Theory of Computation

Thomas Zeugmann

Hokkaido University
Laboratory for Algorithmics

http://www-alg.ist.hokudai.ac.jp/~thomas/ToC/

Lecture 1: Introducing Formal Languages

Motivation I

This course is about the study of a fascinating and important subject:

the theory of computation.

Motivation I

This course is about the study of a fascinating and important subject:

the theory of computation.

It comprises the fundamental mathematical properties of computer hardware, software, and certain applications thereof. We are going to determine what can and cannot be computed. If it can, we also seek to figure out on which type of computational model, how quickly, and with how much memory.

Motivation II

Theory of computation has many connections with engineering practice, and, as a true science, it also comprises philosophical aspects.

Motivation II

Theory of computation has many connections with engineering practice, and, as a true science, it also comprises philosophical aspects.

Since formal languages are of fundamental importance to computer science, we shall start our course by having a closer look at them.

Motivation II

Theory of computation has many connections with engineering practice, and, as a true science, it also comprises philosophical aspects.

Since formal languages are of fundamental importance to computer science, we shall start our course by having a closer look at them.

First, we clarify the subject of *formal language theory*.

Motivation III

Generally speaking, formal language theory concerns itself with sets of strings called *languages* and different mechanisms for generating and recognizing them. Mechanisms for generating sets of strings are usually referred to as *grammars* and mechanisms for recognizing sets of strings are called *acceptors* or *automata*.

Motivation III

Generally speaking, formal language theory concerns itself with sets of strings called *languages* and different mechanisms for generating and recognizing them. Mechanisms for generating sets of strings are usually referred to as *grammars* and mechanisms for recognizing sets of strings are called *acceptors* or *automata*.

A mathematical theory for generating and accepting languages emerged in the later 1950's and has been extensively developed since then. Nowadays there are elaborated theories for both computer languages and natural languages.

Motivation IV

We have to restrict ourselves to the most fundamental parts of formal language theory, i.e., to the regular languages, the context-free languages, and the recursively enumerable languages. This will suffice to obtain a basic understanding of what formal language theory is all about and what are the fundamental proof techniques.

Motivation IV

We have to restrict ourselves to the most fundamental parts of formal language theory, i.e., to the regular languages, the context-free languages, and the recursively enumerable languages. This will suffice to obtain a basic understanding of what formal language theory is all about and what are the fundamental proof techniques.

For having a common ground, we shortly recall the mathematical background needed.

Background I

For any set M, by $\mathrm{card}(M)$ and $\wp(M)$ we denote the *cardinality* of M and the *power set* of M, respectively.

Background I

For any set M, by $\mathrm{card}(M)$ and $\wp(M)$ we denote the *cardinality* of M and the *power set* of M, respectively.

Let $X,\ Y$ be any two sets; then we use $X \cup Y$, $X \cap Y$ and $X \setminus Y$ to denote the union, intersection and difference of X and Y, respectively.

Background I

For any set M, by $\mathrm{card}(M)$ and $\wp(M)$ we denote the *cardinality* of M and the *power set* of M, respectively.

Let X, Y be any two sets; then we use $X \cup Y$, $X \cap Y$ and $X \setminus Y$ to denote the union, intersection and difference of X and Y, respectively.

Furthermore, we denote the *empty set* by $\emptyset$.

Background I

For any set M, by $\mathrm{card}(M)$ and $\wp(M)$ we denote the *cardinality* of M and the *power set* of M, respectively.

Let $X,\ Y$ be any two sets; then we use $X \cup Y$, $X \cap Y$ and $X \setminus Y$ to denote the union, intersection and difference of X and Y, respectively.

Furthermore, we denote the *empty set* by $\emptyset$.

By $\mathbb{N} = \{0, 1, 2, \ldots\}$ we denote the *set of all natural numbers*. We set $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$.

