

Solutions to In-Class Problems — Week 6, Fri

1 Problems

Problem 1. [carried over from Monday, Oct. 7]

The definition of Recursive Ordered Binary Trees, `RecBinT`, from Week 6 Notes is repeated in an Appendix.

(a) Give a recursive definition of the set of leaves of a `RecBinT`.

Solution. We can then define the set of leaves, L_T , of $T \in \text{RecBinT}$ as follows:

1. ($T = (V, E)$ has one vertex.) $L_T ::= V$.
2. ($T = \text{makeleft}(T_1)$.) $L_T ::= L_{T_1}$.
3. ($T = \text{makeright}(T_1)$.) $L_T ::= L_{T_1}$.
4. ($T = \text{makeboth}(T_1, T_2)$.) $L_T ::= L_{T_1} \cup L_{T_2}$.

■

Full Binary Trees, `FullBinT`, are the special case of `RecBinT`'s in which only the `makeboth` constructor is used. For example, Figure 1 shows an FBT with 13 nodes of which 7 are leaves:

(b) Prove by structural induction on the definition of `FullBinT` that for all `FullBinT`'s, t ,

$$2 |\text{leaves}(t)| = |\text{nodes}(t)| + 1.$$

Solution. Let T, t_1, t_2 be FBTs. Let L, ℓ_1, ℓ_2 represent their *number* of leaves and similarly let N, n_1, n_2 represent their number of nodes. We want to prove that for all FBTs we have: $2L = N + 1$

Base case: When $N = 1$ (the tree has only one vertex), then $L = 1$ (it has one leaf) and so the claim is true ($2 = 1 + 1$).

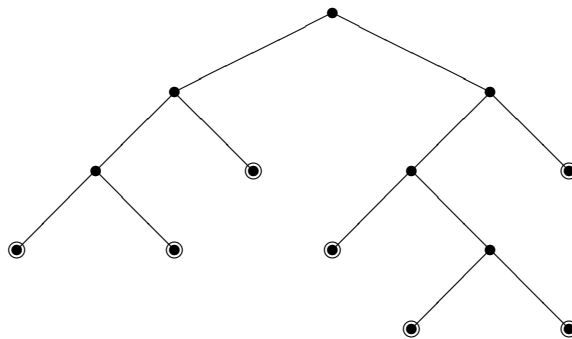


Figure 1: A full binary tree.

Inductive step: Assume that the claim holds for trees t_1 and t_2 . We want to prove that $T = \text{makeboth}(t_1, t_2)$ also satisfies that property. We have:

$$\left. \begin{array}{l} 2 \cdot \ell_1 = n_1 + 1 \\ 2 \cdot \ell_2 = n_2 + 1 \end{array} \right\} \longrightarrow 2 \cdot (\ell_1 + \ell_2) = (n_1 + n_2 + 1) + 1 \longrightarrow 2 \cdot L = N + 1$$

So the claim holds for T as well. ■

Problem 2. [carried over from Monday, Oct. 7]

The Elementary 18.01 Functions (F18's) are the set of functions of one real variable defined recursively as follows:

1. The identity function, $\text{id}(x) ::= x$ is an F18,
2. any the constant function is an F18,
3. \sin, \cos, e^x are F18's,
and if f, g are F18's, then so are
4. $f + g, f - g, fg, f/g$
5. the inverse function f^{-1} ,
6. the composition $f \circ g$.

Show that the Elementary 18.01 Functions are *closed under taking derivatives*. That is, show that if f is an F18, then so is df/dx .

Solution. *Proof.* By Structural Induction on def of F18.

Base Cases: We want to show that the derivatives of all the functions mentioned in rules 1., 2., 3. are in F18. This is easy: for example, $d \operatorname{id}(x)/dx = 1$, and so by rule 2., it is in F18. Similarly, $d \cos(x)/dx = -\sin(x)$ which is also in F18 since $\sin(x) \in \text{F18}$ by rule 3., and therefore $-\sin(x)$ is as well (by rule 4.).

Induction Case: (f^{-1}). Assume $f, df/dx \in \text{F18}$ to prove $df^{-1}(x)/dx \in \text{F18}$.

