

G54FOP: Lecture 2

Basic Formal Language Notions

Henrik Nilsson

University of Nottingham, UK

G54FOP: Lecture 2 – p.1/28

Symbols and Alphabets

What is a symbol, then?

Anything, but it has to come from an **alphabet** Σ which is a **finite** set.

A common (and important) instance is $\Sigma = \{0, 1\}$.

ϵ , the empty word, is **never** an symbol of an alphabet.

G54FOP: Lecture 2 – p.4/28

All Words over an Alphabet (1)

Given an alphabet Σ we define the set Σ^* as set of words (or sequences) over Σ :

- The empty word $\epsilon \in \Sigma^*$.
- given a symbol $x \in \Sigma$ and a word $w \in \Sigma^*$, $xw \in \Sigma^*$.
- These are all elements in Σ^* .

This is called an **inductive definition**.

G54FOP: Lecture 2 – p.7/28

About These Slides

The following slides give a brief recap on some central notions from the theory of formal languages. This is all material that is covered in a typical undergraduate course on formal languages and automata theory, such as G52MAL here in Nottingham.

If you need more detail, I recommend the G52MAL lecture notes which are available on-line via

<http://www.cs.nott.ac.uk/~nhn/G52MAL>

G54FOP: Lecture 2 – p.5/28

Languages: Examples

alphabet $\Sigma = \{a, b\}$
words

Note the distinction between ϵ , \emptyset , and $\{\epsilon\}$!

G54FOP: Lecture 2 – p.8/28

All Words over an Alphabet (2)

Example: Given $\Sigma = \{0, 1\}$, some elements of Σ^* are

- ϵ (the empty word)
- 0, 1
- 00, 10, 01, 11
- 000, 100, 010, 110, 001, 101, 011, 111
- ...

We are just applying the inductive definition.

Note: although there are infinitely many words in Σ^* , each word has a **finite** length!

G54FOP: Lecture 2 – p.8/28

Languages

The terms **language** and **word** are used in a strict technical sense in this course:

- A **language** is a set of words.
- A **word** is a sequence (or string) of symbols.

ϵ denotes the **empty word**, the sequence of zero symbols.

The term **string** is often used interchangeably with the term **word**.

G54FOP: Lecture 2 – p.9/28

Exercises

- Is the set of natural numbers, \mathbb{N} , a possible alphabet? Why/why not?
- What about the set of all natural numbers smaller than some given number n ?
- Suggest an alphabet of a handful of **drink ingredients**. What are the symbols of your alphabet, and how many are they?
- List some words over your alphabet?
- What might an interesting language over your alphabet be? Does your language include **all** possible words over your alphabet?

G54FOP: Lecture 2 – p.6/28

Concatenation of Words (1)

An important operation on Σ^* is **concatenation**:

given $w, v \in \Sigma^*$, their concatenation $wv \in \Sigma^*$.

For example, concatenation of ab and ba yields $abba$.

This operation can be defined by primitive recursion:

$$\begin{aligned}\epsilon v &= v \\ (xw)v &= x(wv)\end{aligned}$$

G54FOP: Lecture 2 – p.9/28

Concatenation of Words (2)

Concatenation is associative and has unit ϵ :

$$u(vw) = (uv)w \\ \epsilon u = u = u\epsilon$$

where u, v, w are words.

054FOP: Lecture 2 - p.10/28

Examples of Languages (2)

- The set of words which contain the same number of 0s and 1s modulo 2 (i.e. both are even or odd) is a language over $\Sigma = \{0, 1\}$.
- The set of palindromes using the English alphabet, e.g. words which read the same forwards and backwards like `abba`. This is a language over $\{a, b, \dots, z\}$.
- The set of correct Java programs. This is a language over the set of UNICODE characters.

054FOP: Lecture 2 - p.13/28

Concatenation of Languages (2)

- Concatenation of languages is associative:

$$L(MN) = (LM)N$$

- Concatenation of languages has unit $\{\epsilon\}$:

$$L\{\epsilon\} = L = \{\epsilon\}L$$

- Concatenation distributes through set union:

$$L(M \cup N) = LM \cup LN \\ (L \cup M)N = LN \cup MN$$

054FOP: Lecture 2 - p.16/28

Languages Revisited

The notion of a language L of a set of words over an alphabet Σ can now be made precise:

- $L \subseteq \Sigma^*$, or equivalently
- $L \in \mathcal{P}(\Sigma^*)$.

054FOP: Lecture 2 - p.11/28

Examples of Languages (3)

- The set of programs that, if executed successfully on a Windows machine, prints the text "Hello World!" in a window. This is a language over $\Sigma = \{0, 1\}$.

054FOP: Lecture 2 - p.17/28

Concatenation of Languages (3)

- Concatenation distributes through set union:

$$L(M \cup N) = LM \cup LN \\ (L \cup M)N = LN \cup MN$$

But note e.g. $L(M \cap N) \neq LM \cap LN!$
For example, with $L = \{\epsilon, a\}$, $M = \{\epsilon\}$, $N = \{a\}$, we have

$$L(M \cap N) = L\emptyset = \emptyset \\ LM \cap LN = \{\epsilon, a\} \cap \{a, aa\} = \{a\}$$

054FOP: Lecture 2 - p.17/28

Examples of Languages (1)

Some examples of languages:

- The set $\{0010, 00000000, \epsilon\}$ is a language over $\Sigma = \{0, 1\}$. This is an example of a **finite** language.
- The set of words with odd length over $\Sigma = \{1\}$.
- The set of words that contain the same number of 0s and 1s is a language over $\Sigma = \{0, 1\}$.

