

Chapter 2: Formal Languages

In this chapter, we:

- Say what symbols, strings, alphabets and (formal) languages are;
- Introduce several string induction principles;
- Give an introduction to the Forlan toolset.

Copyright © 2003–5 Alley Stoughton

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

The \LaTeX source of these slides, the associated book, and the distribution of the Forlan toolset are available on the WWW at <http://people.cis.ksu.edu/~stough/forlan/>.

Section 2.1: Symbols, Strings, Alphabets and (Formal) Languages

In this section, we define the basic notions of the subject: symbols, strings, alphabets and (formal) languages.

In subsequent chapters, we will study four more restricted kinds of languages: the regular (Chapter 3), context-free (Chapter 4), recursive and recursively enumerable (Chapter 5) languages.

(2.1) Symbols

A *symbol* is one of the following finite sequences of ASCII characters:

- One of the *digits* 0–9;
- One of the *upper case letters* A–Z;
- One of the *lower case letters* a–z;
- A \langle , followed by any finite sequence of digits, letters, commas, \langle and \rangle , in which \langle and \rangle are properly nested, followed by a \rangle .

For example, $\langle id \rangle$ and $\langle \langle a, \rangle b \rangle$ are symbols. On the other hand, $\langle a \rangle \rangle$ is not a symbol since \langle and \rangle are not properly nested in $a \rangle$.

(2.1) Symbols (Cont.)

Whenever possible, we will use the mathematical variables a , b and c to name symbols. To avoid confusion, we will try to avoid situations in which we must simultaneously use the symbol a and the mathematical variable a .

(2.1) Symbols (Cont.)

Whenever possible, we will use the mathematical variables a , b and c to name symbols. To avoid confusion, we will try to avoid situations in which we must simultaneously use the symbol a and the mathematical variable a .

We write **Sym** for the set of all symbols. We order **Sym** by length (number of ASCII characters) and then lexicographically (in dictionary order). So, we have that

$$0 < \dots < 9 < A < \dots < Z < a < \dots < z,$$

and, e.g.,

$$z < \langle \text{be} \rangle < \langle \text{by} \rangle < \langle \text{on} \rangle < \langle \text{can} \rangle < \langle \text{con} \rangle.$$

(2.1) Symbols (Cont.)

Obviously **Sym** is infinite, but is it countably infinite?

(2.1) Symbols (Cont.)

Obviously **Sym** is infinite, but is it countably infinite?

To see that the answer is “yes”, let’s first see that it is possible to enumerate (list in some order, without repetition) all of the finite sequences of ASCII characters.

(2.1) Symbols (Cont.)

Obviously **Sym** is infinite, but is it countably infinite?

To see that the answer is “yes”, let’s first see that it is possible to enumerate (list in some order, without repetition) all of the finite sequences of ASCII characters.

We can list these sequences first according to length, and then according to lexicographic order. Thus the set of all such sequences is countably infinite. And since every symbol is such a sequence, it follows that **Sym** is countably infinite, too.

(2.1) Strings

A *string* is a finite sequence of symbols. We write the string with no symbols (the *empty string*) as ϵ , and ab , 0110 and $\langle id \rangle \langle num \rangle$ are other examples of strings.

Whenever possible, we will use the mathematical variables u , v , w , x , y and z to name strings.

(2.1) Strings

A *string* is a finite sequence of symbols. We write the string with no symbols (the *empty string*) as $\%$, and ab , 0110 and $\langle id \rangle \langle num \rangle$ are other examples of strings.

Whenever possible, we will use the mathematical variables u , v , w , x , y and z to name strings.

The *length* of a string x ($|x|$) is the number of symbols in the string. For example: $|\%| = 0$, $|ab| = 2$, $|0110| = 4$ and $|\langle id \rangle \langle num \rangle| =$.

(2.1) Strings

A *string* is a finite sequence of symbols. We write the string with no symbols (the *empty string*) as $\%$, and ab , 0110 and $\langle id \rangle \langle num \rangle$ are other examples of strings.

Whenever possible, we will use the mathematical variables u , v , w , x , y and z to name strings.

The *length* of a string x ($|x|$) is the number of symbols in the string. For example: $|\%| = 0$, $|ab| = 2$, $|0110| = 4$ and $|\langle id \rangle \langle num \rangle| = 2$.

