a, b, c\}$ be the set of terminal symbols and $\{A, S\}$ be the set of non-terminal symbols with the start symbol S. A language L over Σ is defined by the following productions:

$S \to b \,|\, aA, \quad A \to c \,|\, bS$

Examples of strings which belong to the language L:

- Clearly, we can generate b.
- All longer strings begin with a. All strings will either end with b or ac.
- We can make the strings: b, ac, abb, $abac$, $ababb$, $ababac$, $abababb$, ...
- Is the following characterisation correct: 'any string from L contains a, b (in any order) and ends with either b or ac'? ... NO!, e.g. $ba, abaabac \notin L$

Example 3

Let $\Sigma = \{a, b\}$ be the set of terminal symbols, and $\{A, B, S\}$ be the set of non-terminal symbols with the start symbol S. Further, a set of productions is given for a language L:

$S \to AB, \quad A \to \Lambda \,|\, aA, \quad B \to \Lambda \,|\, bB$

Is the string *aab* from the language L?

Yes! For example, we can have

$$S \Rightarrow \underline{A}B \Rightarrow a\underline{A}B \Rightarrow aa\underline{A}B \Rightarrow aa\Lambda B = aa\underline{B} \Rightarrow aab\underline{B} \Rightarrow aab\Lambda = aab,$$

hence $S \Rightarrow^{+} aab$.

This is a leftmost derivation, as we produce the leftmost characters first.

Example 4

Let $\Sigma = \{a, b, c\}$ be the set of terminal symbols, S be the only non-terminal symbol. Which language is described by the following four productions?

$S \to \Lambda$

$S \to aS$

$S \to bS$

$S \to cS$

Or in shorthand: $S \to \Lambda \,|\, aS \,|\, bS \,|\, cS$.

Try to realize that all strings from Σ^* can be generated by these rules and verify it for the string $aacb$.

$$S \Rightarrow a\underline{S} \Rightarrow aa\underline{S} \Rightarrow aac\underline{S} \Rightarrow aacb\underline{S} \Rightarrow aacb\Lambda = aacb$$

Hence, $S \Rightarrow^* aacb$.

Infinite languages

- Notice that there is no bound on the length of strings in an infinite language.
- Therefore there is no bound on the number of derivation steps used to derive the strings.
- If the grammar has n productions, then any derivation consisting of $n + 1$ steps must use some production twice.
- If the language is infinite, then some production or sequence of productions must be used repeatedly to construct the derivations.

Example of infinite languages

Example. The infinite language $\{a^n b \mid n \geqslant 0\}$ can be described by the grammar,

$$S \to b \mid aS.$$

To derive the string $a^n b$, use the production $S \to aS$ repeatedly n times and then stop the derivation by using the production $S \to b$.

The production $S \to aS$ allows us to say

"If S derives w, then it also derives aw."

Recursion/indirect recursion

A production is called recursive if its left side occurs on its right side.

Example. The production $S \to aS$ is recursive.

A production $A \to \ldots$ is indirectly recursive if A derives (in two or more steps) a sentential form that contains A.

Example. If the grammar contains the rules $S \to b \,|\, aA$, $A \to c \,|\, bS$, then both production $S \to aA$ and $A \to bS$ are indirectly recursive:

$$S \Rightarrow a\underline{A} \Rightarrow abS,$$
$$A \Rightarrow b\underline{S} \Rightarrow baA.$$

Recursive grammar

A grammar is recursive if it contains either a recursive production or an indirectly recursive production.

A grammar for an infinite language must be directly or indirectly recursive!

Example. $S \to b \,|\, aA \,|\, cA$, $A \to c \,|\, bB$, $B \to aA \,|\, ac$

Recursive, infinite or not recursive, finite???
Recursive, infinite! $\ldots \{b, ac, cc, abac, cbac, ababac, \ldots\}$

Example. $S \to b \,|\, aA \,|\, bB$, $A \to c \,|\, bB \,|\, aB$, $B \to a \,|\, ba$

Recursive, infinite or not recursive, finite???
Not recursive, finite! $\ldots \{b, ba, bba, ac, abba, aaa, aaba\}$

Constructing grammars – finite languages

Now the opposite problem: finding a grammar for a given language.

Sometimes it is difficult or even impossible to write down a grammar for a given language. And not surprisingly, a language might have more than one grammar.

A simple case: finite languages

If the number of strings in a language is finite, then a grammar can consist of all productions of the form $S \to w$ for each string w in the language.

