Inductive Definitions

Inductive Definitions

- We will frequently define a set by a collection of **rules** that determine the elements of that set.
  - Eg, the "sentential forms" of a grammar
- What does it mean to define a set by a collection of rules?
- If we can define sets, we can define relations. Remember: relations are sets of ordered pairs (or n-tuples).

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

- **Sentential forms** of a grammar G
  - S is a s.f. (where S is the start symbol of G);
  - if $\alpha A \beta$ is a s.f., and A -> γ is a production, then $\alpha\gamma\beta$ is a s.f.
- The **language** L(G) of G is the set of s.f.'s with no non-terminals (the **sentences**).

Analogy with SML

- Numerals in unary

```
datatype num = Zero | Succ of num
```

- Trees

```
datatype tree =
  Empty | Node of tree * tree
```


Analogy with SML

- The **constructors** of the datatype declaration determine the rules.
- Numerals
  - `Zero : num`
  - `if n : num, then Succ(n) : num`
- Trees
  - `Empty : tree`
  - `if l, r : tree, then Node (l, r) : tree`

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?
- This is not a trivial question!
  - Do the rules pick out a unique set?

Defining a Set by Rules

- There can be many sets **closed under** a given collection of rules.
  - Eg, N = { Zero, Succ(Zero), … }
  - Eg, N* = N ∪ { ∞ }, where Succ(∞) = ∞.
- **Both** N and N* are closed under the rules defining numerals (i.e., the rules are "true" of these sets).

Defining a Set by Rules

- Closure of N under the rules:
  - Zero in N;
  - if n in N, Succ(n) in N.
- Closure of N* under the rules:
  - Zero is in N*, since Zero is in N;
  - If n in N*, then either n in N or n = ∞. If n in N, then Succ(n) in N, so Succ(n) in N*. Since Succ(∞)= ∞, this is also in N*.

Defining a Set by Rules

- A possible worry: does the "infinite" numeral ∞ really exist?
  - That is, does there exist a **fixed point** for the Succ function, a "numeral" n such that Succ(n) = n?
  - Is there more than one? (We only care if there exists at least one.)

Infinity in SML/NJ

- Here's how to define ∞ in SML/NJ, using a "lazy" datatype:

```
datatype lazy nat_star =
  Zero | Succ of nat_star
val rec lazy infty = Succ infty
```

- Key idea: using a **recursion equation** to define `infty`.
  - "`val rec`" solves equations!

Lazy datatypes are not "Standard" ML; enable them by `Compiler.Control.lazysml := true;`

Defining Sets by Rules

- Closure under a given set of rules is not enough to determine a **unique** set.
  - Both N and N* are closed under the rules.
- Something more is needed, an **extremal clause**.
  - "and nothing else"
  - "the least set closed under these rules"

Inductive Definitions

- An **inductively-defined** set is the **least** set closed under a given set of rules.
- Example: N is the **least set** closed under these rules:
  - Zero in N;
  - if n in N, then Succ(n) in N;
  - (nothing else is in N)

Inductive Definitions

- What do we mean by "least"?
  - Smallest, with respect to the subset ordering on sets.
  - Contains no "junk", only what is required by the rules.
- Since $N \subset N^*$, N^* is **ruled out** by the extremal clause.
  - N is "ruled in" because it has no "junk".

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 is **closed under** the rules of R.
- Sometimes called **structural induction** or **rule induction**.

Induction Principles

- How can we justify this principle?
  - Properties are sets. We are trying to show that $I \subseteq P$.
  - Remember that I is (by definition) the **smallest** set closed under R.
  - Hence if P is closed under R, then $I \subseteq P$.
- This is why the extremal clause matters so much!

Induction Principles

- To show that every n in N has property P, it is enough to show:
  - Zero has property P;
  - if n has property P, then Succ(n) has property P.
- This is just ordinary "mathematical induction"!

Induction Principles

- 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);
- This is "structural induction" on trees.

