Inductive Definitions

- What are Inductive Definitions?
- Extremal Clause
- Proofs by Induction
- Defining Sets by Rules in Java
We will frequently define a set by a collection of rules that determine the elements of that set.

Example: the set of valid sentences of a grammar

What does it mean to define a set by a collection of rules?
Examples

- Numerals, in unary (base-1) notation.
  - *Zero* is a numeral;
  - if *n* is a numeral, then so is *Succ*(n).

- Binary trees (w/o data at nodes):
  - *Empty* is a binary tree;
  - If *l* and *r* are binary trees, then so is *Node*(l, r).
Examples (more formally)

- Numerals: The set $\text{Num}$ is defined by the rules
  
  $\begin{align*}
  \text{Zero} & : n \\
  \text{Succ} & : n \\
  \end{align*}$

- Binary trees: The set $\text{Tree}$ is defined by the rules
  
  $\begin{align*}
  \text{Empty} & : t_l, t_r \\
  \text{Node} & : t_l, t_r \\
  \end{align*}$
Defining a Set by Rules

- Given a collection of rules, what set does it define?
  - What is the set of numerals?
  - What is the set of trees?
- Do the rules pick out a unique set?
Defining a Set by Rules

- There can be many sets that satisfy a given collection of rules.
  - \( \text{MyNum} = \{ \text{Zero}, \text{Succ(Zero)}, \ldots \} \)
  - \( \text{YourNum} = \text{MyNum} \cup \{ \infty, \text{Succ(\infty)}, \ldots \} \), where \( \infty \) is an arbitrary symbol
- Both \( \text{MyNum} \) and \( \text{YourNum} \) satisfy the rules defining numerals (i.e., the rules are true for these sets). Really?
MyNum Satisfies the Rules

\[ n \]

**Zero**  \[ Succ(n) \]

\[ MyNum = \{ \text{Zero}, Succ(\text{Zero}), Succ(Succ(\text{Zero})), \ldots \} \]

Does MyNum satisfy the rules?

- Zero \( \in \) MyNum.  \( \checkmark \)
- If \( n \in \) MyNum, then \( Succ(n) \in \) MyNum.  \( \checkmark \)
**YourNum Satisfies the Rules**

YourNum =
\{Zero, Succ(Zero), Succ(Succ(Zero)), \ldots\} \cup \{\infty, Succ(\infty), \ldots\}

Does YourNum satisfy the rules?

- Zero ∈ YourNum. √
- If n ∈ YourNum, then Succ(n) ∈ YourNum. √
Defining Sets by Rules

- Both *MyNum* and *YourNum* satisfy all rules.
- It is not enough that a set satisfies all rules.
- Something more is needed: an *extremal* clause.
  - “and nothing else” or
  - “the least set that satisfies these rules”
Example 1: *Num*

*Num* is the least set that satisfies these rules:

- *Zero* is included
- If *n* is included, then *Succ*(n) is included.
Example 2: *Tree*

*Tree* is the least set that satisfies these rules:

- *Zero* is included
- If $t_l$ and $t_r$ are included, then $Node(t_l, t_r)$ is included.
Inductive Definitions

Question: What do we mean by “least”?
Answer: The smallest with respect to the subset ordering on sets.

- Contains no “junk”, only what is required by the rules.
- Since \( \text{YourNum} \supseteq \text{MyNum} \), \( \text{YourNum} \) is ruled out by the extremal clause.
- \( \text{MyNum} \) is “ruled in” because it has no “junk”.

Inductive Definitions
What’s the Big Deal?

- Inductively defined sets “come with” an induction principle.
- Suppose $I$ is inductively defined by rules $R$.
- To show that every $x \in I$ has property $P$, it is enough to show that $P$ satisfies the rules of $R$.
- Sometimes called *structural induction* or *rule induction*. 
Induction Principle

- To show that every $n \in \text{Num}$ has property $P$, it is enough to show:
  - $\text{Zero}$ has property $P$.
  - if $n$ has property $P$, then $\text{Succ}(n)$ has property $P$.

- This is just ordinary mathematical induction!
Induction Principle

- To show that every tree has property $P$, it is enough to show that
  - *Empty* has property $P$.
  - If $l$ and $r$ have property $P$, then so does $Node(l, r)$.
- We call this *structural induction on trees*. 
How can we justify this principle?