Background II

If we have countably many sets $X_0, X_1, \ldots$, then we use $\bigcup_{i \in \mathbb{N}} X_i$ to denote the union of all X_i, i.e.,

$$\bigcup_{i \in \mathbb{N}} X_i \quad = \quad X_0 \cup X_1 \cup \cdots \cup X_n \cup \cdots \, . \tag{1}$$

Background II

If we have countably many sets $X_0, X_1, \ldots$, then we use $\bigcup_{i\in\mathbb{N}} X_i$ to denote the union of all X_i, i.e.,

$$\bigcup_{i\in\mathbb{N}} X_i \quad = \quad X_0 \cup X_1 \cup \cdots \cup X_n \cup \cdots \,. \tag{1}$$

Analogously, we write $\bigcap_{i\in\mathbb{N}} X_i$ to denote the intersection of all X_i, i.e.,

$$\bigcap_{i\in\mathbb{N}} X_i = X_0 \cap X_1 \cap \cdots \cap X_n \cap \cdots \,. \tag{2}$$

Background III

It is useful to have the following notions: Let X, Y be any sets and let $f\colon X \to Y$ be a function. For any $y \in Y$ we define

$$f^{-1}(y) = \{x \mid x \in X,\ f(x) = y\}\,. \tag{3}$$

We refer to $f^{-1}(y)$ as to the set of *pre-images* of y.

Background III

It is useful to have the following notions: Let X, Y be any sets and let $f\colon X \to Y$ be a function. For any $y \in Y$ we define

$$f^{-1}(y) = \{x \mid x \in X,\ f(x) = y\}\,. \tag{3}$$

We refer to $f^{-1}(y)$ as to the set of *pre-images* of y.
Also, we need the following definition:

Definition 1

Let X, Y be any sets and let $f\colon X \to Y$ be a function. The function f is said to be

(1) ***injective*** if $f(x) = f(y)$ implies $x = y$ for all $x, y \in X$;

(2) ***surjective*** if for every $y \in Y$ there is an $x \in X$ such that $f(x) = y$;

(3) ***bijective*** if f is injective and surjective.

Background IV

Next, let X and Y be any sets. We say that X and Y have the *same cardinality* if there exists a bijection $f\colon X \to Y$. If a set X has the same cardinality as the set $\mathbb{N}$ of natural numbers, then we say that X is *countably infinite*.

Background IV

Next, let X and Y be any sets. We say that X and Y have the *same cardinality* if there exists a bijection $f \colon X \to Y$. If a set X has the same cardinality as the set $\mathbb{N}$ of natural numbers, then we say that X is *countably infinite*.

A set X is *at most countably infinite* if it is finite or countably infinite. If $X \neq \emptyset$ and X is at most countably infinite, then there exists a surjection $f \colon \mathbb{N} \to X$, i.e.,

$$X = \{f(0), f(1), f(2), \ldots\} .$$

So, intuitively, we can enumerate all the elements of X (where repetitions are allowed).

Background V

The following theorem is of fundamental importance:

Theorem 1 (Cantor's Theorem)

For every countably infinite set X the set $\wp(X)$ is not countably infinite.

Background V

The following theorem is of fundamental importance:

Theorem 1 (Cantor's Theorem)

For every countably infinite set X *the set* $\wp(X)$ *is not countably infinite.*

Since X is countably infinite, there is a bijection $f\colon \mathbb{N} \to X$. Suppose that $\wp(X)$ is countably infinite. Then there must exist a bijection $g\colon \mathbb{N} \to \wp(X)$.

Background V

The following theorem is of fundamental importance:

Theorem 1 (Cantor's Theorem)

For every countably infinite set X the set $\wp(X)$ is not countably infinite.

Since X is countably infinite, there is a bijection $f\colon \mathbb{N} \to X$. Suppose that $\wp(X)$ is countably infinite. Then there must exist a bijection $g\colon \mathbb{N} \to \wp(X)$.
Now, we define a diagonal set D as follows:

$$D = \{f(j) \mid j \in \mathbb{N},\ f(j) \notin g(j)\}\ .$$

By construction, $D \subseteq X$ and hence $D \in \wp(X)$. Consequently, there must be a number $d \in \mathbb{N}$ such that $D = g(d)$.

Background VI

Now, we consider $f(d)$. By the definition of f we know that $f(d) \in X$. Since $D \subseteq X$, there are two possible cases, either $f(d) \in D$ or $f(d) \notin D$.

Background VI

Now, we consider $f(d)$. By the definition of f we know that $f(d) \in X$. Since $D \subseteq X$, there are two possible cases, either $f(d) \in D$ or $f(d) \notin D$.

Case 1. $f(d) \in D$.

