Functional Programming - Mandatory 2

Daniel Kuzin
s184225

Mads Kahler
s184206

October 22, 2019

1

1.1

```
type Tree = | A of Tree list
            | B of bool
            | C of Tree * Tree
```

1.2

1. If $t_1 ... t_n$ are trees then $A([t_1 ... t_n])$ is a tree (A)
2. The value $A[]$ is a tree
3. A value of type 'bool' is a tree (B)
4. If t_1, t_2 are trees then $C(t_1, t_2)$ is a tree (C)
5. The type Tree contains no other values than those generated by repeated use of Rules 1, 2, 3, and 4.

1.3

`A[]`

- Use rule 2 to determine that `A[]` is a Tree

`B true`

- Use rule 3 to determine that the value `true` is a Tree.

`C(A [B false; B true], B true)`

- Use rule 3 to determine that `B false` and `B true` are Trees.
- Use rule 1 to determine that `A [B false; B true]` is a Tree.
- Use rule 4 to determine that `C(A [B false; B true], B true)` is a Tree

```
C(A [B true; C(A [B true],B true);B false],
  A [B true; C(A [B true],B true);B false])
```

- Use rule 3 to determine that `B false` and `B true` are Trees.
- Use rule 1 to determine that `A [B true]` is a Tree.
- Use rule 4 to determine that `C(A [B true], B true)` is a Tree
- Use rule 1 to determine that `A [B true; C(A [B true],B true);B false]` is a Tree
- Use rule 4 to determine that the entire expression $C(...)$ is a tree

1.4

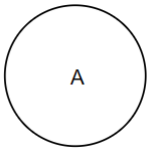

Figure 1: Figure representation of `A []`

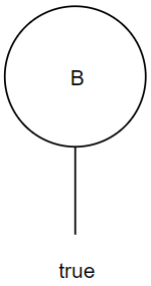

Figure 2: Figure representation of `B true`

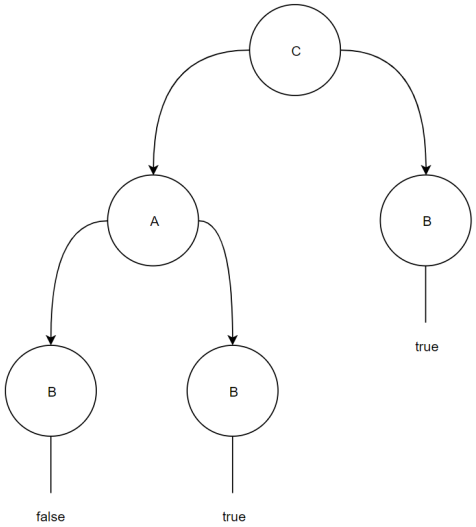

Figure 3: Figure representation of `C(A [B false; B true],B true)`

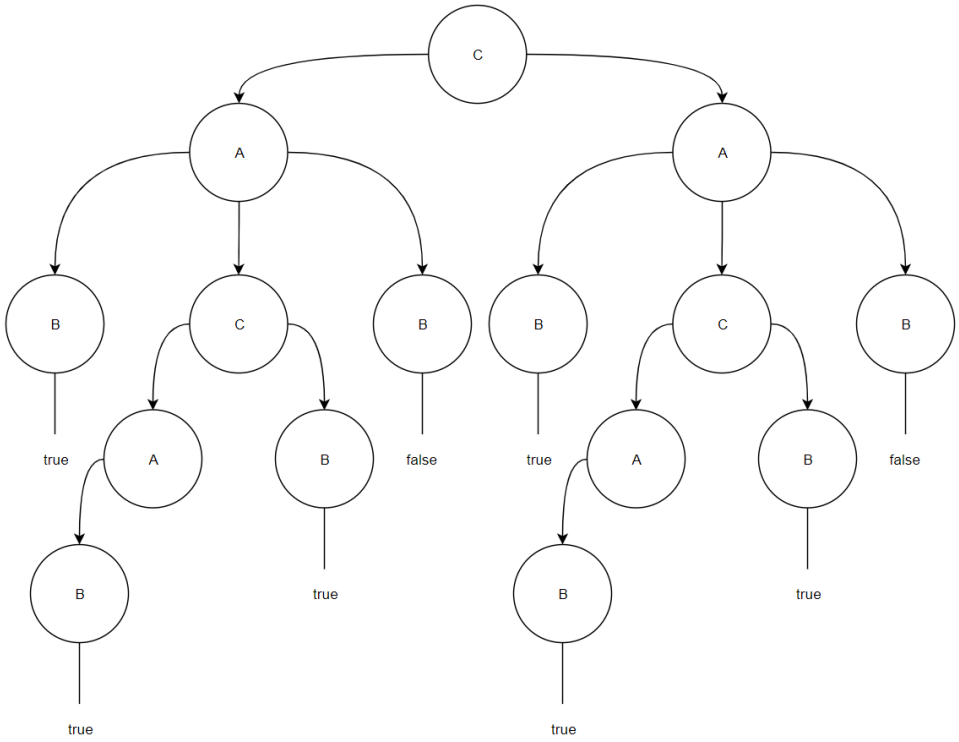

Figure 4: Figure representation of `C(A [B true; C(A [B true],B true);B false], A [B true; C(A [B true],B true);B false])`