Example. The finite language $\{a, ba\}$ can be described by the grammar

$$S \to a \mid ba$$

Constructing grammars – infinite languages

A not so simple case: infinite languages

There is no universal method for finding a grammar for an infinite language, so we need to think :-) The method of combining grammars can be useful!

Example. Find a grammar for the following simple language: $\{\Lambda, a, aa, \ldots, a^n, \ldots\} = \{a^n \ : \ n \in \mathbb{N}\}$

A solution:

- the set of terminals: $T = \{a\}$,
- the only non-terminal start symbol S,
- the set of production rules:
  $S \to \Lambda$, $S \to aS$

Combining grammars

Suppose L and M are languages for which we are able to find the grammars. Then there exist simple rules for creating grammars which produce the languages $L \cup M$, $L \cdot M$ and L^*.

Idea: We can describe L and M with grammars having disjoint sets of non-terminals.

Assign the start symbols for the grammars of L and M to be A and B, respectively:

$$L : A \to \ldots, \qquad M : B \to \ldots$$

and then we combine in appropriate way both grammars to get the language, more in the following slides.

Union rule

The union of the two languages, $L \cup M$, starts with the two productions

$$S \to A \mid B$$

followed by

- the grammar rules of L (with the start symbol A) and
- M (with the start symbol B).

Union rule/example

Example. Suppose we want to write a grammar for the following language:

$$K = \{\Lambda, a, b, aa, bb, aaa, bbb, \ldots, a^n, b^n, \ldots\}.$$

K is the union of the two languages:

$$L = \{a^n \mid n \in \mathbb{N}\} \quad \text{and} \quad M = \{b^n \mid n \in \mathbb{N}\}.$$

Thus we can write a grammar for K as follows:

$A \to \Lambda \,|\, aA$ (grammar for L),
$B \to \Lambda \,|\, bB$ (grammar for M,)
$S \to A \,|\, B$ (union rule).

Product rule

Similarly, the product of the two languages, $L \cdot M$, starts with the production

$$S \to AB$$

followed by, as above,

- the grammar rules of L (with the start symbol A) and
- M (with the start symbol B).

Product rule/example

Example. Suppose we want to write a grammar for the following language:

$$K = \{a^m b^n \mid m, n \in \mathbb{N}\} = \{\Lambda, a, b, aa, ab, aaa, bb, \dots\}$$

K is the product of the two languages:

$L = \{a^n \mid n \in \mathbb{N}\}$ and $M = \{b^n \mid n \in \mathbb{N}\}$.

Thus we can write a grammar for K as follows:

$A \to \Lambda \,|\, aA$ (grammar for L),

$B \to \Lambda \,|\, bB$ (grammar for M,)

$S \to AB$ (product rule).

Closure rule

Finally, the grammar for the closure of a language, L^*, starts with the production

$$S \rightarrow AS \mid \Lambda$$

followed by

- the grammar rules of L (started from A).

Closure rule/example

Example. Suppose we want to construct the language L of all possible strings made up from zero or more occurrences of aa or bb.

$$L = \{aa, bb\}^* = M^*$$

where $M = \{aa, bb\}$.
Thus,

$$L = \{\Lambda, aa, bb, aaaa, aabb, bbbb, bbaa, \ldots\}$$

So we can write a grammar for L as follows:

$S \to AS \mid \Lambda$ (closure rule),
$A \to aa \mid bb$ (grammar for $\{aa, bb\}$).

Equivalent grammar

Grammars are not unique! A given language can have many grammars which could produce it.

We can simplify the previous grammar:

- Replace the occurrence of A in $S \to AS$ by the right side of $A \to aa$ to obtain the production $S \to aaS$.
- Replace A in $S \to AS$ by the right side of $A \to bb$ to obtain the production $S \to bbS$.
- This allows us to write the the grammar in simplified form as:

$$S \to aaS \mid bbS \mid \Lambda.$$

Some simple grammars

Language	Grammar
$\{a, ab, abb, abbb\}$	?
$\{\Lambda, a, aa, aaa, \ldots\}$	?
$\{b, bbb, bbbbb, \ldots, b^{2n+1}\}$	?
$\{b, abc, aabcc, \ldots, a^n bc^n\}$	?
$\{ac, abc, abbc, \ldots, ab^n c\}$	?

Given a simple language, you should be able to come up with a grammar to produce it!

Next lecture

We have discussed:

- grammars – sets of production rules for producing the strings of a language

In the next lecture we will discuss:

- a particularly simple subset (family) of languages: the regular languages