By the definition of D and d we directly get

$$f(d) \in D \iff f(d) \notin g(d) \iff f(d) \notin D \,,$$

since $g(d) = D$. This contradiction shows that Case 1 cannot happen.

Background VII

Case 2. $f(d) \notin D$.

Again, by construction, $f(d) \notin D$ holds if and only if $f(d) \in g(d)$ if and only if $f(d) \in D$, again a contradiction.

Thus, Case 2 cannot happen either, and hence the supposition that $\wp(X)$ is countably infinite, cannot hold. $\blacksquare$

Relations I

Let X, Y be any non-empty sets. We set
$X \times Y = \{(x, y) \mid x \in X \text{ and } y \in Y\}$.
Every $\rho \subseteq X \times Y$ is said to be a *binary relation*.

We sometimes use the notation $x\rho y$ instead of writing $(x, y) \in \rho$. Of special importance is the case where $X = Y$. If $\rho \subseteq X \times X$ then we also say that *ρ is a binary relation over X*.

Relations II

Definition 2

Let $X \neq \emptyset$ be any set, and let ρ be any binary relation over X. The relation ρ is said to be

(1) *reflexive* if $(x, x) \in \rho$ for all $x \in X$;

(2) *symmetric* if $(x, y) \in \rho$ implies $(y, x) \in \rho$ for all $x, y \in X$;

(3) *transitive* if $(x, y) \in \rho$ and $(y, z) \in \rho$ implies $(x, z) \in \rho$ for all $x, y, z \in X$;

(4) *antisymmetric* if $(x, y) \in \rho$ and $(y, x) \in \rho$ implies $x = y$ for all $x, y, z \in X$.

Relations II

Definition 2

Let $X \neq \emptyset$ be any set, and let ρ be any binary relation over X. The relation ρ is said to be

(1) *reflexive* if $(x, x) \in \rho$ for all $x \in X$;

(2) *symmetric* if $(x, y) \in \rho$ implies $(y, x) \in \rho$ for all $x, y \in X$;

(3) *transitive* if $(x, y) \in \rho$ and $(y, z) \in \rho$ implies $(x, z) \in \rho$ for all $x, y, z \in X$;

(4) *antisymmetric* if $(x, y) \in \rho$ and $(y, x) \in \rho$ implies $x = y$ for all $x, y, z \in X$.

Any binary relation satisfying (1) through (3) is called *equivalence relation*. For any $x \in X$, we write $[x]$ to denote the *equivalence class* generated by x, i.e.,
$[x] = \{y \mid y \in X \text{ and } (x, y) \in \rho\}$.

Relations III

Any relation satisfying (1), (3) and (4) is called *partial order*. In this case, we also say that (X, ρ) is a *partially ordered set*.

Relations III

Any relation satisfying (1), (3) and (4) is called *partial order*. In this case, we also say that (X, ρ) is a *partially ordered set*.

Definition 3

Let $\rho \subseteq X \times Y$ and $\tau \subseteq Y \times Z$ be binary relations. The ***composition*** of ρ and τ is the binary relation $\zeta \subseteq X \times Z$ defined as

$$\begin{aligned} \zeta &= \rho\tau \\ &= \{(x, z) \mid \text{ there is a } y \in Y \text{ such that } (x, y) \in \rho,\ (y, z) \in \tau\} . \end{aligned}$$

Relations IV

Now, let $X \neq \emptyset$ be any set; there is a special binary relation ρ^0 called *equality*, and defined as $\rho^0 = \{(x, x) \mid x \in X\}$.
Moreover, let $\rho \subseteq X \times X$ be any binary relation. We inductively define $\rho^{i+1} = \rho^i \rho$ for each $i \in \mathbb{N}$.

Relations IV

Now, let $X \neq \emptyset$ be any set; there is a special binary relation ρ^0 called *equality*, and defined as $\rho^0 = \{(x, x) \mid x \in X\}$.
Moreover, let $\rho \subseteq X \times X$ be any binary relation. We inductively define $\rho^{i+1} = \rho^i \rho$ for each $i \in \mathbb{N}$.

Definition 4

Let $X \neq \emptyset$ be any set, and ρ be any binary relation over X. The ***reflexive-transitive closure*** of ρ is the binary relation $\rho^* = \bigcup_{i \in \mathbb{N}} \rho^i$.

