

# Finite Automata Theory and Formal Languages

## TMV027/DIT321– LP4 2018

### Lecture 3

Ana Bove

March 22nd 2018

## Recap: Logic, Sets, Relations, Functions

- Propositions, truth values, connectives, predicates, quantifiers;
- Sets, how to define them, membership, operations on sets, equality, laws;
- Relations, properties (reflexive, symmetric, antisymmetric, transitive, equivalence), partial vs total order, partitions, equivalence class, quotient;
- Functions, domain, codomain, image, partial vs total, injective, surjective, bijective, inverse, composition, restriction.

## Overview of Today's Lecture

- Formal proofs;
- Simple/strong induction;
- Mutual induction;
- Inductively defined sets;
- Recursively defined functions.

### Contributes to the following learning outcome:

- Prove properties of languages, grammars and automata with rigorously formal mathematical methods.

## How Formal Should a Proof Be?

Depends on the purpose but

- Should be convincing;
- Should not leave too much out;
- The validity of each step should be easily understood.

Valid steps are for example:

- Reduction to definition:  
“ $x$  is divisible by 2” is equivalent to “ $\exists k \geq 0. x = 2k$ ”;
- Use of hypotheses;
- Combining previous facts in a valid way:  
“Given  $A \Rightarrow B$  and  $A$  we can conclude  $B$  by *modus ponens*”.

## Form of Statements

Statements we want to prove are usually of the form

$$\text{If } \underbrace{H_1 \text{ and } H_2 \dots \text{ and } H_k}_{\text{hypotheses}} \text{ then } \underbrace{C_1 \text{ and } \dots \text{ and } C_m}_{\text{conclusions}}$$

or

$$P_1 \text{ and } \dots \text{ and } P_n \text{ iff } Q_1 \text{ and } \dots \text{ and } Q_m$$

for  $k \geq 0; n, m \geq 1$ .

**Note:** Observe that one proves the *conclusion* assuming the validity of the *hypotheses*!

**Example:** We can easily prove “if  $0 = 1$  then  $4 = 2.000$ ”.

## Different Kinds of Proofs

### Proofs by Contradiction

If  $H$  then  $C$

is logically equivalent to

$H$  and not  $C$  implies “the impossible” (bottom,  $\perp$ ).

**Example:** If  $x \neq 0$  then  $x^2 \neq 0$  vs.  $x \neq 0 \wedge x^2 = 0 \Rightarrow \perp$

### Proofs by Contrapositive

“If  $H$  then  $C$ ” is logically equivalent to “If not  $C$  then not  $H$ ”.

See both truth tables!

### Proofs by Counterexample

We find an example that “breaks” what we want to prove.

**Example:** All Natural numbers are odd.

# Proving a Property over the Natural Numbers

How to prove an statement over *all* the Natural numbers?

**Example:**  $\forall n \in \mathbb{N}. 1 + 2 + 3 + \dots + n = \frac{n * (n + 1)}{2}$ .

First we need to look at what the Natural numbers are ...

They are an *inductively defined set* defined by the following *two* rules:

$$\frac{}{0 \in \mathbb{N}} \qquad \frac{n \in \mathbb{N}}{n + 1 \in \mathbb{N}}$$

(More on inductively defined sets on slides 17–19.)

## Mathematical/Simple Induction

$$\frac{\text{base case} \quad \underbrace{\forall n \in \mathbb{N}. \overbrace{P(n)}^{\text{IH}} \Rightarrow P(n+1)}_{\text{inductive step}}}{\underbrace{\forall n \in \mathbb{N}. P(n)}_{\text{statement to prove}}}$$

More generally:

$$\frac{P(i), P(i + 1), \dots, P(j) \quad \forall n \in \mathbb{N}. j \leq n \Rightarrow \overbrace{P(n)}^{\text{IH}} \Rightarrow P(n + 1)}{\forall n \in \mathbb{N}. i \leq n \Rightarrow P(n)}$$

IH  $\equiv$  *inductive hypothesis*

## Example: Proof by Induction

**Proposition:** Let  $f(0) = 0$

$$f(n+1) = f(n) + n + 1.$$

Then,  $\forall n \in \mathbb{N}. f(n) = \frac{n * (n + 1)}{2}.$

**Proof:** By *mathematical induction* on  $n$ .

Let  $P(n)$  be  $f(n) = \frac{n * (n + 1)}{2}.$

**Base case:** We prove that  $P(0)$  holds.

**Inductive step:** We prove that if  $P(n)$  holds (our IH) for a given  $0 \leq n$ , then  $P(n+1)$  also holds.

**Closure:** Now we have established that for *all*  $n$ ,  $P(n)$  is true!

In particular,  $P(0), P(1), P(2), \dots, P(15), \dots$  hold.

