Finite Automata Theory and Formal Languages

TMV027/DIT321– LP4 2017

Lecture 3
Ana Bove

March 23rd 2017

Overview of today's lecture:

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

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.

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 "$\exists k \geqslant 0. x = 2k$";
- 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

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

or

$$P_1 \text{ and } \ldots \text{and } P_k \text{ iff } Q_1 \text{ and } \ldots \text{and } Q_m$$

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

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

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 page 16.)

Mathematical/Simple Induction

$$\frac{\overbrace{P(0)}^{\text{base case}} \qquad \overbrace{\forall n \in \mathbb{N}.\ \overbrace{P(n)}^{\text{IH}} \Rightarrow P(n+1)}^{\text{inductive step}}}{\underbrace{\forall n \in \mathbb{N}.\ P(n)}_{\text{statement to prove}}}$$

More generally:

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

IH $\equiv$ *inductive hypothesis*

Example: Proof by Induction

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

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

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

Proof: By *mathematical induction* on n.

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

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

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

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

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

Course-of-Values/Strong Induction

Variant of mathematical induction.

$$\frac{\overbrace{P(0)}^{\text{base case}} \qquad \overbrace{\forall n \in \mathbb{N}.\ (\overbrace{\forall m \in \mathbb{N}.\ 0 \leqslant m \leqslant n \Rightarrow P(m)}^{\text{IH}}) \Rightarrow P(n+1)}^{\text{inductive step}}}{\underbrace{\forall n \in \mathbb{N}.\ P(n)}_{\text{statement to prove}}}$$

Or more generally:

$$\frac{\begin{array}{c} P(i), P(i+1), \ldots, P(j) \\ \forall n \in \mathbb{N}.\ i < n \Rightarrow (\forall m.\ i \leqslant m < n \Rightarrow P(m)) \Rightarrow P(n) \end{array}}{\forall n \in \mathbb{N}.\ i \leqslant n \Rightarrow P(n)}$$

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

Example: Proof by Induction

Proposition: *If $n \geqslant 8$ 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 $P(8), P(9), P(10), \ldots, P(n)$ hold for $n \geqslant 10$ (our IH) then $P(n+1)$ holds.

Observe that if $n \geqslant 10$ then $n \geqslant n+1-3 \geqslant 8$.
Hence by inductive hypothesis $P(n+1-3)$ holds.
By adding an extra 3 then $P(n+1)$ holds as well.

Closure: $\forall n \geqslant 8$. 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 $P_1(n), P_2(n), \ldots, P_k(n)$ *simultaneously* by induction on n.

This is very common in automata theory where we need an statement for each of the states of the automata.

Example: Proof by Mutual Induction

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

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

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

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

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 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)
```

Observe the similarity between the definition of `Nat` above and the rules in slide 5...

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 $t_1, \ldots, t_k$ are tree over the set A,
then $(x, t_1, \ldots, t_k)$ is a tree over A;
Closure: There is no other way to construct trees.

$\vdots$

Inductively Defined Sets (Cont.)

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

Base cases: $e_1, \ldots, e_m \in S$;

Inductive steps: Given $s_1, \ldots, s_{n_i} \in S$,
then $\mathrm{c}_1[s_1, \ldots, s_{n_1}], \ldots, \mathrm{c}_\mathrm{k}[s_1, \ldots, s_{n_k}] \in S$;

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

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.

Recursive Functions over Inductively Defined Sets

To define a function $f : S \to A$ over an inductively defined set we need to:

Base cases: $f(e_1), \ldots, f(e_m) \in A$;

We define f on all base elements:

Recursive cases: Given $s_1, \ldots, s_{n_i} \in S$,

$$
\begin{array}{rcl}
f(\mathsf{c}_1[s_1, \ldots, s_{n_1}]) & = & h_1[f(s_1), \ldots, f(s_{n_1})] \\
\vdots & = & \vdots \\
f(\mathsf{c}_\mathsf{k}[s_1, \ldots, s_{n_k}]) & = & h_k[f(s_1), \ldots, f(s_{n_k})]
\end{array}
$$

We define f on each "complex" element in terms of the result of f on the *structurally smaller* elements!

Example: Recursive Functions over Lists

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

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

$$
\begin{array}{l}
\mathsf{len}\ [] = 0 \\
\mathsf{len}\ (x : xs) = 1 + \mathsf{len}\ xs
\end{array}
$$

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

$$
\begin{array}{l}
[] +\!\!+\ ys = ys \\
(x : xs) +\!\!+\ ys = x : (xs +\!\!+\ ys)
\end{array}
$$

Overview of Next Lecture

Chapter 5 in the *Mathematics for Computer Science* book and section 1.2 in the main book:

- Structural induction;
- Concepts of automata theory.

See even Koen Claessen's notes on structural induction (see course web page on Literature).

DO NOT MISS THIS LECTURE!!!