(2.1) Strings

A *string* is a finite sequence of symbols. We write the string with no symbols (the *empty string*) as $\%$, and ab , 0110 and $\langle id \rangle \langle num \rangle$ are other examples of strings.

Whenever possible, we will use the mathematical variables u , v , w , x , y and z to name strings.

The *length* of a string x ($|x|$) is the number of symbols in the string. For example: $|\%| = 0$, $|ab| = 2$, $|0110| = 4$ and $|\langle id \rangle \langle num \rangle| = 2$.

We write **Str** for the set of all strings. We order **Str** first by length and then lexicographically, using our order on **Sym**. Thus, e.g.,

$$\% < ab < a\langle be \rangle < a\langle by \rangle < \langle can \rangle \langle be \rangle < abc.$$

(2.1) Strings

A *string* is a finite sequence of symbols. We write the string with no symbols (the *empty string*) as $\%$, and ab , 0110 and $\langle id \rangle \langle num \rangle$ are other examples of strings.

Whenever possible, we will use the mathematical variables u , v , w , x , y and z to name strings.

The *length* of a string x ($|x|$) is the number of symbols in the string. For example: $|\%| = 0$, $|ab| = 2$, $|0110| = 4$ and $|\langle id \rangle \langle num \rangle| = 2$.

We write \mathbf{Str} for the set of all strings. We order \mathbf{Str} first by length and then lexicographically, using our order on \mathbf{Sym} . Thus, e.g.,

$$\% < ab < a\langle be \rangle < a\langle by \rangle < \langle can \rangle \langle be \rangle < abc.$$

Since every string is a finite sequence of ASCII characters, it follows that \mathbf{Str} is countably infinite.

(2.1) Concatenation of Strings

The *concatenation* of strings x and y ($x @ y$) is the string consisting of the symbols of x followed by the symbols of y .

For example, $\% @ abc = abc$ and $01 @ 10 = 0110$.

(2.1) Concatenation of Strings

The *concatenation* of strings x and y ($x @ y$) is the string consisting of the symbols of x followed by the symbols of y .

For example, $\% @ abc = abc$ and $01 @ 10 = 0110$.

Concatenation is associative: for all $x, y, z \in \mathbf{Str}$,

$$(x @ y) @ z = x @ (y @ z).$$

And, $\%$ is the identify for concatenation: for all $x \in \mathbf{Str}$,

$$\% @ x = x @ \% = x.$$

(2.1) Concatenation of Strings (Cont.)

We often abbreviate $x @ y$ to xy . This abbreviation introduces some harmless ambiguity. For example, all of $0 @ 10$, $01 @ 0$ and $0 @ 1 @ 0$ are abbreviated to 010 . Fortunately, all of these expressions have the same value, so this kind of ambiguity is not a problem.

(2.1) Raising a String to a Power

We define the string x^n resulting from raising a string x to a power $n \in \mathbb{N}$ by recursion on n :

$$\begin{aligned}x^0 &= \%, \text{ for all } x \in \mathbf{Str}; \\x^{n+1} &= xx^n, \text{ for all } x \in \mathbf{Str} \text{ and } n \in \mathbb{N}.\end{aligned}$$

We assign this operation higher precedence than concatenation, so that xx^n means $x(x^n)$ in the above definition.

(2.1) Raising a String to a Power

We define the string x^n resulting from raising a string x to a power $n \in \mathbb{N}$ by recursion on n :

$$\begin{aligned}x^0 &= \%, \text{ for all } x \in \mathbf{Str}; \\x^{n+1} &= xx^n, \text{ for all } x \in \mathbf{Str} \text{ and } n \in \mathbb{N}.\end{aligned}$$

We assign this operation higher precedence than concatenation, so that xx^n means $x(x^n)$ in the above definition.

For example, we have that

$$(ab)^2 = (ab)(ab)^1 = (ab)(ab)(ab)^0 = (ab)(ab)\% = abab.$$

(2.1) Raising a String to a Power (Cont.)

Proposition 2.1.1

For all $x \in \mathbf{Str}$ and $n, m \in \mathbb{N}$, $x^{n+m} = x^n x^m$.

Proof. An easy mathematical induction on n . The string x and the natural number m can be fixed at the beginning of the proof. \square

(2.1) Raising a String to a Power (Cont.)

Proposition 2.1.1

For all $x \in \mathbf{Str}$ and $n, m \in \mathbb{N}$, $x^{n+m} = x^n x^m$.

