Abstract data type: Numbers

Abstract data type

- Set of possible values of a variable of that type.
- Operations that can be performed on the values.
- Assumptions about the operations and values.
- Nothing to do with a computer or application.
- Essentially a mathematical object.

Numbers

- The most basic type, values are 0, 1, 2,….
- How to define numbers formally?
- What operations to assume?
- Assume as little as possible, and derive as much as possible from the assumptions.

Peano Axioms

- One possible way of defining numbers.
- There can be alternative definitions.
- These are one of the simplest.
- The whole subject of Number Theory can be built using just these.

Peano Axioms

1) There exists a number called *zero* denoted *0*.
2) There is an operation called *next* such that for any number *n*, *next(n)* is also a number.
3) If *S* is a set of values such that
    a) *0* belongs to *S*.
    b) If *n* belongs to *S* then so does *next(n)*, for all *n*.

   then *S* contains all natural numbers.

Recursive definition

- Base case: assume an initial value (*0*).
- An operation to generate new values (*next*)
  - For numbers, this is just adding one.
- There are no other possible values.
- Many data types follow the same pattern.
- Implicit assumption that the operation generates \`new' values and *next(n) != 0* for all *n*.
- *n = m* if and only if *next(n) = next(m)*.

Induction

- The third assumption is essentially induction.
- To prove a property *P*(*n*) holds for all numbers *n*.
  - Show that *P*(*0*) is true.
  - Assuming *P*(*n*) is true for some *n*, show that *P*(*next*(*n*)) is true.
- Let *S* be the set of values of *n* for which *P*(*n*) is true.
  - *0* belongs to *S* by the base case.
  - If *n* belongs to *S* then so does *next(n)*, by induction step.
- *S* contains all numbers by the third axiom.

Defining operations

- All other operations defined using induction.
- Addition:
  - $add(n,0) := n$ for all numbers n.
  - $add(n,next(m)) := next(add(n,m))$ for all n, m.
- Defines addition for all numbers n and m.
- Prove that $add(n,m) = add(m,n)$ for all n, m.
  - Not obvious from the definition.

More operations

- Use already defined operations to define others.
- Multiplication.
  - $mult(n,0) := 0$ for all n.
  - $mult(n,next(m)) := add(mult(n,m),n)$ for all n,m.
- Again, not obvious that $mult(n,m) = mult(m,n)$.
- Prove: $mult(x,add(y,z)) = add(mult(x,y),mult(x,z))$.

Ordering

- Define a relation <= between numbers.
- *0* <= *m* := *true* for all *m*.
- *next(n)* <= *0* := *false* for all *n*.
- *next(n)* <= *next(m)* := *n* <= *m for all n, m.*
- Double induction.
  - First *0* <= *m* is defined for all *m*.
  - Then *next(n)* <= *m* is defined, by induction on *m*.
  - By induction on *n*, *n* <= *m* is defined for all *n, m*.

Ordering properties

- Prove these properties of the <= relation.
- For all x, x <= x.
- If x <= y and y <= z then x <= z.
- If x <= y and y <= x then $x = y$.
- For all x, y either x <= y or y <= x.
- If x <= *next(y)* then either x <= y or x = *next(y)* for all x,y.
- Such a relation is called a **total order**.

Minimum Example

- Another way of stating the induction axiom.
- More convenient to use.
- If a set S contains at least one number n, then it contains a number m with the property that

  $m <= x$ for any number x in S.
- Every non-empty subset of numbers has a **smallest element**.
- This property is called **well-ordering**.

Strong Induction

- define an operation for a number *n*, assuming it is defined for all numbers $x < n$.
  - $x < n$ means $x <= n$ and $x != n$.
- The set of numbers for which it is not defined must be empty.
- If it had a number *n*, it must have a smallest number *m*, but it is defined for *m*, since it is defined for all $x < m$, a contradiction.

Examples

- *even(0) := true; even(next(n)) := !even(n)*.
- *half(0) := 0*,
- *half(next(n)) := next(half(n)) if even(next(n))*

  := half(n) otherwise.
- *log(n) :=* undefined *if n = 0,*

  := 0, if n = next(0) (1),

  := next(log(half(n)) otherwise.

Examples

- $f(0) = f(1) = 1$.
- $f(n) = f(999*n/2)$ *if even(n) else n.*
- When n is even, $f(n)$ defined in terms of f value of a larger number.
- Function is still well-defined. Why?
- The largest power of 2 that divides $999*n/2$ is smaller than the largest power that divides n.

Examples

- $f(0) = f(1) = 1$
- $f(n) = f(n/2)$ if n is even else $f((3n+1)/2)$.
- Is this function well-defined?
- Don’t know, conjectured that $f(n) = 1$ for all n is the only function that satisfies this.
- Collatz problem, unsolved for nearly 300 years.

Recursion

- Recursion is essentially the only way of defining operations on numbers, or in general, other recursively defined data types.
- Define the operation for the base case.
- Assuming the operation is defined for `smaller' numbers, define for a number n.
- Guarantees operation is well-defined for all n.
- Same method used to prove properties.

Summary

- Recursively defined abstract data types.
- Operations also defined recursively.
- Induction used to prove properties.
- A clear precise way of defining.
- May not be the best way of implementing the type.
- Mainly useful for understanding properties.