Section 4.3
Recursive Definitions and Structural Induction

Recursive or *inductive* definitions of sets and functions on recursively defined sets are similar.

1. *Basis step:*

For sets-

• State the basic building blocks (BBB's) of the set.

or

For functions-

• State the values of the function on the BBB’s.

2. *Inductive or recursive step:*

For sets-

• Show how to build new things from old with some construction rules.

or

For functions-

• Show how to compute the value of a function on the new things that can be built knowing the value on the old things.

3. *Extremal clause:*

For sets-

• If you can't build it with a finite number of applications of steps 1. and 2. then it isn't in the set.

For functions-

• A function defined on a recursively defined set does not require an extremal clause.

Note: Your author doesn't mention the extremal clause.

It is a standard part of an inductive definition of a set but often ignored (“since everybody knows it is supposed to be there”).

Also note:

• To prove something is in the set you must show how to construct it with a finite number of applications of the basis and inductive steps.

• To prove something is not in the set is often more difficult.

Example:

A recursive definition of N:

1. *Basis:*

   0 is in N (0 is the BBB).

2. *Induction:*

   if n is in N then so is n + 1 (how to build new objects from old: “add one to an old object to get a new one”).

3. *Extremal clause:*

   If you can't construct it with a finite number of applications of 1. and 2., it isn't in N.

Now given the above recursive definition of N we can give recursive definitions of functions on N:

1. $f(0) = 1$ (the *initial condition* or the value of the function on the BBB’s).

2. $f(n + 1) = (n + 1) f(n)$ (the *recurrence* equation, how to define f on the new objects based on its value on old objects)

f is the *factorial function*: $f(n) = n!$.

Note how it follows the recursive definition of N.

Proof of assertions about inductively defined objects usually involves a

Proof by induction.

• Prove the assertion is true for the BBBs in the basis step.

• Prove that if the assertion is true for the old objects it must be true for the new objects you can build from the old objects.

• Conclude the assertion must be true for all objects.

Example:

We define a^n inductively where n is in N.

• Basis: $a^0 = 1$

• Induction: $a^{(n+1)} = a^n a$

Theorem: $m \quad n[a^m a^n = a^{m+n}]$

Proof:

Since the powers of *a* have been defined inductively we must use a proof by induction somewhere.

Get rid of the first quantifier on *m* by Universal Instantiation:

• Assume *m* is arbitrary.

Now prove the remaining quantified assertion

$$n[a^m a^n = a^{m+n}]$$

by induction:

1. *Basis step:* Show it holds for n = 0.

The left side becomes $a^m a^0 = a^m(1) = a^m$

The right side becomes $a^{m+0} = a^m$

Hence, the two sides are equal to the same value.

2. *Induction step:* <u>The Induction hypothesis:</u>

Assume the assertion is true for n: $a^m a^n = a^{m+n}$.

Now show it is true for n + 1.

The left side becomes

$$a^m a^{n+1} = a^m(a^n a) = (a^m a^n)a = a^{m+n}a$$

which follows from

- the inductive step in the definition of a^n and
- the induction hypothesis and
- the associativity of multiplication.

The right side becomes

$$a^{m+(n+1)} = a^{(m+n)+1} = a^{m+n}a$$

which follows from

- the inductive definition of the powers of a
- the associativity of addition.

Hence, we have shown for arbitrary *m* that

$$n[a^m a^n = a^{m+n}]$$

is true by induction.

Since *m* was arbitrary, by Universal Generalization,

$$m \quad n[a^m a^n = a^{m+n}].$$

Q. E. D.

Example: A recursive definition of the *Fibonacci sequence*

1. *Basis:*

f(0) = f(1) = 1

(two initial conditions)

2. *Induction:*

f(n + 1) = f (n) + f(n - 1)

(the recurrence equation).

Example:

A recursive definition of the set of strings over a finite alphabet Σ.

The set of all strings (including the empty or null string λ) is called (the monoid) Σ^*.

(Excluding the empty string it is called Σ^+.)

1. *Basis:*

   The empty string λ is in Σ^*.

2. *Induction:*

   If w is in Σ^* and a is a symbol in Σ, then wa is in Σ^*.

Note: we can concatenate a on the right or left, but it makes a difference in proofs since concatenation is not commutative!

3. *Extremal clause.*

Note: infinitely long strings cannot be in Σ^*. (why?)

Example:

Let $\sum = \{a, b\}$. Then aab is in $\sum^*$.

Proof:

We construct it with a finite number of applications of the basis and inductive steps in the definition of $\sum^*$:

1. λ is in $\sum^*$ by the basis step.

2. By step 1., the induction clause in the definition of $\sum^*$ and the fact that a is in $\sum$, we can conclude that $\lambda a = a$ is in $\sum^*$.

3. Since a is in $\sum^*$ from step 2., and a is a symbol in $\sum$, applying the induction clause again we conclude that aa is in $\sum^*$.

4. Since aa is in $\sum^*$ from step 3 and b is in $\sum$, applying the induction clause again we conclude that aab is in $\sum^*$.

Since we have shown aab is in $\sum^*$ with a finite number of applications of the basis and induction clauses in the definition we have finished the proof.

Q.E.D.

Example:

We give an inductive definition of the well formed parenthesis strings P:

1. *Basis clause:*

( ) is in P

2. *Induction clause:*

if w is in P then so are

( ) w, (w), and w( )

3. *Extremal clause*

Example:

(( )( )) is in P.

Proof:

1. ( ) is in P by the basis clause

2. ( )( ) must be in P by step 1. and the induction clause

3. (( ) ( )) must be in P by step 2. and the induction clause.

Q. E. D.

Note:))(( ) is not in P. Why? Can you prove it?

(Hint: what can you say about the length of strings in P? How can you order the strings in P?)

One More Example:

The set S of bit strings with no more than a single 1.

Basis:

λ , 0 , 1 are in S

Induction:

if w is in S, then so are 0w and w0

Extremal Clause