Relations IV

Now, let $X \neq \emptyset$ be any set; there is a special binary relation ρ^0 called *equality*, and defined as $\rho^0 = \{(x, x) \mid x \in X\}$.
Moreover, let $\rho \subseteq X \times X$ be any binary relation. We inductively define $\rho^{i+1} = \rho^i \rho$ for each $i \in \mathbb{N}$.

Definition 4

Let $X \neq \emptyset$ be any set, and ρ be any binary relation over X. The ***reflexive-transitive closure*** of ρ is the binary relation $\rho^* = \bigcup_{i \in \mathbb{N}} \rho^i$.

Next, we introduce a formalism to deal with strings and sets of strings.

Alphabets

By Σ we denote a finite non-empty set called *alphabet*. The elements of Σ are assumed to be *indivisible symbols* and referred to as *letters* or *symbols*.

Examples:
$\Sigma = \{0, 1\}$ is an alphabet containing the letters 0 and 1, and
$\Sigma = \{a, b, c\}$ is an alphabet containing the letters a, b, and c.

In compiling, we may also have alphabets containing for example **begin** and **end**. But the letters **begin** and **end** are also assumed to be indivisible.

Alphabets

By Σ we denote a finite non-empty set called *alphabet*. The elements of Σ are assumed to be *indivisible symbols* and referred to as *letters* or *symbols*.

Examples:
$\Sigma = \{0, 1\}$ is an alphabet containing the letters 0 and 1, and
$\Sigma = \{a, b, c\}$ is an alphabet containing the letters a, b, and c.

In compiling, we may also have alphabets containing for example **begin** and **end**. But the letters **begin** and **end** are also assumed to be indivisible.

Alphabets

By Σ we denote a finite non-empty set called *alphabet*. The elements of Σ are assumed to be *indivisible symbols* and referred to as *letters* or *symbols*.

Examples:
$\Sigma = \{0, 1\}$ is an alphabet containing the letters 0 and 1, and
$\Sigma = \{a, b, c\}$ is an alphabet containing the letters a, b, and c.

In compiling, we may also have alphabets containing for example **begin** and **end**. But the letters **begin** and **end** are also assumed to be indivisible.

Strings

Definition 5

A ***string*** over an alphabet Σ is a finite length sequence of letters from Σ. A typical string is written as $s = a_1a_2\cdots a_k$, where $a_i \in \Sigma$ for $i = 1, \ldots, k$.

Strings

Definition 5

A ***string*** over an alphabet Σ is a finite length sequence of letters from Σ. A typical string is written as $s = a_1 a_2 \cdots a_k$, where $a_i \in \Sigma$ for $i = 1, \ldots, k$.

Note that we also allow $k = 0$ resulting in the *empty* string which we denote by λ. We call k the *length* of s and denote it by $|s|$, so $|\lambda| = 0$. By Σ^* we denote the set of all strings over Σ, and we set $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$.

Concatenation

Let $s, w \in \Sigma^*$; we define a binary operation called *concatenation* (or *word product*). The concatenation of s and w is the string sw.

Concatenation

Let $s, w \in \Sigma^*$; we define a binary operation called *concatenation* (or *word product*). The concatenation of s and w is the string sw.

Example: Let $\Sigma = \{0, 1\}$, $s = 000111$ and $w = 0011$; then $sw = 0001110011$.

Properties of Concatenation

Proposition 1

Let Σ *be any alphabet.*

(1) *Concatenation is associative, i.e., for all* $x, y, z \in \Sigma^*$, $x(yz) = (xy)z$.

(2) *The empty string* λ *is a two-sided identity for* Σ^**, i.e., for all* $x \in \Sigma^*$,

$$x\lambda = \lambda x = x\,.$$

(3) Σ^* *is free of nontrivial identities, i.e., for all* $x, y, z \in \Sigma^*$,

i) $zx = zy$ implies $x = y$ and,

ii) $xz = yz$ implies $x = y$.

(4) *For all* $x, y \in \Sigma^*$, $|xy| = |x| + |y|$.

