cs2001

Introduction Theory of Computation

Lecture 6

Ruth Urner

January 25, 2023

Mathematical statements and proofs

Reading: ITC Section 0.3 andd 0.4

Basic elements of mathematical text

Definition

A statement that cle[illegible]es an object/structure/concept based on previously d[illegible]ms.

Theorem

A statement that has been proven to be true.

Proof

A clean, deductive argument for why a statement is true.

Lemma

A "helper theorem", typically only stated as a step in a proof of some theorem.

Examples of what not to do..

Examples of what not to do..

Some comments

- Understanding a statement is not the same as understanding why the statement is true (or false). The first step in attempting to prove a statement, is alway[illegible]ure you understand the statement fully.
- When you attempt [illegible]mething, I recommend to always first develop an intuitio[illegible]tement and what may be the proof. Eg, first come up with some simple examples to illustrate the statement, then develop an intuition for why the statement is true, then develop a proof for it.
- Understanding whether a proof is correct and complete, is an important skill. It's important that you learn to evaluate whether your own proofs are correct.
- If you want to prove a statement, you need to provide a general argument. If you want to disprove a statement, you need to present a counter-example.
- Learning to prove mathematical statements is a skill that develops with practice. Be patient with yourself :)

Types of proofs–constructive proofs

If the statement that we are aiming to proof is a claim about existence of some object, then often we can prove the statement by constructing such an object.

Definition

For a natural number $k \in \mathbb{N}$, we call a graph $G = (V, E)$ a *k-regular graph* if eve[illegible] in V has degree k.

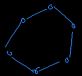
Theorem

For every even natural number $n \geq 4$, there exists a *3-regular graph* with n vertices.

Proof

Types of proofs–constructi

a 2-regular graph:



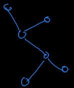
Definition

For a natural … we call a graph $G = (V, E)$ a *k*-regular grap… …x in V has degree k.

for claims of existence, often "proof by construction" is a useful technique

Theorem

For every even natural number $n \geq 4$, there exists a 3-regular graph with n vertices.



not a k-regular graph for any k

Proof (by construction).

Let $n \geq 4$ …

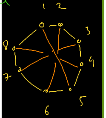
vertex set $V = [n] = \{1, 2, \ldots, n\}$.

We define the following …

$E = \{\{i, i+1\} \mid i \in \{1, \ldots, n-1\}\} \cup \{\{n, 1\}\}$

$\cup \{\{i, \frac{n}{2}+i\} \mid i \in \{1, 2, \ldots, \frac{n}{2}\}\}$

This results in a graph where every vertex has degree 3.



Talking:

Types of proofs–"by way of contradiction"

Sometimes, in order to prove that some statement is true, we assume that the statement is false and then show that this assumption leads to a contradiction. This, in turn, implies that the statement is true.

Types of proofs– "by way of contradiction"

Sometimes, in order to prove that some statement is true, we assume that the statement is false and then show that this assumption leads to a contradiction. This, in turn, implies that the statement is true.

To prove statement p,

$((\neg p) \Rightarrow F)$ is a true statement.

This then lets us conclude that $(\neg p)$ must be false, and the statement p must be true.

Talking: Ruth Urner

Types of proofs– examples of proofs “by way of contradiction”

Theorem

$\sqrt{2}$ is not a rational

Proof

See textbook.

Types of proofs– examples of proofs "by way of contradiction"

Theorem

There are infinitely m[illegible]e numbers.

Proof

Types of proofs– examples of proofs "by way of contradiction"

Theorem

There are infi[...]me numbers.

Proof

B.w.o.c. (by way of contradiction), let's assume the [...] is false, that is that there are only finitely, say $n \in \mathbb{N}$, many prime numbers [...] p_2, ..., p_n. (and let's assume $p_i > 1$ for all i).

Now, let's [...]

$$N = (p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_n) + 1$$

We have N [...] and none of the p_i divides N. Thus N must be a new prime number, a contradiction to the assumption that there are only n primes.

Thus, the assumption was false, and therefore there exist infinitely many prime numbers.

Types of proofs– examples of proofs “by way of contradiction”

Theorem

The set $\mathbb{R}$ of real numbers is not countable.

Proof

We’ll proves this in the tutorial on Friday.

Types of proofs–proof by (structural) induction

We can use a proof by induction, if we want to prove a statement about elements of a set that is (or can be) defined inductively.

Inductive definition of sets – motivating example

Say, I'd like to define [illegible] f all (biological) relatives of mine (living and dead ones [illegible]

- **I can not make** [illegible]
  (I don't know them all, especially not those that lived a thousand years ago..)