054FOP: Lecture 2 - p.12/28

Concatenation of Languages (1)

Concatenation of words is extended to languages by:

$$MN = \{uv \mid u \in M \wedge v \in N\}$$

Example:

$$M = \{\epsilon, a, aa\} \\ N = \{b, c\} \\ MN = \\ = \\ =$$

054FOP: Lecture 2 - p.15/28

Context-Free Grammars (1)

A **Context-Free Grammar** (CFG) is a way of formally describing **Context-Free Languages** (CFL):

- The CFLs captures ideas common in programming languages such as
 - nested structure
 - balanced parentheses
 - matching keywords like `begin` and `end`.
- Most "reasonable" CFLs can be recognised by a fairly simple machine: a **deterministic pushdown automaton**.

054FOP: Lecture 2 - p.19/28

Context-Free Grammars (2)

Thus, describing a programming language by a "reasonable" CFG

- allows context-free constraints to be expressed
- imparts a hierarchical structure to the words in the language
- allows simple and efficient **parsing**:
 - determining if a word belongs to the language
 - determining its **phrase structure** if so.

054FOP: Lecture 2 - p.19/28

Context-Free Grammars: Notation

- Productions with the same LHS are usually grouped together. For example, the productions for S from the previous example:

$$S \rightarrow \epsilon \mid aA$$

This is (roughly) what is known as **Backus-Naur Form**.

- Another common way of writing productions is

$$A ::= \alpha$$

054FOP: Lecture 2 - p.22/28

The Derives Relation (1)

The relation $\xrightarrow[G]{*}$, read "**derives** in grammar G ", is the reflexive, transitive closure of $\xrightarrow[G]$.

That is, $\xrightarrow[G]{*}$ is the least relation on strings over $N \cup T$ such that:

- $\alpha \xrightarrow[G]{*} \beta$ if $\alpha \xrightarrow[G]{} \beta$
- $\alpha \xrightarrow[G]{*} \alpha$ (reflexive)
- $\alpha \xrightarrow[G]{*} \beta$ if $\alpha \xrightarrow[G]{*} \gamma \wedge \gamma \xrightarrow[G]{*} \beta$ (transitive)

054FOP: Lecture 2 - p.25/28

Context-Free Grammars (3)

A **Context-Free Grammar** is a 4-tuple (N, T, P, S) where

- N is a finite set of **nonterminals**
- T is a finite set of **terminals** (the **alphabet** of the language being described)
- $N \cap T = \emptyset$ (N and T are disjoint)
- S , the **start symbol**, is a distinguished element of N
- P is a finite set of **productions**, written $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (N \cup T)^*$

054FOP: Lecture 2 - p.20/28

The Directly Derives Relation (1)

To formally define the language generated by

$$G = (N, T, P, S)$$

we first define a binary relation $\xrightarrow[G]$ on strings over $N \cup T$, read "**directly derives** in grammar G ", being the least relation such that

$$\alpha A \gamma \xrightarrow[G]{} \alpha \beta \gamma$$

whenever $A \rightarrow \beta$ is a production in G .

Note: a production can be applied regardless of context, hence **context-free**.

054FOP: Lecture 2 - p.22/28

The Derives Relation (2)

Again, we use $\xrightarrow[G]{*}$ instead of $\xrightarrow[G]{*}$ when G is obvious.

Example: Given the grammar

$$\begin{aligned} S &\rightarrow \epsilon \mid aA \\ A &\rightarrow bS \end{aligned}$$

we have

$$\begin{aligned} S &\xrightarrow{*} \epsilon & S &\xrightarrow{*} abS \\ S &\xrightarrow{*} aA & S &\xrightarrow{*} ababS \\ aA &\xrightarrow{*} abS & S &\xrightarrow{*} abab \end{aligned}$$

054FOP: Lecture 2 - p.26/28

Context-Free Grammar: Example

$$G = (\{S, A\}, \{a, b\}, P, S)$$

where P consists of the productions

$$\begin{aligned} S &\rightarrow \epsilon \\ S &\rightarrow aA \\ A &\rightarrow bS \end{aligned}$$

054FOP: Lecture 2 - p.21/28

The Directly Derives Relation (2)

When it is clear which grammar G is involved, we use \Rightarrow instead of $\xrightarrow[G]$.

Example: Given the grammar

$$\begin{aligned} S &\rightarrow \epsilon \mid aA \\ A &\rightarrow bS \end{aligned}$$

we have

$$\begin{aligned} S &\Rightarrow \epsilon & aA &\Rightarrow abS \\ S &\Rightarrow aA & SaAa &\Rightarrow SabSaa \end{aligned}$$

054FOP: Lecture 2 - p.24/28

Language Generated by a Grammar

The language generated by a context-free grammar

$$G = (N, T, P, S)$$

denoted $L(G)$, is defined as follows:

$$L(G) = \{w \mid w \in T^* \wedge S \xrightarrow[G]{*} w\}$$

A language L is a **Context-Free Language** (CFL) iff $L = L(G)$ for some CFG G .

A string $\alpha \in (N \cup T)^*$ is a **sentential form** iff $S \xrightarrow{*} \alpha$.

054FOP: Lecture 2 - p.27/28

Language Generation: Example

Given the grammar
 $G = (N = \{S, A\}, T = \{a, b\}, P, S)$ where P are
the productions

$$\begin{aligned} S &\rightarrow \epsilon \mid aA \\ A &\rightarrow bS \end{aligned}$$

we have

$$\begin{aligned} L(G) &= \{(ab)^i \mid i \geq 0\} \\ &= \{\epsilon, ab, abab, ababab, abababab, \dots\} \end{aligned}$$