



## Programming Language Semantics and Compiler Design / Sémantique des Langages de Programmation et Compilation

Maths Reminders

Yliès Falcone

ylies.falcone@univ-grenoble-alpes.fr — www.ylies.fr  
Univ. Grenoble Alpes, and LIG-Inria team CORSE

Master of Sciences in Informatics at Grenoble (MoSIG)  
Master 1 info

Univ. Grenoble Alpes - UFR IM<sup>2</sup>AG  
www.univ-grenoble-alpes.fr - im2ag.univ-grenoble-alpes.fr

Academic Year 2017 - 2018

### Some proof techniques

Proofs by contradiction, reducto-ad-absurdum, contraposition,...

...they rely on the principles of propositional and predicate logics.

### Proof by structural induction

- ▶ Proof for the basic elements, atoms, of the set.
- ▶ Proof for composite elements (created by applying) rules:
  - ▶ assume it holds for the immediate components (**induction hypothesis**)
  - ▶ prove the property holds for the composite element

### Induction on the shape of a derivation tree

- ▶ Proof for 'one-rule' derivation trees, i.e., axioms.
- ▶ Proof for composite trees:
  - ▶ For each rule  $R$ , consider a composite tree where  $R$  is the last rule applied
  - ▶ Proof for the composite tree
    - ▶ Assume it holds for subtrees, or premises of the rule (**induction hypothesis**)
    - ▶ Proof for the composite tree

### Outline - Maths Reminders

Proof by induction

Proof by structural induction

A notation: derivation tree

### Outline - Maths Reminders

Proof by induction

Proof by structural induction

A notation: derivation tree

## Proof by induction

Proving a predicate  $P(n)$  that depends on some parameter  $n \in \mathbb{N}$ .

### Example (Predicate)

- ▶  $P(n)$  = The sum of the  $n$  natural numbers is  $\frac{n \times (n+1)}{2}$ ."
- ▶  $P(n)$  = "If  $q \geq 2$ , we have  $n \leq q^n$ ."
- ▶  $P(n)$  = "Every polynomial of degree  $n$  has at most  $n$  roots."

We want to prove  $\forall n \in \mathbb{N} : P(n)$ .

### Principle

- ▶ Prove the **base case**: prove that  $P(0)$  holds (or  $P(k)$  if the minimal value of the parameter is  $k$ ).
- ▶ Prove the **induction step**: prove that if for some  $n$ ,  $P(n)$  holds, then  $P(n+1)$  holds.

The principle of induction ensures that  $P(n)$  holds, for any  $n \geq k$ .

## Proof by induction (example)

### Example (Proof by induction)

Let's prove that

$$\forall n \in \mathbb{N} : \sum_{i=0}^n i = \frac{n(n+1)}{2}$$

- ▶ Base case:  $\sum_{i=0}^n i = 0$ .
- ▶ Induction step.
  - ▶ Suppose that the property holds for some  $n \in \mathbb{N}$ .
  - ▶ We have:

$$\begin{aligned} \sum_{i=0}^{n+1} i &= \sum_{i=0}^n i + n + 1 \\ &= \frac{n(n+1)}{2} + n + 1 \quad (\text{induction hypothesis}) \\ &= \frac{(n+2) \times (n+1)}{2} \end{aligned}$$

## Proof by complete induction

Proving a predicate  $P(n)$  that depends on some parameter  $n \in \mathbb{N}$ .

That is, we want to prove  $\forall n \in \mathbb{N} : P(n)$ .

### Principle of complete induction

- ▶ Prove the **base case**: prove that  $P(0)$  holds (or  $P(k)$  if the minimal value of the parameter is  $k$ ).
- ▶ Prove the **complete induction step**: prove that if for some  $n$ ,  $\forall m \leq n : P(m)$  holds, then  $P(n+1)$  holds.

The principle of induction ensures that  $P(n)$  holds for any  $n \geq k$ .

## Proof by complete induction: example

The  $n$ -th fibonacci number  $f_n$  is defined as follows:

- ▶  $f_0 = 0$
- ▶  $f_1 = 1$
- ▶  $\forall k \geq 2 : f_k = f_{k-1} + f_{k-2}$

Let us prove that the  $n$ -th Fibonacci number is even iff  $n$  is a multiple of 3.

### Example (The $n$ -th Fibonacci number is even iff $n$ is a multiple of 3.)

- ▶ Base case. We can see that it holds for  $n = 0$ .
- ▶ complete induction step.
  - ▶ Let us suppose that the property holds for any integer lesser than or equal to some  $n \in \mathbb{N}$ .
  - ▶ Consider  $f_{n+1}$  and distinguish three cases according to the rest of the division of  $n+1$  by 3.
    - ▶ Case  $n+1 \bmod 3 = 0$ .  $f_{3k} = f_{3k-1} + f_{3k-2} = \text{odd} + \text{odd} = \text{even}$
    - ▶ Case  $n+1 \bmod 3 = 1$ .  $f_{3k+1} = f_{3k} + f_{3k-1} = \text{even} + \text{odd} = \text{odd}$
    - ▶ Case  $n+1 \bmod 3 = 2$ .  $f_{3k+2} = f_{3k+1} + f_{3k} = \text{odd} + \text{even} = \text{odd}$

## Outline - Maths Reminders

Proof by induction

Proof by structural induction

A notation: derivation tree

## Inductive/Compositional definitions

Let us consider:

- ▶  $E$  a set ,
- ▶  $f : E \times E \times \dots \times E \rightarrow E$  a partial function,
- ▶  $A \subseteq E$  a subset of  $E$ .