- **I can not give a precise characteristic**
  (Maybe I could if I was a biologist, but I am not..)

- **But I know some operations that will allow me to get from me to all of them!**
  The idea is to start with me, and consider everyone that can be reached by successively considering all children and all parents of previously reached people.

Inductive definition of sets – motivating example

Here is a more forma[illegible]efining all my biological relatives:

Consider the followin[illegible]components:

1. **Universe:** all people
2. **Core set:** me
3. **Operations:** parent-of, child-of

The set of all my relatives: Start with me, and successively apply the operations parent-of and child-of. The set of all my relatives are all people that can be "reached" this way.

Inductive definition of sets – general pattern

An inductive definition of a set consists of

1. **A universe set** U
2. **A core set** $C \subseteq U$
3. **A finite set** $O = \{o_1, o_2, \ldots, o_n\}$ **of operations from** $o_i : U^{r_i} \to U$ **for some arities** $r_i \in \mathbb{N}$

We define $\mathcal{I}(U, C, O)$ as the set of elements that we obtain by starting with the core set and putting all those elements of U into $\mathcal{I}(U, C, O)$ that one can reach by successively applying the operations in O.

Structural induction–general definition

Consider some inductively defined set $\mathcal{A} = \mathcal{I}(U, C, O)$. To show that all elements of $\mathcal{A}$ have property $\mathcal{P}$ we prove the following:

Base case Show that all elements $c \in C$ of the core set satisfy the property.

Induction hypothesis Assume that some $a_1, a_2, \ldots a_n \in I(U, C, O)$ satisfy the property (where n is the largest arity of the operations in O).

Induction step Show that for all operation $o_i \in O$, if the induction hypothesis holds, then the property also holds for

$$o_i(a_1, a_2, \ldots a_{r_i}).$$

Structural induction

- When proving something by (structural) induction, it is very important that you clearly state the hypothesis and make it clear to yourself where in the induction step you are actually using it. If it is not clear where you use it, there is likely something wrong with your proof..!

Game with cups

We consider three cups on a table as follows:

$$\cup \quad \cap \quad \cup$$

(That is, two upright and the middle one upside down.)

- We can now play with the cups by, at each step, flipping exactly two of them
- Eg, flipping the two left ones results in $\cap \ \cup \ \cup$

Question: Can we, by repeatedly flipping two cups, end up with all cups upright $\cup \ \cup \ \cup$?

Structural induction–example

First, we note that w ne the set of all reachable cup-configurations as tively defined set:

- Universe: $U_c =$ All ways to place three cups on the table.
  (Question for you: How big is this universe?)
- Coreset: The initial configuration, $C_c = \{\cup\cap\cup\}$
- Operations: $O_c = \{$flip-left-two, flip-outer-two, flip-right-two$\}$

Question: Is $\cup\cup\cup \in \mathcal{I}(U_c, C_c, O_c)$?

Structural induction–example

First, we note ... efine the set of all reachable cup-configura... ctively defined set:

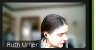
- Universe: U_c =All ways to place three cups on the table.
  (Question for you: How big is this universe?)
- Coreset: The initial configuration, $C_c = \{\cup\cap\cup\}$
- Operation: O_c = {flip-left-two, flip-outer-two, flip-right-two}

→ This defines (inductively) the set of all reachable states in this game.

Question: Is $\cup\cup\cup \in I(U_c, C_c, O_c)$?

→ No!

|| we'll prove that the number of upside cups is always even. ||



Tony Zhao

Shenice Thomas

Catherine Tighe

Conjecture: It is not [illegible]o get all cups upright..

We will prove the following property by induction:
In all reachable states the number of upright cups is even.

Since $\bigcup\bigcup\bigcup$ has an odd number of upright cups, this will imply that this state is not reachable.

Structural induction–example

2) if we flip two up cups, then XYZ must have had two up cups (since even), thus we end up with zero up-cups.
3) if we flip two down-cups, XYZ must have had 0 up-cups and we end up with 2 up cups

Proof by

Base case: ...itial configuration UUU, there are two (i.e. even number of U) of up-cups.

Induction hypothesis: Assume XYZ is a ... number of up-cups.

Induction step: ... XYZ, there are three cases: 1) if we flip one up and one down, the number of up cups stays the same.

Thus the property is maintained under the operation, and therefore holds for all reachable configurations by induction.

Structural induction–example

Property: The number of upright cups is even.

