Self Reference: Recursive Types

CSCI/MATH 243
Wheaton College
Thomas VanDrunen
Spring 2020

Reminder: Project prototype due Apr 13

Chapter 6 in context:

- Chapter 5 Relations: Builds on proofs about sets
- Chapter 6 Self Reference: Interlude between Chapters 5 and 7, focuses on recursive thinking
- Chapter 7 Function: Builds on proofs about relations

Chapter 6 outline:

- Recursive definitions, recursive types (Apr 1)
- Recursive proofs I: Structural induction (Apr 3)
- Recursive proofs II: Mathematical induction (Apr 6)
- Math induction applied: Loop invariants (Apr 8 & 13)

Axiom 7
There exists a whole number 0.

Axiom 8
Every whole number n has a successor, `succ` *n.*

Axiom 9
No whole number has 0 as its successor.

Axiom 10
If $a, b \in \mathbb{W}$*, then* $a = b$ *iff* `succ` $a =$ `succ` b*.*

A *whole number* is either zero or one more than another whole number.

Compare to:
A *list* is either empty or an element together with its following list.

5 is a whole number because

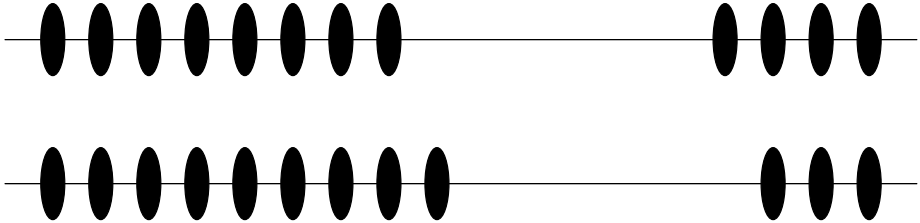
5 is a whole number because it is the successor of
4, which is a whole number because

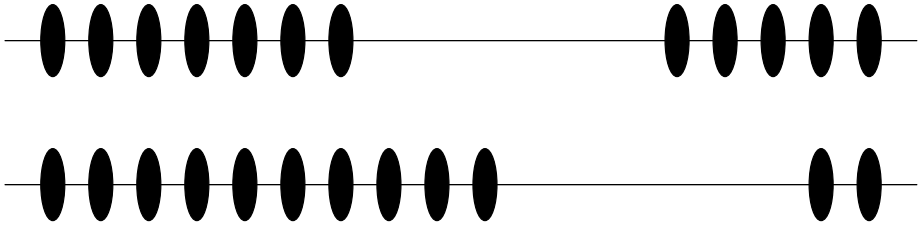
5 is a whole number because it is the successor of
4, which is a whole number because it is the successor of
3, which is a whole number because

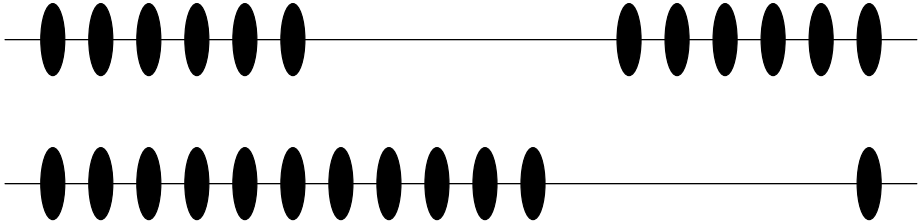
5 is a whole number because it is the successor of
4, which is a whole number because it is the successor of
3, which is a whole number because it is the successor of
2, which is a whole number because

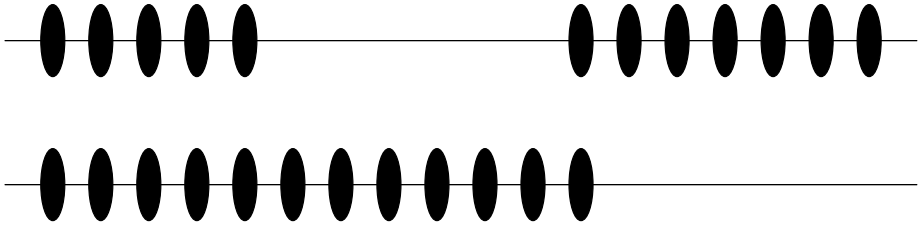
5 is a whole number because it is the successor of
4, which is a whole number because it is the successor of
3, which is a whole number because it is the successor of
2, which is a whole number because it is the successor of
1, which is a whole number because

5 is a whole number because it is the successor of
4, which is a whole number because it is the successor of
3, which is a whole number because it is the successor of
2, which is a whole number because it is the successor of
1, which is a whole number because it is the successor of
0, which is a whole number by Axiom 7.









Lemmas for addition:

- $0 + b = b$
- $a + 0 = a$
- $a + b = (a + 1) + (b - 1)$

Lemmas for subtraction:

- $a - 0 = a$
- $a - b = (a - 1) - (b - 1)$

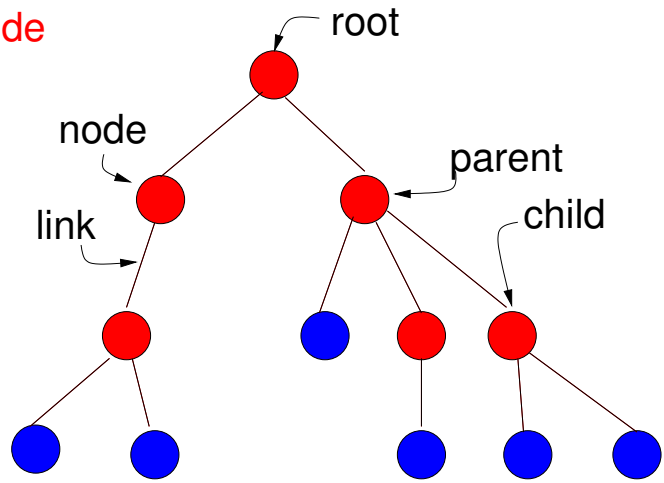
Lemmas for multiplication:

- $a \cdot 0 = 0$
- $0 \cdot b = 0$
- $a \cdot 1 = a$
- $a \cdot b = a + (a \cdot (b - 1))$

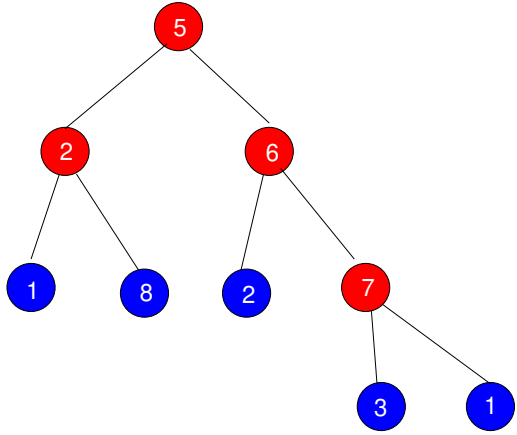
Tree

internal node

leaf



Full Binary Tree



Expression trees:

```
datatype operation = Plus | Minus | Mul | Div;
datatype expression =  Internal of operation * expression * expression
      | Leaf of int;
```

$((5-7)*((3+2)/8))$

```
val exprExample = Internal(Mul, Internal(Minus,Leaf(5), Leaf(7)),
                                Internal(Div,
                                         Internal(Plus, Leaf(3),
                                                        Leaf(2)),
                                         Leaf(8)));
```