Proof. An easy mathematical induction on n . The string x and the natural number m can be fixed at the beginning of the proof. \square

Thus, if $x \in \mathbf{Str}$ and $n \in \mathbb{N}$, then

$$x^{n+1} = x x^n \quad (\text{definition}),$$

and

$$x^{n+1} = x^n x^1 = x^n x \quad (\text{Proposition 2.1.1}).$$

(2.1) Prefixes, Suffixes and Substrings

Suppose x and y are strings. We say that:

- x is a *prefix* of y iff $y = xv$ for some $v \in \mathbf{Str}$;
- x is a *suffix* of y iff $y = ux$ for some $u \in \mathbf{Str}$;
- x is a *substring* of y iff $y = uxv$ for some $u, v \in \mathbf{Str}$.

A prefix, suffix or substring of a string other than the string itself is called *proper*.

For example:

- $\%$ is a *proper substring* of ab ;
- a is a *proper prefix* of ab ;
- b is a *proper suffix* of ab ;
- ab is a *substring* of ab .

(2.1) Prefixes, Suffixes and Substrings

Suppose x and y are strings. We say that:

- x is a *prefix* of y iff $y = xv$ for some $v \in \mathbf{Str}$;
- x is a *suffix* of y iff $y = ux$ for some $u \in \mathbf{Str}$;
- x is a *substring* of y iff $y = uxv$ for some $u, v \in \mathbf{Str}$.

A prefix, suffix or substring of a string other than the string itself is called *proper*.

For example:

- $\%$ is a proper prefix, suffix and substring of ab ;
- a is a *proper prefix* of ab ;
- b is a *proper suffix* of ab ;
- ab is a *proper substring* of ab .

(2.1) Prefixes, Suffixes and Substrings

Suppose x and y are strings. We say that:

- x is a *prefix* of y iff $y = xv$ for some $v \in \mathbf{Str}$;
- x is a *suffix* of y iff $y = ux$ for some $u \in \mathbf{Str}$;
- x is a *substring* of y iff $y = uxv$ for some $u, v \in \mathbf{Str}$.

A prefix, suffix or substring of a string other than the string itself is called *proper*.

For example:

- $\%$ is a proper prefix, suffix and substring of ab ;
- a is a proper prefix and substring of ab ;
- b is a suffix of ab ;
- ab is a proper substring of ab .

(2.1) Prefixes, Suffixes and Substrings

Suppose x and y are strings. We say that:

- x is a *prefix* of y iff $y = xv$ for some $v \in \mathbf{Str}$;
- x is a *suffix* of y iff $y = ux$ for some $u \in \mathbf{Str}$;
- x is a *substring* of y iff $y = uxv$ for some $u, v \in \mathbf{Str}$.

A prefix, suffix or substring of a string other than the string itself is called *proper*.

For example:

- $\%$ is a proper prefix, suffix and substring of ab ;
- a is a proper prefix and substring of ab ;
- b is a proper suffix and substring of ab ;
- ab is a of ab .

(2.1) Prefixes, Suffixes and Substrings

Suppose x and y are strings. We say that:

- x is a *prefix* of y iff $y = xv$ for some $v \in \mathbf{Str}$;
- x is a *suffix* of y iff $y = ux$ for some $u \in \mathbf{Str}$;
- x is a *substring* of y iff $y = uxv$ for some $u, v \in \mathbf{Str}$.

A prefix, suffix or substring of a string other than the string itself is called *proper*.

For example:

- $\%$ is a proper prefix, suffix and substring of ab ;
- a is a proper prefix and substring of ab ;
- b is a proper suffix and substring of ab ;
- ab is a (non-proper) prefix, suffix and substring of ab .

(2.1) Alphabets

An *alphabet* is a finite subset of **Sym**. We use Σ (upper case Greek letter sigma) to name alphabets.

For example, \emptyset , $\{0\}$ and $\{0, 1\}$ are alphabets.

We write **Alp** for the set of all alphabets. **Alp** is countably infinite.

(2.1) Alphabets

An *alphabet* is a finite subset of **Sym**. We use Σ (upper case Greek letter sigma) to name alphabets.

For example, \emptyset , $\{0\}$ and $\{0, 1\}$ are alphabets.

We write **Alp** for the set of all alphabets. **Alp** is countably infinite.

