Mathias Jarl Søndergaard
Mads Esben Hansen
Anton Ruby Larsen

18-10-2018

Technical University of Denmark

Functional programming, Assignment 2

Task 1

1. Here we are dealing with a recursive type declaration. This specific type declaration represents the type commonly known as a "finite trees". As the task suggests, these trees can be built by using three specific rules. This type we are dealing with can consist of 3 parts. The rules can be formulated as:

1. "V of string": If str is any string, then V(str) is of datatype Term
2. "C of int": If n is an integer, then C(n) is of datatype Term.
3. "F of string*Term list": If F contains a tuple with a string and a list containing Terms then F itself will be a Term

As one can see rule 1 & 2 only takes one input, whereas rule 3 takes two and one of them is its own datatype. Therefore, this is a declaration of a recursive datatype.

2. A)

1. Now we are dealing with 5 cases of the type Term. We must justify that these 5 cases are belonging to the type Term, thereby showing that they follow the rules.

2. V "x": Here we are dealing with a pure example of rule 1. Rule 1 states that if we have a string connected to V, then this is a Term. Here we obviously have a string, "x", which is connected to V. So therefore V "x" is a Term.

3. C 3: This value is generated by rule 2. Here we see an integer connected by C, making it the type, Term.

4. F("f0",[]): This value is generated by rule 3. We have a string, "f0" and a Term list, which in this case is empty. The empty list is allowed since it can represent a Term list with no elements present. These two types, string and Term list, follows rule 3, and we can thereby make the type, Term.

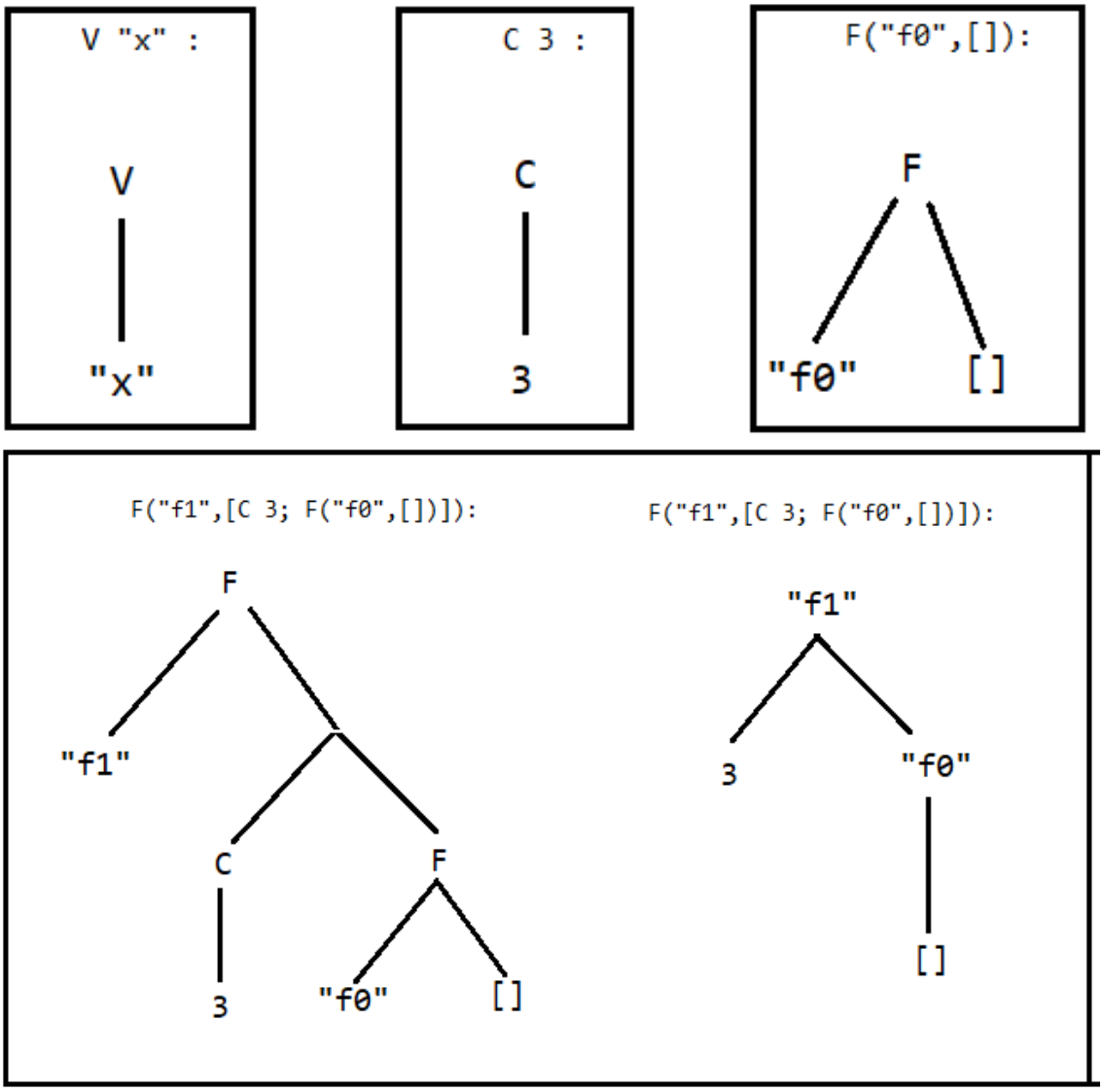
5. F("f1",[C 3; F("f0",[])]): In this case we use rule 3 and rule 2. We start by describing the outermost layer in this tree. This is F("f1",[…]). As one can see we have a string followed by a Term list, connected by F. This is following rule 3 and therefore not violating the rules of being the type Term.

The next layer is the Term list: [C 3; F(“f0”,[])] . This Term list is containing two elements. C 3, which follows rule 2, and F(“f0”,[]), which follows rule 3. We have justified how these two elements are generated and can thereby conclude that they are of type Term implying that they can be elements in Term list.
Therefore the expression F(“f1”,[C 3; F(“f0”,[])]) must be of type Term.

6. F(“max”, [V “x”; C3]): Here we are dealing with a value where all the 3 rules are applied. We have the connector F with a string and a Term list. The two elements in the Term list are generated by rule 1 and rule 2 respectively, enabling us the generate a value like the one presented.

B)

We can represent the illustration of the five trees mentioned with paint:



As one can see there are two versions of the last two values. This is to show how one can create illustrations of these trees in complex and simple form.