## Course-of-Values/Strong Induction

Variant of mathematical induction.

$$\begin{array}{c}
 \text{base case} \quad \underbrace{\hspace{15em}}_{\text{inductive step}} \\
 \underbrace{P(0)} \quad \underbrace{\forall n \in \mathbb{N}. (\underbrace{\forall m \in \mathbb{N}. 0 \leq m \leq n \Rightarrow P(m)}_{\text{IH}})}_{\underbrace{\forall n \in \mathbb{N}. P(n)}_{\text{statement to prove}}} \Rightarrow P(n+1)
 \end{array}$$

Or more generally:

$$\frac{P(i), P(i+1), \dots, P(j) \quad \forall n \in \mathbb{N}. i \leq n \Rightarrow (\forall m. i \leq m \leq n \Rightarrow P(m)) \Rightarrow P(n+1)}{\forall n \in \mathbb{N}. i \leq n \Rightarrow P(n)}$$

Here we might have *several inductive hypotheses*  $P(m)$ !

## Example: Proof by Induction

**Proposition:** *If  $n \geq 8$  then  $n$  can be written as a sum of 3's and 5's.*

**Proof:** By *course-of-values induction* on  $n$ .

Let  $P(n)$  be “ $n$  can be written as a sum of 3's and 5's”.

**Base cases:**  $P(8)$ ,  $P(9)$  and  $P(10)$  hold.

**Inductive step:** We shall prove that if  $P(8), P(9), P(10), \dots, P(n)$  hold for  $n \geq 10$  (our IH) then  $P(n+1)$  holds.

Then every  $m$  with  $8 \leq m \leq n$  can be written as a sum of 3's and 5's.

Observe that if  $n \geq 10$  then  $n \geq n+1-3 \geq 8$ .

Hence by inductive hypothesis  $P(n+1-3)$  holds.

By adding an extra 3 then  $P(n+1)$  holds as well.

**Closure:**  $\forall n \geq 8$ .  $n$  can be written as a sum of 3's and 5's.

## Example: All Horses have the Same Colour



## Example: Proof by Induction

**Proposition:** *All horses have the same colour.*

**Proof:** By *mathematical induction* on  $n$ .

Let  $P(n)$  be “in any set of  $n$  horses they all have the same colour”.

**Base cases:**  $P(0)$  is not interesting in this example.  
 $P(1)$  is clearly true.

**Inductive step:** Let us show that  $P(n)$  (our IH) implies  $P(n + 1)$ .

Let  $h_1, h_2, \dots, h_n, h_{n+1}$  be a set of  $n + 1$  horses.

Take  $h_1, h_2, \dots, h_n$ . By IH they all have the same colour.

Take now  $h_2, h_3, \dots, h_n, h_{n+1}$ . Again, by IH they all have the same colour.

Hence, by transitivity, all horses  $h_1, h_2, \dots, h_n, h_{n+1}$  must have the same colour.

**Closure:**  $\forall n$ . all  $n$  horses in the set have the same colour.

## Example: What Went Wrong???



# Mutual Induction

Sometimes we cannot prove a single statement  $P(n)$  but rather a group of statements  $P'_1(n), P'_2(n), \dots, P'_k(n)$  *simultaneously* by induction on  $n$ .

In practice we have that  $P(n) = P'_1(n) \wedge P'_2(n) \wedge \dots \wedge P'_k(n)$ .

This is very common in automata theory where we need to prove an statement for each of the states in an automaton, or when we work with grammars where we need to prove an statement for each of the variables in a grammar.

## Example: Proof by Mutual Induction

Let  $f, g, h : \mathbb{N} \rightarrow \{0, 1\}$  be as follows:

$$\begin{array}{lll} f(0) = 0 & g(0) = 1 & h(0) = 0 \\ f(n+1) = g(n) & g(n+1) = f(n) & h(n+1) = 1 - h(n) \end{array}$$

Observe that the functions are defined by mutual dependency!

**Proposition:**  $\forall n. h(n) = f(n)$ .

**Proof:** If  $P(n)$  is " $h(n) = f(n)$ " then it is not possible to prove  $P(n) \Rightarrow P(n+1)$ .

We strengthen  $P(n)$  to  $P'(n)$ : Let  $P'(n)$  be " $h(n) = f(n) \wedge h(n) = 1 - g(n)$ ".

By mathematical induction on  $n$ .

We prove  $P'(0) : h(0) = f(0) \wedge h(0) = 1 - g(0)$ .

Then we prove that  $P'(n) \Rightarrow P'(n+1)$ .

Since by induction  $\forall n. P'(n)$  is true then  $\forall n. P(n)$  is also true.



## Recursive Data Types

What are (the data types of) Natural numbers, lists, trees, ... ?