Extension to Sets of Strings

Let X, Y be sets of strings. Then the *product* of X and Y is defined as

$$XY = \{xy \mid x \in X \text{ and } y \in Y\}.$$

Let $X \subseteq \Sigma^*$; define $X^0 = \{\lambda\}$ and for all $i \geqslant 0$ set $X^{i+1} = X^i X$. The *Kleene closure* of X is defined as $X^* = \bigcup_{i \in \mathbb{N}} X^i$, and the *semigroup closure* of X is $X^+ = \bigcup_{i \in \mathbb{N}^+} X^i$.
Finally, we define the *transpose* of a string and of sets of strings.

Extension to Sets of Strings

Let X, Y be sets of strings. Then the *product* of X and Y is defined as

$$XY = \{xy \mid x \in X \text{ and } y \in Y\}.$$

Let $X \subseteq \Sigma^*$; define $X^0 = \{\lambda\}$ and for all $i \geqslant 0$ set $X^{i+1} = X^i X$. The *Kleene closure* of X is defined as $X^* = \bigcup_{i \in \mathbb{N}} X^i$, and the *semigroup closure* of X is $X^+ = \bigcup_{i \in \mathbb{N}^+} X^i$.

Finally, we define the *transpose* of a string and of sets of strings.

Definition 6

Let Σ be any alphabet. The ***transpose*** operator is defined on strings in Σ^* and on sets $X \subseteq \Sigma^*$ of strings as follows:

$$\begin{aligned} \lambda^T &= \lambda\,, \quad \text{and} \\ (xa)^T &= a(x^T) \text{ for all } x \in \Sigma^* \text{ and all } a \in \Sigma \\ X^T &= \{x^T \mid x \in X\}\,. \end{aligned}$$

Languages I

Definition 7

Let Σ be any alphabet. Every subset $L \subseteq \Sigma^*$ is called *language*.

Languages I

Definition 7

Let Σ be any alphabet. Every subset $L \subseteq \Sigma^*$ is called ***language***.

Note that the empty set as well as $L = \{\lambda\}$ are also languages.

Languages I

Definition 7

Let Σ be any alphabet. Every subset $L \subseteq \Sigma^*$ is called *language*.

Note that the empty set as well as $L = \{\lambda\}$ are also languages.

Next, we ask how many languages there are. Let m be the cardinality of Σ. There is precisely one string of length 0, i.e., λ, there are m strings of length 1, i.e., a for all $a \in \Sigma$, there are m^2 many strings of length 2, and in general there are m^n many strings of length n. Thus, the cardinality of Σ^* is ***countably infinite***.

Languages II

By Cantor's theorem we know that $\mathrm{card}(M) < \mathrm{card}(\wp(M))$. So, we can conclude that there are ***uncountably*** many languages (as much as there are real numbers).

Since the generation and recognition of languages should be done algorithmically, we immediately see that only countably many languages can be generated and recognized by an algorithm.

Palindromes I

A ***palindrome*** is a string that reads the same from left to right and from right to left, e.g.,

AKASAKA

Palindromes I

A *palindrome* is a string that reads the same from left to right and from right to left, e.g.,

AKASAKA

トビコミコビト
にわとりとことりとわに
テングノハハノグンテ

Palindromes I

A *palindrome* is a string that reads the same from left to right and from right to left, e.g.,

AKASAKA

トビコミコビト
にわとりとことりとわに
テングノハハノグンテ

Now we ask how can we describe the language of all palindromes over the alphabet $\{a, b\}$ (just to keep it simple).

Palindromes II

So let us try it. Of course λ, a, and b are palindromes. Every palindrome must begin and end with the same letter, and if we remove the first and last letter of a palindrome, we still get a palindrome. This observation suggests the following basis and induction for defining L_{pal}:

Palindromes II

So let us try it. Of course λ, a, and b are palindromes. Every palindrome must begin and end with the same letter, and if we remove the first and last letter of a palindrome, we still get a palindrome. This observation suggests the following basis and induction for defining L_{pal}:

Induction Basis: λ, a, and b are palindromes.

Palindromes II

So let us try it. Of course λ, a, and b are palindromes. Every palindrome must begin and end with the same letter, and if we remove the first and last letter of a palindrome, we still get a palindrome. This observation suggests the following basis and induction for defining L_{pal}:

Induction Basis: λ, a, and b are palindromes.