- Properties are sets. We are trying to show that $P \supseteq I$.
- Remember that $I$ is (by definition) the smallest set satisfying the rules in $R$.
- Hence if $P$ satisfies the rules of $R$, then $P \supseteq I$.
- This is why the extremal clause matters so much!
Example: Size of a Tree

To show: Every tree has a size, defined as follows:

Definition of size

- The size of Empty is 1.
- If tree $l$ has size $h_l$ and tree $r$ has size $h_r$, then the tree $Node(l, r)$ has size $1 + h_l + h_r$.

Clearly, every tree has at most one size, but does it have a size at all?
Example: size

It may seem obvious that every tree has a size, but notice that the justification relies on structural induction!

- An “infinite tree” does not have a size!
- But the extremal clause rules out the infinite tree!
Example: size

- We prove that every tree (as defined above) has a size (as defined above).
- Proceed by induction on the rules defining trees, showing that the property “has a size” satisfies the rules defining trees.
- Since the set of trees is the least set that satisfies the rules, the property “has a size” must be a superset of the set of trees!
Example: size

Definition of size

- The size of $Empty$ is 1.
- If tree $l$ has size $h_l$ and tree $r$ has size $h_r$, then the tree $Node(l, r)$ has size $1 + h_l + h_r$.

Rule 1 of Def of Tree: $Empty$ is included.
Do all things that have a size fulfill this rule?
Does $Empty$ have a size? **yes**

Rule 2 of Def of Tree: If $l$ and $r$ are included, then $Node(l, r)$ is included.
Does all things that have a size fulfill this rule?
If $l$ and $r$ have sizes, then $Node(l, r)$ has a size? **yes**
Example: size (summary)

- We have defined $Tree$ as the least set satisfying:
  - $Zero$ is included
  - If $t_l$ and $t_r$ are included, then $Node(t_l, t_r)$ is included.
- We have shown that the property “has a size” is a set satisfying
  - $Zero$ is included
  - If $t_l$ and $t_r$ are included, then $Node(t_l, t_r)$ is included.
- Thus, the property “has a size” is a superset of $Tree$, meaning: Every $Tree$ has a size.
Encoding Numerals in Java

```java
interface Num {}
class Zero implements Num {}
class Succ implements Num {
    public Num pred;
    Succ(Num p) {pred = p;}
}
Num my_num = new Zero();
Num my_other_num =
    new Succ(new Succ(new Zero()));
```
Encoding Trees in Java

```java
interface Tree {}
class Empty implements Tree {}
class Node implements Tree {
    public Tree left, right;
    Node(Tree l, Tree r) {
        left = l; right = r;
    }
}
Tree my_tree =
    new Node(new Empty(),
        new Node(new Node(new Empty(),
            new Empty()),
            new Empty()));
```
Constructors and Rules

- The constructors of the classes correspond to the rules in the inductive definition.

- Numerals
  - `new Zero()` is of type `Num`
  - `if n is of type Num, then new Succ(n) is of type Num`

- Trees
  - `new Empty()` is of type `Tree`
  - `if l and r are of type Tree, then new Node(l, r) is of type Tree`
Analogy with Java

- We assume an implicit extremal clause: no other classes implement the interface.
- The associated induction principle may be used to prove termination and correctness of functions.
Example: Size in Java

```java
interface Tree {
    public int size();
}
class Empty implements Tree {
    public int size() {return 1;}
}
class Node implements Tree {
    public Tree left, right;
    Node(Tree l, Tree r) {left = l; right = r;}
    public int size() {
        return 1 + left.size() + right.size();
    }
}
```
Proving Termination of Java Program

Why does $\text{size}(t)$ terminate for every tree $t$?

- For every $t$ of type $\text{Tree}$, does there exist $h$ such that $\text{size}(t)$ returns $h$?
- Proof similar to above!
An inductively defined set is the least set closed under a collection of rules.

Rules have the form:

“If $x_1 \in X$ and \ldots and $x_n \in X$, then $x \in X$.”

\[ x_1 \quad \ldots \quad x_n \]

Notation:

\[ \begin{array}{c}
  x_1 \\
  \vdots \\
  x_n \\
\end{array} \]

\[ \underline{X} \]
Inductive Definitions

What are Inductive Definitions?
Extremal Clause
Proofs by Induction
Defining Sets by Rules in Java

Summary

- Inductively defined sets admit proofs by rule induction.
- For each rule
  \[ x_1 \cdots x_n \]
  \[ \overline{\hspace{\textwidth}} \]
  \[ x \]
  assume that \( x_1 \in P, \ldots, x_n \in P \), and show that \( x \in P \).
- Conclude that every element of the set is in \( P \).