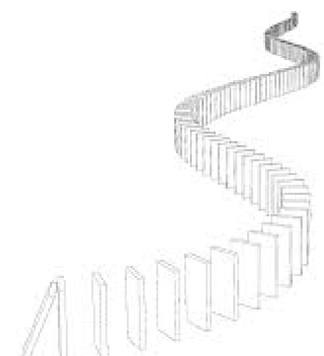
COMP 1002

Logic for Computer Scientists

Lecture 24



Admin stuff

- Assignment 3 extension
  - Because of the power outage, assignment 3 now due on Tuesday, March 14 (also 7pm)

- Assignment 4 is posted.
  - Due March 23rd.

Puzzle

- Do the following two English sentences have the same parse trees?
  - Time flies like an arrow.

  - Fruit flies like an apple.

Structural induction

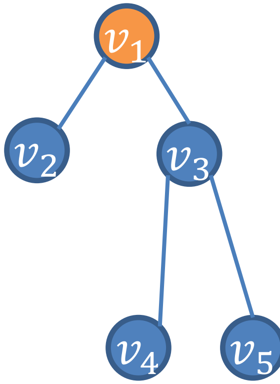
- Let $S \subseteq U$ be a recursively defined set, and F(x) is a property (of $x \in U$).
- Then
  - if all x in the base of S have the property,
  - and applying the recursion rules preserves the property,
  - then all elements in S have the property.

Multiples of 3

- Let's define a set S of numbers as follows.
  - Base: $3 \in S$
  - Recursion: if $x, y \in S$, then $x + y \in S$
- Claim: all numbers in S are divisible by 3
  - That is, $\forall x \in S\ \exists\ z \in \mathbb{N}\ x = 3z$.
- Proof (by structural induction).
  - Base case: 3 is divisible by 3 (y=1).
  - Recursion: Let $x, y \in S$. Then $\exists z, u \in \mathbb{N}\ x = 3z \wedge y = 3u$.
  - Then $x + y = 3z + 3u = 3(z + u)$.
  - Therefore, $x + y$ is divisible by 3.
  - As there are no other elements in S except for those constructed from 3 by the recursion rule, all elements in S are divisible by 3.

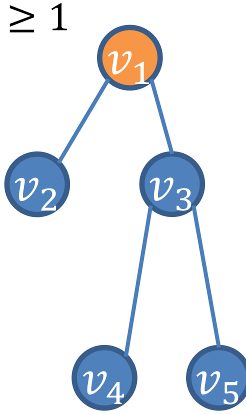
Binary trees

- **Rooted trees** are trees with a special vertex designated as a root.
  - Rooted trees are **binary** if every vertex has at most three edges: one going towards the root, and two going away from the root. Full if every vertex has either 2 or 0 edges going away from the root.
- Recursive definition of full binary trees:
  - Base: A single vertex v is a full binary tree with that vertex as a root.
  - Recursion:
    - Let T_1, T_2 be full binary trees with roots r_1, r_2, respectively. Let v be a new vertex.
    - A new full binary tree with root v is formed by connecting r_1 and r_2 to v.
  - Restriction:
    - Anything that cannot be constructed with this rule from this base is not a full binary tree.



Height of a full binary tree

- The **height** of a rooted tree, $h(T)$, is the maximum number of edges to get from any vertex to the root.
  - Height of a tree with a single vertex is 0.
- Claim: Let $n(T)$ be the number of vertices in a full binary tree T. Then $n(T) \leq 2^{h(T)+1} - 1$
- Proof (by structural induction)
  - Base case: a tree with a single vertex has $n(T) = 1$ and $h(T) = 0$. So $2^{h(T)+1} - 1 = 1 \geq 1$
  - Recursion: Suppose T was built by attaching T_1, T_2 to a new root vertex v.
    - Number of vertices in T is $\mathrm{n(T)} = n(T_1) + n(T_2) + 1$
    - Every vertex in T_1 or T_2 now has one extra step to get to the new root in T. So $h(T) = 1 + \max(h(T_1), h(T_2))$
    - By the induction hypothesis, $n(T_1) \leq 2^{h(T_1)+1} - 1$ and $n(T_2) \leq 2^{h(T_2)+1} - 1$
    - $$\begin{aligned} \mathrm{n(T)} &= n(T_1) + n(T_2) + 1 \\ &\leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1\ ) \\ &\leq 2 \cdot \max(2^{h(T_1)+1}, 2^{h(T_2)+1})\ \ - 1 \\ &\leq 2 \cdot 2^{\max(h(T_1), h(T_2))+1} - 1 \\ &= 2 \cdot 2^{h(T)} - 1 = 2^{h(T)+1} - 1 \end{aligned}$$
  - Therefore, the number of vertices of any binary tree T is less than $2^{h(T)+1} - 1$
- Alternatively, height of a binary tree is at least $\log_2 n(T)$
  - If you have a recursive program that calls itself twice (e.g, within if … then … else …)
  - Then if this code executes n times (maybe on n different cases)
  - Then the program runs in time at least $\log_2 n$, even when cases are checked in parallel.



Height 2