Proof by in induction:

Base case In the initial configuration $\cup \bigcirc \cup$, the property holds (2 cups are up, which is even).

Induction hypothesis Assume that for some configuration $XYZ \in \mathcal{I}(U_c, C_c, O_c)$ the number of up-cups is even.

Structural induction–example

Induction step First, note that if the number of up-cups in XYZ is even, it is either 0 or 2 (si[illegible] the only even numbers smaller than 3). This observati[illegible] following case distinction:

- **Case 1: It is 0** Then flipping two cups results in 2 up-cups, which is even again.

- **Case 2: It is 2** Then we either flip the two up-cups in XYZ or we flip one up-cup and one down-cup. In the first case, we end up with 0 up-cups, which is even, in the second case, we maintain 2 up-cups.

Thus in all cases the number of up-cups in flip-left-two(XYZ), flip-outer-two(XYZ), flip-right-two(XYZ) is even again.

Question for you: Where did we use the induction hypothesis?

Types of proofs–proof by induction

When we use a structural induction proof for the set of natural numbers, we often simply call it "proof by induction".

Example of proof by induction for natural numbers

Theorem

For a natural number $n \geq 1$, we let $S(n)$ denote the sum of the natural numbers up t… the following equality holds:

$$\frac{1}{2} \cdot n \cdot (n+1)$$

Proof

This is part of exercise 0.11, and we'll prove it in the Tutorial on Friday.

Example of proof by induction for natural numbers

Theorem

For a natural ... we let $S(n)$ denote the sum of the natural numb... en the following equality holds:

$$\ldots = \frac{1}{2} \cdot n \cdot (n+1)$$

Inductive definition of $\mathbb{N}$:

Universe: real numbers $\mathbb{R}$

Core set: $\{0\}$

Operation: $O_{+1} : n \mapsto n+1$

$O_{+1}(n) = n+1$

Talking: Ruth Urner

Example of proof by induction for natural numbers

Theorem

For a natural ... we let $S(n)$ denote the sum of the natural numb... en the following equality holds:

$$\ldots = \frac{1}{2} \cdot n \cdot (n+1)$$

Defining integers $\mathbb{Z}$ inductively:

Universe: $\mathbb{R}$

Core-set: $\{0\}$

Operations:

$o_{+1}: n \mapsto n+1$

$o_{-1}: n \mapsto n-1$

Inductive definition of $\mathbb{N}$:

Universe: real numbers $\mathbb{R}$

Core set: $\{0\}$

Operation: $o_{+1}: n \mapsto n+1$

$o_{+1}(n) = n+1$

Regular Languages

Finite Automata

Reading: ITC Section 1.1

Finite Automata

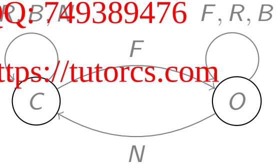
A finite automaton or finite state machine is a simple computational model.

We will work with this model of computation for the next part of this course.

A simple automaton–sliding door example

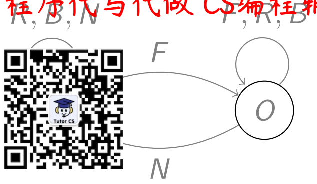
Consider an automatic sliding door sliding door with two pads that receive signals if someone is standing on them:

We can model the controller of the sliding door as a simple automaton:



Here we use: $C = CLOSED$, $O = OPEN$, $F = FRONT$, $R = REAR$, $B = BOTH$, $N = NEITHER$

A simple automaton–sliding door example



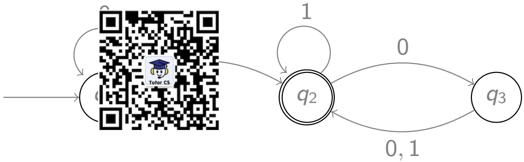
Here we use: $C = CLOSED$, $O = OPEN$, $F = FRONT$, $R = REAR$, $B = BOTH$, $N = NEITHER$

The behavior of the door can be described in terms of the following transition function:

	NEITHER	FRONT	REAR	BOTH
CLOSED	CLOSED	OPEN	CLOSED	CLOSED
OPEN	CLOSED	OPEN	OPEN	OPEN

State diagram of M_1

We can use a state diagram to describe a finite automaton M_1:



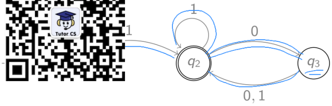
Interpretation of the state diagram: The arrow "coming out of nowhere" going into the leftmost state, signals, that this state is the start state. This automaton can read letters from the alphabet $\Sigma = \{0, 1\}$. Being in some state q, receiving letter σ, the computation finds the outgoing edge from q that has a label σ, and moves along that arrow to a new state.