Induction Step: If $w \in \{a, b\}^*$ is a palindrome, then awa and bwb are also palindromes. Furthermore, no string $w \in \{a, b\}^*$ is a palindrome, unless it follows from this basis and induction step.

Palindromes II

So let us try it. Of course λ, a, and b are palindromes. Every palindrome must begin and end with the same letter, and if we remove the first and last letter of a palindrome, we still get a palindrome. This observation suggests the following basis and induction for defining L_{pal}:

Induction Basis: λ, a, and b are palindromes.

Induction Step: If $w \in \{a, b\}^*$ is a palindrome, then awa and bwb are also palindromes. Furthermore, no string $w \in \{a, b\}^*$ is a palindrome, unless it follows from this basis and induction step.

But stop, we could have also used the transpose operator T to define the language of all palindromes, i.e.,
$\tilde{L}_{pal} = \{w \in \{a, b\}^* \mid w = w^T\}$.

Palindromes III

We used a different notation in the latter definition, since we still do not know whether or not $L_{pal} = \tilde{L}_{pal}$. For getting this equality, we need a proof.

Palindromes III

We used a different notation in the latter definition, since we still do not know whether or not $L_{pal} = \tilde{L}_{pal}$. For getting this equality, we need a proof.

Theorem 2

$L_{pal} = \tilde{L}_{pal}$.

Palindromes III

We used a different notation in the latter definition, since we still do not know whether or not $L_{pal} = \tilde{L}_{pal}$. For getting this equality, we need a proof.

Theorem 2

$L_{pal} = \tilde{L}_{pal}$.

Proof. Equality of sets X, Y is often proved by showing $X \subseteq Y$ and $Y \subseteq X$. So, let us first show that $L_{pal} \subseteq \tilde{L}_{pal}$.

Palindromes IV

We start with the strings defined by the basis, i.e., λ, a, and b. By the definition of the transpose operator, we have $\lambda^{\mathsf{T}} = \lambda$. Thus, $\lambda \in \tilde{L}_{pal}$. Next, we deal with a. In order to apply the definition of the transpose operator, we use Property (2) of Proposition 1, i.e., $a = \lambda a$. Then, we have

$$a^{\mathsf{T}} = (\lambda a)^{\mathsf{T}} = a\lambda^{\mathsf{T}} = a\lambda = a \ .$$

The proof for b is analogous and thus omitted.

Palindromes IV

We start with the strings defined by the basis, i.e., λ, a, and b. By the definition of the transpose operator, we have $\lambda^{\mathsf{T}} = \lambda$. Thus, $\lambda \in \tilde{L}_{pal}$. Next, we deal with a. In order to apply the definition of the transpose operator, we use Property (2) of Proposition 1, i.e., $a = \lambda a$. Then, we have

$$a^{\mathsf{T}} = (\lambda a)^{\mathsf{T}} = a\lambda^{\mathsf{T}} = a\lambda = a \, .$$

The proof for b is analogous and thus omitted.

Now, the induction hypothesis is that for all strings w with $|w| \leqslant n$, we have $w \in L_{pal}$ implies $w \in \tilde{L}_{pal}$. In accordance with our definition of L_{pal}, the induction step is from n to $n+2$. Let $w \in L_{pal}$ be any string with $|w| = n+2$. Thus, $w = ava$ or $w = bvb$, where $v \in \{a, b\}^*$ such that $|v| = n$. Then v is a palindrome in the sense of the definition of L_{pal}, and by the induction hypothesis, we know that $v = v^{\mathsf{T}}$.

Palindromes V

The following claims provide a special property of the transpose operator:

Claim 1. Let Σ *be any alphabet,* $n \in \mathbb{N}^+$*, and* $w = w_1 \dots w_n \in \Sigma^*$*, where* $w_i \in \Sigma$ *for all* $i \in \{1, \dots, n\}$*. Then* $w^T = w_n \dots w_1$.