We define **alphabet** \in **Str** \rightarrow **Alp** by recursion:

$$\mathbf{alphabet}(\%) = \emptyset;$$

$$\mathbf{alphabet}(ax) = \{a\} \cup \mathbf{alphabet}(x), \text{ for all } a \in \mathbf{Sym} \text{ and } x \in \mathbf{Str}.$$

I.e., **alphabet**(w) consists of all of the symbols occurring in the string w . E.g., **alphabet**(01101) = $\{0, 1\}$.

(2.1) Alphabets (Cont)

If Σ is an alphabet, then we write Σ^* for

$$\{w \in \mathbf{Str} \mid \mathbf{alphabet}(w) \subseteq \Sigma\}.$$

I.e., Σ^* consists of all of the strings that can be built using the symbols of Σ .

For example, the elements of $\{0, 1\}^*$ are:

$$\%, 0, 1, 00, 01, 10, 11, 000, \dots$$

(2.1) (Formal) Languages

We say that L is a *formal language* (or just *language*) iff $L \subseteq \Sigma^*$, for some $\Sigma \in \mathbf{Alp}$. In other words, a language is a set of strings over some alphabet. If $\Sigma \in \mathbf{Alp}$, then we say that L is a Σ -*language* iff $L \subseteq \Sigma^*$.

(2.1) (Formal) Languages

We say that L is a *formal language* (or just *language*) iff $L \subseteq \Sigma^*$, for some $\Sigma \in \mathbf{Alp}$. In other words, a language is a set of strings over some alphabet. If $\Sigma \in \mathbf{Alp}$, then we say that L is a Σ -*language* iff $L \subseteq \Sigma^*$.

Here are some example languages (all are $\{0, 1\}$ -languages):

- \emptyset ;
- $\{0, 1\}^*$;
- $\{010, 1001, 1101\}$;
- $\{0^n 1^n \mid n \in \mathbb{N}\}$;
- $\{w \in \{0, 1\}^* \mid w \text{ is a palindrome}\}$.

(A palindrome is a string that reads the same backwards and forwards, i.e., that is equal to its own reversal.)

(2.1) More on Languages

Since **Str** is countably infinite and every language is a subset of **Str**, it follows that every language is countable. Furthermore, Σ^* is countably infinite, as long as the alphabet Σ is

(2.1) More on Languages

Since **Str** is countably infinite and every language is a subset of **Str**, it follows that every language is countable. Furthermore, Σ^* is countably infinite, as long as the alphabet Σ is nonempty.

(2.1) More on Languages

Since **Str** is countably infinite and every language is a subset of **Str**, it follows that every language is countable. Furthermore, Σ^* is countably infinite, as long as the alphabet Σ is nonempty.

We write **Lan** for the set of all languages. It turns out that **Lan** is uncountable. In fact even $\mathcal{P}(\{0, 1\}^*)$, the set of all $\{0, 1\}$ -languages, has the same size as $\mathcal{P}(\mathbb{N})$, and is thus uncountable.

(2.1) More on Languages

Since **Str** is countably infinite and every language is a subset of **Str**, it follows that every language is countable. Furthermore, Σ^* is countably infinite, as long as the alphabet Σ is nonempty.

We write **Lan** for the set of all languages. It turns out that **Lan** is uncountable. In fact even $\mathcal{P}(\{0, 1\}^*)$, the set of all $\{0, 1\}$ -languages, has the same size as $\mathcal{P}(\mathbb{N})$, and is thus uncountable.

Given a language L , we write **alphabet**(L) for the alphabet

$$\bigcup \{ \mathbf{alphabet}(w) \mid w \in L \}.$$

I.e., **alphabet**(L) consists of all of the symbols occurring in the strings of L . Note that, for all languages L , $L \subseteq \mathbf{alphabet}(L)^*$.

(2.1) More on Languages (Cont.)

If A is an infinite subset of **Sym** (and so is not an alphabet), we allow ourselves to write A^* for

$$\{x \in \mathbf{Str} \mid \mathbf{alphabet}(x) \subseteq A\}.$$

I.e., A^* consists of all of the strings that can be built using the symbols of A .

For example, $\mathbf{Sym}^* =$.

(2.1) More on Languages (Cont.)

If A is an infinite subset of **Sym** (and so is not an alphabet), we allow ourselves to write A^* for

$$\{x \in \mathbf{Str} \mid \mathbf{alphabet}(x) \subseteq A\}.$$

I.e., A^* consists of all of the strings that can be built using the symbols of A .

For example, $\mathbf{Sym}^* = \mathbf{Str}$.