### Definition (closure)

$A$  is closed by  $f$  iff  $f(A \times \dots \times A) \subseteq A$ .

### Definition (Construction rule)

A construction rule for a set states either:

- ▶ that a *basis element* belongs to the set, or
- ▶ how to produce a new element from existing elements (*production rule* given by a partial function).

### Definition (Inductive definition)

An inductive definition on  $E$  is a family of rules defining the smallest subset of  $E$  that is *closed* by these rules.

## Inductive definitions: examples

### Example (Natural numbers)

How can define them?

- ▶ basis element 0
- ▶ 1 rule:  $x \mapsto \text{succ}(x)$

2 is the natural number defined as  $\text{succ}(\text{succ}(0))$

### Example (Even numbers)

- ▶ basis element 0
- ▶ 1 rule  $x \mapsto x + 2$

### Example (Palindromes on $\{a, b\}$ )

- ▶ basis elements  $\epsilon, a, b$
- ▶ 2 rules:  $w \mapsto a \cdot w \cdot a$ ,  $w \mapsto b \cdot w \cdot b$

## Binary trees

### Definition and examples

### Definition (Binary Tree – Informal definition)

A tree is a **binary tree** if each node has *at most two children* (possibly empty).

### Definition (Binary Tree – Mathematical definition)

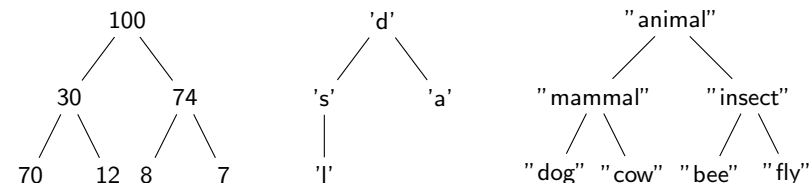
The smallest set  $Bt(Elt)$  s.t.:

$$Bt(Elt) = \{EmptyT\} \cup \{Node(tL, e, tR) \mid e \in Elt \wedge tL, tR \in Bt(Elt)\}$$

### Example (Binary trees of natural numbers)

$$Bt(\mathbb{N}) = \{EmptyT\} \cup \{Node(tL, e, tR) \mid e \in \mathbb{N} \wedge tL, tR \in Bt(\mathbb{N})\}$$

### Example (Binary trees)



## Proof by Structural Induction

Proving that the proof holds for any element "however it is built".

### Principle

- ▶ Proof for the basic elements, atoms, of the set.
- ▶ Proof for composite elements (created by applying) rules:
  - ▶ assume it holds for the immediate components (**induction hypothesis**)
  - ▶ prove the property holds for the composite element

## Proof by Structural Induction: example

### Example (Proofs by induction)

All proofs by induction are proofs by structural induction where the inductive set is  $\mathbb{N}$ .

### Example (Properties of size and depth of a binary tree)

Let us consider  $t \in Bt(El)$ , a binary tree:

- ▶  $depth(t)$  be the depth of tree  $t$ : length of longest path from root to leaf.
- ▶  $size(t)$  be the size of tree  $t$ : number of nodes + leaves.

For any type  $El$  and any  $t \in Bt(El)$ :

- ▶  $depth(t) \leq size(t)$ ,
- ▶  $size(t) \leq 2^{depth(t)-1}$ .

## Inductive definitions: examples

### Example (Natural numbers)

How can define them?

- ▶ basis element 0
- ▶ 1 rule:  $x \mapsto succ(x)$

2 is the natural number defined as  $succ(succ(0))$

### Example (Even numbers)

- ▶ basis element: 0;
- ▶ 1 rule:  $x \mapsto x + 2$ .

### Example (Palindromes on $\{a, b\}$ )

- ▶ basis elements:  $\epsilon, a, b$ ;
- ▶ 2 rules:  $w \mapsto a \cdot w \cdot a, w \mapsto b \cdot w \cdot b$ .

## Outline - Maths Reminders

Proof by induction

Proof by structural induction

A notation: derivation tree

## A notation: derivation tree

Notation for  $t = f(x_1, \dots, x_n)$

$$\frac{x_1 \quad \dots \quad x_n}{t} f$$

"t is built/obtained from  $x_1, \dots, x_n$ " by applying operator  $f$ .

### Example (Derivation trees)

- $2 = \text{succ}(\text{succ}(\text{succ}(0)))$  is a natural number

$$\frac{\frac{\frac{0}{1} \text{ succ}}{2} \text{ succ}}{\text{succ}} \text{ succ}$$

- $aba$  is a palindrome:

$$\frac{\bar{b}}{aba}$$

- $ababa$  is a palindrome:

$$\frac{\frac{\bar{a}}{bab}}{ababa}$$

## Abstract syntax trees and derivation trees

Consider an abstract syntax tree, produced by syntactic analysis.

### Derivation tree

For each node, computing information from the information of its sons.

### Example (Derivation trees in type analysis)

We obtain for each node a type (or error) based on types of its sons.

### Definition of derivation trees by a formal system

Generally, we have information, stored in some environment  $\Gamma$ . The formal system states how to *deduce* knowledge from existing knowledge, where knowledge is of the form  $\Gamma \vdash \mathcal{P}$ , which means  $\mathcal{P}$  holds on  $\Gamma$ .

- a set of axiom schemes
- a set of inference rules : a rule of the form

$$\frac{\Gamma_1 \vdash \mathcal{P}_1 \quad \dots \quad \Gamma_n \vdash \mathcal{P}_n}{\Gamma \vdash \mathcal{C}}$$

"if the hypothesis (premisses)  $\mathcal{P}_i$  hold, then the conclusion  $\mathcal{C}$  holds.