Example: height of a tree

- To show: every tree has a **height** h:
  - the height of Empty is 0;
  - if l has height hl and r has height hr, then Node(l,r) has height 1+max(hl,hr).
- Clearly, every tree has **at most one** height, but does it have a height at all?

Example: height

- It may seem obvious that every tree has a height, but notice that the justification relies on structural induction!
  - The "infinite tree" does **not** have a height!
  - But the extremal clause rules out the infinite tree!

Example: height

- Formally, we prove that for every tree t, there exists a number h satisfying the specification of height.
- Proceed by induction on the rules defining trees, showing that the property "there exists a height h for t" is closed under these rules.

Example: height

- Rule 1: Empty is a tree.
  - Does there exist h st h is the height of Empty? Yes! Take h=0.
- Rule 2: Node(l, r) is a tree if l and r are trees.
  - Suppose that there exists hl and hr, the heights of l and r, resp. Does there exist h st h is the height of Node(l,r)? Yes! Take h = 1 + max(hl, hr).

Analogy with ML

- An ML datatype declaration has an **implicit** extremal clause: no other constructors.
  - Hence the "exhaustiveness" check on patterns is possible!
- The associated induction principle may be used to prove **termination** and **correctness** of functions.

Analogy with ML

- Example: height in ML.

```
fun height Empty = 0
  | height (Node (l, r)) =
    1 + max (height l, height r)
```

- Why does `height` terminate?
  - For every `t:tree`, does there exist h such that `height t` evaluates to h?
  - Same form as previous argument!

Analogy with ML

- Rule 1: `Empty`.
  - Clearly, `height Empty` evaluates to `0`.
- Rule 2: `Node(l, r)`.
  - Assume that `height l` terminates with `hl`, and `height r` terminates with `hr`.
  - Clearly, `height (Node (l, r))` terminates with `1 + max (hl, hr)`.

"Stages" of Definition

- We may think of an inductive set as being built up in "stages".
  - Stage 0: empty.
  - Stage n+1: all the elements obtained by applying a rule to elements at stage n.
- eg, $T_0 = \{\}$, $T_{n+1} = \{ \text{Node}(l,r) \mid l, r \text{ in } T_n \}$
  $T = \cup\{ T_n \mid n >= 0 \}$

Summary

- An inductively-defined set is the least set closed under a collection of rules.
  - Rules have the form if x1 in X and … and xn in X, then x=x(x1,…,xn) in X.
  - Notation: x1 … xn -> x, or x1 … xn / x, usually written vertically.

Summary

- Inductively-defined sets admit proofs by **rule induction**.
  - For each rule x1 … xn / x, assume that x1 in P, …, xn in P, and show that x in P.
  - Conclude that every element of the set is in P.

Thought Questions

- Is there a "least set" closed under these rules?
  - Zero in X;
  - if n in X, then Succ(Succ(n)) in X;
  - if n not in X, then Succ(n) in X.
- Or these:
  - Zero is in X;
  - if n not in X, then n in X.

Coinductive Definitions

- N* can be defined **coinductively** as the **largest** set **consistent with** the rules.
  - **largest** wrt set containment, within a universe built up using Zero and Succ.
  - X is **consistent with** a rule x1 … xn / x iff whenever x in X, so are x1, …, xn.

Coinductive Definitions

- Why is N* consistent with the rules?
  - Since Zero in N*, all of the premises of rules introducing Zero must be in N*. This is true vacuously.
  - Suppose Succ(n) in N*. We must show that n in N*. Two cases: n in N, in which case n in N*, or n=∞=Succ(∞) in N* (!).

Coinduction

- Suppose that C is coinductively defined by a set of rules R.
- To show that P⊆C (note reversal!), it suffices to show that P is consistent with R.
  - eg, if Succ(n) in P, then n in P.
- Observe that this covers infinite elements!

Coinduction

- Example: to show that N subset N*, it suffices to show that N is consistent with the rules defining N*.
  - That is, if Succ(n) in N, then n in N. This is clearly true, by definition of N.
- We'll see more interesting examples much later in the course (time permitting).