Examples:

- If we feed the string 10010 to M_1, we move through the states $q_1, q_2, q_3, q_2, q_2, q_3$, and end up in state q_3, which is not an accept state.
- If we feed the string 1101 to M_1, we end up in state q_2, which is an accept state (accept states are the nodes with a double circle).
- If we feed the empty string ϵ to M_1, we end up in state q_1, which is not an accept state.

State diagram of M_1

We can use a st… scribe a finite automaton M_1:



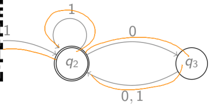
Interpretation of the state diagram: The arrow “coming out of nowhere” going into the leftmost state, signals, that this state is the start state. This automaton can read letters from the alphabet $\Sigma = \{0, 1\}$. Being in some state q, receiving letter σ, the computation finds the outgoing edge from q that has a label σ, and moves along that arrow to a new state.

Examples:

- If we feed the string 10010 to M_1

State diagram of M_1

We can use a st[illegible]scribe a finite automaton M_1:



Interpretation of the state diagram: The arrow “coming out of nowhere” going into the leftmost state, signals, that this state is the start state. This automaton can read letters from the alphabet $\Sigma = \{0, 1\}$. Being in some state q, receiving letter σ, the computation finds the outgoing edge from q that has a label σ, and moves along that arrow to a new state.

Examples:

- If we feed the string 10010 to M_1, we move through the states $q_1, q_2, q_3, q_2, q_2, q_3$, and end up in state q_3, which is not an accept state.
- If we feed the string 1101 to M_1, ends up in q_2, string is accepted.

Definition

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the set of states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Formal definition of a finite automaton

Definition

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the set of states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Formal description of M_1

The above state diagram corresponds to the following formal description:
$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

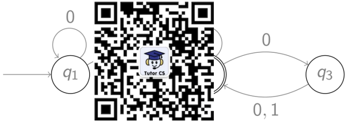
1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0, 1\}$,
3. δ is defined by the following table:

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

4. q_1 is the start state, and
5. $F \subseteq \{q_2\}$.

Given the description of an automaton, we can ask: which strings will lead to an accept state when fed into the automaton? As we have seen in the example computations with M_1 before, some strings do and others don't. The set of strings that do lead to an accept state form a language over Σ, the language of M_1.

Formal description of M_1



The above state diagram corresponds to the following formal description:
$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

$Q = \{q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

q_1 is starting state

$F = \{q_2\}$

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

Language accepted by an automaton

Let Σ be the alphabet of the automaton M. Then we let

$$L(M) = \{w \in \Sigma^k \mid k \in \mathbb{N} \text{ and } w \text{ is accepted by } M\}$$

denote the language of machine M. The is $L(M)$ is the set of all words over Σ that are accepted by machine M.

For the language $A = L(M)$ we also say machine M recognizes (or accepts) A.

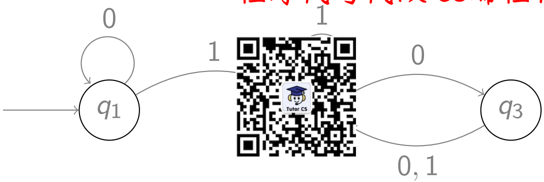
Formal definition of acceptance

Definition

Let $M = (Q, \Sigma, \delta, q_0, F)$ a finite automaton and $\mathbf{w} = w_1 w_2 \ldots w_n$ a string over Σ. We say that M **accepts w** if there exists a sequence $s_0 s_1 s_2 \ldots s_n$ of states such that

1. $s_0 = q_0$,
2. $\delta(s_i, w_{i+1}) = s_{i+1}$ for $i = 0, 1, \ldots, n-1$, and
3. $s_n \in F$.

Language accepted by M_1

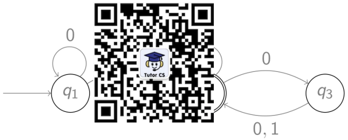


For the machine M_1 we get

$$L(M_1) =$$

Task for you: figure out what exactly is the set of words accepted by this automaton.

Language accepted by M_1



What is the language of M_1 ?

- all words that end with 1 are accepted

$\{ w = w_1 w_2 \dots w_n \in \Sigma^* \mid w_n = 1 \} \subseteq L(M_1)$

- but there are other words (e.g. 100) that don't end with 1 and are also accepted.