Palindromes V

The following claims provide a special property of the transpose operator:

Claim 1. Let Σ *be any alphabet,* $n \in \mathbb{N}^+$*, and* $w = w_1 \dots w_n \in \Sigma^*$*, where* $w_i \in \Sigma$ *for all* $i \in \{1, \dots, n\}$*. Then* $w^T = w_n \dots w_1$*.*

Claim 2. For all $n \in \mathbb{N}$*, if* $p = p_1 x p_{n+2}$ *then* $p^T = p_{n+2} x^T p_1$ *for all* p_1, $p_{n+2} \in \{a, b\}$ *and* $x \in \{a, b\}^*$*, where* $|x| = n$*.*

Palindromes V

The following claims provide a special property of the transpose operator:

Claim 1. *Let* Σ *be any alphabet,* $n \in \mathbb{N}^+$, *and* $w = w_1 \ldots w_n \in \Sigma^*$, *where* $w_i \in \Sigma$ *for all* $i \in \{1, \ldots, n\}$. *Then* $w^T = w_n \ldots w_1$.

Claim 2. *For all* $n \in \mathbb{N}$, *if* $p = p_1 x p_{n+2}$ *then* $p^T = p_{n+2} x^T p_1$ *for all* $p_1, p_{n+2} \in \{a, b\}$ *and* $x \in \{a, b\}^*$, *where* $|x| = n$.

Note that Claim 1 is needed to show Claim 2. The proofs are left as exercise.

By using Claim 2 just established we get

$$w^T = (ava)^T = av^Ta \underbrace{=}_{\text{by IH}} ava = w\,.$$

Again, the case $w = bvb$ can be handled analogously and is thus omitted.

Palindromes VI

Finally, we have to show $\tilde{L}_{pal} \subseteq L_{pal}$.
For the induction basis, we know that $\lambda = \lambda^{T}$, i.e., $\lambda \in \tilde{L}_{pal}$ and by the induction basis of the definition of L_{pal}, we also know that $\lambda \in L_{pal}$.
Thus, we have the induction hypothesis that for all strings w of length n: if $w = w^{T}$ then $w \in L_{pal}$.

Palindromes VI

Finally, we have to show $\tilde{L}_{pal} \subseteq L_{pal}$.
For the induction basis, we know that $\lambda = \lambda^T$, i.e., $\lambda \in \tilde{L}_{pal}$ and by the induction basis of the definition of L_{pal}, we also know that $\lambda \in L_{pal}$.
Thus, we have the induction hypothesis that for all strings w of length n: if $w = w^T$ then $w \in L_{pal}$.

The induction step is from n to $n + 1$. That is, we have to show if $|w| = n + 1$ and $w = w^T$ then $w \in L_{pal}$.

Palindromes VI

Finally, we have to show $\tilde{L}_{pal} \subseteq L_{pal}$.
For the induction basis, we know that $\lambda = \lambda^T$, i.e., $\lambda \in \tilde{L}_{pal}$ and by the induction basis of the definition of L_{pal}, we also know that $\lambda \in L_{pal}$.
Thus, we have the induction hypothesis that for all strings w of length n: if $w = w^T$ then $w \in L_{pal}$.

The induction step is from n to $n + 1$. That is, we have to show if $|w| = n + 1$ and $w = w^T$ then $w \in L_{pal}$.

Since the case $n = 1$ directly results in a and b and since $a, b \in L_{pal}$, we assume $n > 1$ in the following.

Palindromes VII

So, let $w \in \{a, b\}^*$ be any string with $|w| = n + 1$ and $w = w^T$, say $w = a_1 \dots a_{n+1}$, where $a_i \in \Sigma$. Thus, by assumption we have

$$a_1 \dots a_{n+1} = a_{n+1} \dots a_1 \, .$$

Palindromes VII

So, let $w \in \{a, b\}^*$ be any string with $|w| = n + 1$ and $w = w^T$, say $w = a_1 \ldots a_{n+1}$, where $a_i \in \Sigma$. Thus, by assumption we have

$$a_1 \ldots a_{n+1} = a_{n+1} \ldots a_1 \ .$$

Now, applying Property (3) of Proposition 1 directly yields $a_1 = a_{n+1}$. We have to distinguish the cases $a_1 = a$ and $a_1 = b$. Since both cases can be handled analogously, we consider only the case $a_1 = a$ here. Thus, we can conclude $w = ava$, where $v \in \{a, b\}^*$ and $|v| = n - 1$. Next, applying the property of the transpose operator established above, we obtain $v = v^T$, i.e., $v \in L_{pal}$. Finally, the "induction" part of the definition of L_{pal} directly implies $w \in L_{pal}$. ∎

Thank you!