Letting $y = f(x)$, so $x = f^{-1}(y)$, we know from Leibniz' rule in calculus that

$$df^{-1}(y)/dy = dx/dy = \frac{1}{dy/dx}. \quad (1)$$

For example,

$$d \sin^{-1}(y)/dy = 1/(d \sin(x)/dx) = 1/\cos(x) = 1/\cos(\sin^{-1}(y)).$$

Stated as in (1), this rule is easy to remember, but can easily be misleading because of the variable switching between x and y . It's more clearly stated using variable-free notation:

$$(f^{-1})' = (1/f') \circ f^{-1}. \quad (2)$$

Now, since $f' \in \text{F18}$ (by assumption), so is $1/f'$ (by 4.) and f^{-1} (by 5.), and therefore so is their composition (by 6). Hence the righthand side of equation (2) defines a function in F18.

Induction Case: ($f \circ g$). Assume $f, g, df/dx, dg/dx \in \text{F18}$ to prove $d(f \circ g)(x)/dx \in \text{F18}$.

The Chain Rule states that

$$\frac{d(f(g(x)))}{dx} = \frac{df(g)}{dg} \cdot \frac{dg}{dx}.$$

Stated more clearly in variable-free notation, this is

$$(f \circ g)' = (f' \circ g) \cdot g'.$$

The righthand side of this equation defines a function in F18 by rules 6. and 4.

The other Induction cases are similar.

□

■

Problem 3. We didn't get to this problem on Friday, so we'll do it on the pset.

We can generalize win-lose 2-player terminating games of perfect information to games with "pay-off" amounts. In these games, two players called the *max-player* and the *min-player* alternate moves until the game ends with the min-player paying some payoff amount to the max-player. How much the min-player pays depends on how the game ends. Negative payoffs mean the max-player pays the min-player. The max-player moves first.

Such games are defined by finite-path trees with leaves labelled with real numbers. These are the payoff amounts. The max-player tries to arrive at a leaf with as large a payoff as possible, and the min-player tries to minimize the payoff to the max-player.

Definition. The set of payoff-game trees, PayT , can be defined recursively as follows:

1. If T is a graph with one vertex, v , and no edges, then T is a PayT and $\text{root}(T) ::= v$.
2. if \mathcal{S} is a set of PayT 's such that no vertex occurs in more than one tree in \mathcal{S} , and v is a “new” element that is not a vertex of any tree in \mathcal{S} , then T is in PayT where $\text{root}(T) = v$ and the edges of T are the edges of all the trees in \mathcal{S} along with edges connecting $\text{root}(T)$ to the roots of each of the trees in \mathcal{S} . The trees in \mathcal{S} are called the *children* of T .

We define functions $\text{max-value}(T)$ and $\text{min-value}(T)$ on payoff-game trees, $T \in \text{PayT}$, recursively on the definition of PayT :

1. If T is a single node labelled r , then

$$\text{max-value}(T) = \text{min-value}(T) ::= r.$$

2. If the nonempty set \mathcal{S} is the set of children of T , then

$$\begin{aligned} \text{max-value}(T) & ::= \text{lub} \{ \text{min-value}(S) \mid S \in \mathcal{S} \} \\ \text{min-value}(T) & ::= \text{glb} \{ \text{max-value}(S) \mid S \in \mathcal{S} \}. \end{aligned}$$

(a) Suppose a payoff-game tree, T , is *finite*. Prove that

1. If the max-player is the first player to move in T , then he has a strategy that guarantees his payoff will be at least $\text{max-value}(T)$, no matter how the min-player behaves.
2. If the max-player is the second player to move in T , then he has a strategy that guarantees his payoff will be at least $\text{min-value}(T)$.
3. Likewise, if the min-player is the first player to move in T , then she has a strategy that guarantees the payoff to max-player will be at most $\text{min-value}(T)$.
4. If the min-player is the second player to move in T , then she has a strategy that guarantees the payoff to max-player will be at most $\text{max-value}(T)$.

(So the players may as well skip playing and just have the min-player pay $\text{max-value}(T)$ to the max-player.)

Solution. To appear in Problem Set 6-7 solutions. ■

(b) Now generalize the previous part to arbitrary PayT 's. *Hint:* It might be helpful to assume the payoff amounts at the leaves are bounded above and below by particular numbers. After settling this case, try it without assuming bounds. Note that in the unbounded case, $\text{max-value } T$ may be $+\infty$ and $\text{min-value } T$ may be $-\infty$.