This is how you would defined them in Haskell:

```
data Nat = Zero | Succ Nat
```

```
data List a = Nil | Cons a (List a)
```

```
data BTree a = Leaf a | Node a (BTree a) (BTree a)
```

They are defined in terms of themselves!

Observe the similarity between the definition of  $\mathbb{N}$ at above and the rules in slide 6...

## Inductively Defined Sets

### Natural Numbers:

*Base case:* 0 is a Natural number;

*Inductive step:* If  $n$  is a Natural number then  $n + 1$  is a Natural number;

*Closure:* There is no other way to construct Natural numbers.

### Finite Lists:

*Base case:*  $[]$  is the empty list over any set  $A$ ;

*Inductive step:* If  $x \in A$  and  $xs$  is a list over  $A$  then  $x : xs$  is a list over  $A$ ;

*Closure:* There is no other way to construct lists.

### Finitely Branching Trees:

*Base case:* If  $x \in A$  then  $(x)$  is a tree over any set  $A$ ;

*Inductive step:* If  $x \in A$  and  $t_1, \dots, t_k$  are tree over the set  $A$ ,  
then  $(x, t_1, \dots, t_k)$  is a tree over  $A$ ;

*Closure:* There is no other way to construct trees.

⋮

## Inductively Defined Sets (Cont.)

To define a set  $S$  by induction we need to specify:

**Base cases:**  $e_1[\dots], \dots, e_m[\dots] \in S$ ;

**Inductive steps:** Given  $\dots$  and given  $s_1, \dots, s_{k_i} \in S$ ,  
then  $c_1[\dots, s_1, \dots, s_{k_1}], \dots, c_n[\dots, s_1, \dots, s_{k_n}] \in S$ ;

**Closure:** There is no other way to construct elements in  $S$ .  
(We will usually omit this part.)

**Note:** Each base element  $e_i$  or each recursive element  $c_i$  could also depend on other (previously defined) sets  $A$ 's. Denoted as  $\dots$  above.

**Example:** See the case of lists and trees with elements of a certain type  $A$  in previous slide.

See notes on *Inductive sets and induction* linked from course web page.

## Examples of Inductively Defined Sets

**Example:** The set of simple Boolean expressions is defined as:

*Base cases:* true and false are Boolean expressions;

*Inductive steps:* if  $a$  and  $b$  are Boolean expressions then

(a)    not  $a$      $a$  and  $b$      $a$  or  $b$

are also Boolean expressions.

**Example:** The set of simple data types:

*Base cases:*  $\mathbb{N}$ , Bool and String are types;

*Inductive steps:* if  $A$  and  $B$  are types then

$A \rightarrow B$      $A \times B$      $A + B$

are also types.

## Recursive Functions over Inductively Defined Sets

We define a (terminating) function  $f : S \rightarrow A$  over an inductively defined set where the recursive calls are on *structural smaller* elements as follows:

**Base cases:**  $f(e_1[\dots]), \dots, f(e_m[\dots]) \in A$ ;

We define  $f$  on all base elements.

**Recursive cases:** Given  $\dots$  and given  $s_1, \dots, s_{k_i} \in S$ ,

$$\begin{aligned} f(c_1[\dots, s_1, \dots, s_{k_1}]) &= h_1[\dots, f(s_1), \dots, f(s_{k_1})] \\ &\vdots \\ f(c_n[\dots, s_1, \dots, s_{k_n}]) &= h_n[\dots, f(s_1), \dots, f(s_{k_n})] \end{aligned}$$

We define  $f$  on each “complex” element in terms of the result of  $f$  on the structurally smaller elements!

See notes on *Inductive sets and induction* linked from course web page.

## Example: Recursive Functions over Lists

Recall lists are either  $[]$  (empty) or  $x : xs$  (not empty).  $[x]$  denotes  $x : []$ .

**Example:** Let us (recursively) define the length of a list.

$$\begin{aligned} \text{len } [] &= 0 \\ \text{len } (x : xs) &= 1 + \text{len } xs \end{aligned}$$

**Example:** Let us (recursively) define the append over lists.

$$\begin{aligned} [] ++ ys &= ys \\ (x : xs) ++ ys &= x : (xs ++ ys) \end{aligned}$$

**Example:** Let us (recursively) define the reverse of a list.

$$\begin{aligned} \text{rev } [] &= [] \\ \text{rev } (x : xs) &= \text{rev } xs ++ [x] \end{aligned}$$

## Overview of Next Lecture

Chapter 5 in the *Mathematics for Computer Science* book, section 1.2 in the main book, and notes on *Inductive sets and induction*:

- Structural induction;
- Concepts of automata theory.

See also even Claessen's notes on proof methods linked from course web page.

**DO NOT MISS THIS LECTURE!!!**