МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего образования
ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт математики, механики и компьютерных наук имени И. И. Воровича

Направление подготовки
02.03.02 — Фундаментальная информатика и информационные технологии

СПЕЦИФИКАЦИЯ СТРУКТУР ДАННЫХ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ COQ

Выпускная квалификационная работа
на степень бакалавра

Студента 4 курса
А. И. Насека

Научный руководитель:
старший преподаватель В. Н. Брагилевский

Допущено к защите:
руководитель направления ФИИТ ________________ В. С. Пилиди

Ростов-на-Дону
2017

Contents

Introduction 3

1. Binary tree 4
   1.1. Inductive definition 4
   1.2. Functions and properties 5

Introduction

1. Binary tree

1.1. Inductive definition

The first basic data structure will be a binary tree. To implement this data structure in Coq, we give an inductive definition $BinaryTree\ T$ for some type T:

```
Inductive BinaryTree (T : Type) : Type :=
    | Nill : BinaryTree T
    | Node : BinaryTree T -> BinaryTree T -> T -> BinaryTree T ->
             BinaryTree T.
```

Type $Nill$ is a bacic constructor, describing the empty tree. Type $Node$ is a tree node constructor, which includes four fields:

- The fisrt argument have a type $BinaryTree\,T$ and it serves to provide additional information (e.g., if we have the tree with parents, this argument can be the reference to the **parent** or if we have the tree with siblings - it can be the reference to the **sibling**).
- The second argument have a type $BinaryTree\,T$ and it's a reference to the **left child**.
- The third argument have a type T. This field is a **value** of a tree node (e.g., as a type T, we can take a standart type in Coq - num and then values of this type will be natural numbers).
- The fourth argument have a type $BinaryTree\ T$ and it's a reference to the **right child**.

We give a several examples of trees (as the type T - we select num):

```
Check (Nill nat).                                      (* Admission *)
Check (Node (Nill nat) (Nill nat) 5 (Nill nat)).       (* Admission *)
Check (Node (Nill nat) (Nill nat) 5
      (Node (Nill nat) (Nill nat) 7 (Nill nat))).      (* Admission *)
```

- In the first example we have the empty tree.
- In the second - binary tree with only one node, in which all references to parent/sibling and childs are empty.
- In the last example we have the tree with two nodes. Root of tree have a value 5 and its right child have a value 7.

1.2. Functions and properties