PRIMITIVE RECURSIVE FUNCTIONS

BASE FUNCTIONS ARE PRFs

$C_a(\vec{x}) = a$ CONSTANTS

$I^n_i(x_1, \ldots, x_n) = x_i$ PROJECTIONS (IDENTITY)

$S(x) = x+1$ SUCCESSOR (INCREMENT)

BUILD MORE VIA

$F(\vec{x}) = H(G_1(\vec{x}), \ldots, G_k(\vec{x}))$

PRFs → H, G_1, …, G_k

COMPOSITION

$F(\vec{x}, 0) = G(\vec{x})$

$F(\vec{x}, y+1) = H(\vec{x}, y, F(x, y))$

PRFs → G, H

INDUCTION (PRIMITIVE RECURSION)

Building New PRFs

Addition: Formal

$$+(x,0) = I^1_1(x)$$

$$+(x,y+1) = S(I^3_3(x,y,+(x,y)))$$

Composition

Addition: Less Formal

$$x + 0 = x$$

$$x + (y+1) = (x+y) + 1$$

Multiplication: Formal

$$*(x,0) = C_0(x)$$

$$*(x,y+1) = H(x,y,*(x,y))$$

$$H(x,y,z) = +(I^3_1(x,y,z), I^3_3(x,y,z))$$

Multiplication: Less Formal

$$x * 0 = 0$$

$$x * (y+1) = x * y + x$$

More Basic Arithmetic

Predecessor: (Limited)

$$0 - 1 = 0$$

$$(x+1) - 1 = x$$

Subtraction: (Limited)

$$x \dot{-} 0 = x$$

$$x \dot{-} (y+1) = (x - y) - 1$$

Factorial:

$$0! = 1$$

$$(x+1)! = x! * (x+1)$$

RELATIONS

ISZERO:

$$0 == 0 = 1$$

$$(x+1) == 0 = 0$$

EQUALITY AND ONE OTHER:

$$x == y = ((x-y) + (y-x)) == 0$$

$$x \leq y = (x-y) == 0$$

BOOLEANS

NEGATION

$$\sim x = x == 0$$

AND

$$x \&\& y = (x * y)$$

OR

$$x \| y = \sim((x==0) \&\& (y==0))$$

BOUNDED MINIMIZATION

$$f(x) = \mu z (z \le x)[P(z)] \quad \text{IF } \exists \text{ SUCH } z$$
$$= x+1 \quad \text{OTHERWISE}$$

$$f(0) = 1 - P(0)$$
$$f(x+1) = [f(x) * (f(x) \le x))$$
$$+ ((x+2-P(x+1)) * \sim (f(x) \le x))$$

$$f(x) = \mu z (z < x)[P(z)] \quad \text{IF } \exists \text{ SUCH } z$$
$$= x \quad \text{OTHERWISE}$$

$$f(0) = 0$$
$$f(x+1) = \mu z (z \le x)[P(z)]$$

DIVISION & DIVISIBILITY

DIVISION:

NEED A VALUE

$x // 0 = 0$

$x // (y+1) = \mu z (z < x) [(z+1) * (y+1) > x]$

DIVISIBILITY

$x \mid y = ((y // x) * x) == y$

EXPONENTS

POWER

$x \wedge 0 = 1$

$x \wedge (y+1) = x * (x \wedge y)$

ABBREVIATE x^y

PRIMALITY

$\text{FIRSTFACTOR}(x) = \mu z (2 \leq z \leq x) [z \mid x]$

0 IF NONE

$\text{ISPRIME}(x) = \text{FIRSTFACTOR}(x) = x \ \&\& \ (x > 1)$

$\text{PRIME}(0) = 2$

$\text{PRIME}(x+1) = \mu z (\text{PRIME}(x) < z \leq \text{PRIME}(x)! + 1) [\text{ISPRIME}(z)]$

ABBREVIATE PRIME(i) AS p_i

Pairing Functions

- **pair(x,y)** $= \langle x,y \rangle = 2^x (2y + 1) - 1$
- with inverses

  $\langle z \rangle_1 = \exp(z+1,0)$

  $\langle z \rangle_2 = (((z + 1) // 2^{\langle z \rangle_1}) - 1) // 2$
- These are very useful and can be extended to encode **n**-tuples

  $\langle x,y,z \rangle = \langle x, \langle y,z \rangle \rangle$ (note: stack analogy)

Pairing Function is 1-1 Onto

Prove that the pairing function $\langle x,y \rangle = 2^x (2y + 1) - 1$ is 1-1 onto the natural numbers.

Approach 1:

We will look at two cases, where we use the following modification of the pairing function, $\langle x,y \rangle + 1$, which implies the problem of mapping the pairing function to Z^+.

Case 1 ($x=0$)

Case 1:

For $x = 0$, $\langle 0,y \rangle+1 = 2^0(2y+1) = 2y+1$. But every odd number is by definition one of the form $2y+1$, where $y \geq 0$; moreover, a particular value of y is uniquely associated with each such odd number and no odd number is produced by $2^x(2y+1)$ when $x>0$. Thus, $\langle 0,y \rangle+1$ is 1-1 onto the odd natural numbers.

Case 2 (x > 0)

Case 2:

For $x > 0$, $\langle x,y\rangle+1 = 2^x(2y+1)$, where $2y+1$ ranges over all odd number and is uniquely associated with one based on the value of y (we saw that in case 1). 2^x must be even, since it has a factor of 2 and hence $2^x(2y+1)$ is also even. Moreover, from elementary number theory, we know that every even number except zero is of the form $2^x z$, where $x>0$, z is an odd number and this pair x,z is unique. Thus, $\langle x,y\rangle+1$ is 1-1 onto the even natural numbers, when $x>0$.

The above shows that $\langle x,y\rangle+1$ is 1-1 onto Z^+, but then $\langle x,y\rangle$ is 1-1 onto $\aleph$, as was desired.

μ Recursive

4^{th} Model

A Simple Extension to Primitive Recursive

μ Recursive Concepts

- All primitive recursive functions are algorithms since the only iterator is bounded. That's a clear limitation.
- There are algorithms like Ackerman's function that cannot be represented by the class of primitive recursive functions.
- The class of recursive functions adds one more iterator, the minimization operator (μ), read "the least value such that."

Ackermann's Function

- **A(1, j)=2j for j ≥ 1**
- **A(i, 1)=A(i-1, 2)** for **i ≥ 2**
- **A(i, j)=A(i-1, A(i, j-1))** for **i, j ≥ 2**
- Wilhelm Ackermann observed in 1928 that this is not a primitive recursive function.
- Ackermann's function grows too fast to have a for-loop implementation.
- The inverse of Ackermann's function is important to analyze Union/Find algorithm. Note: A(4,4) is a super exponential number involving six levels of exponentiation. $\alpha(n) = A^{-1}(n, n)$ grows so slowly that it is less than 5 for any value of n that can be written using the number of atoms in our universe.

Union/Find

- Start with a collection **S** of unrelated elements – singleton equivalence classes
- **Union(x,y)**, **x** and **y** are in **S**, merges the class containing **x** (**[x]**) with that containing **y** (**[y]**)
- **Find(x)** returns the canonical element of **[x]**
- Can see if **x≡y**, by seeing if **Find(x)==Find(y)**
- How do we represent the classes?
- You should have learned that in CS2

The μ Operator

- Minimization:

  If **G** is already known to be recursive, then so is **F**, where

  F(x1,…,xn) = μy (G(y,x1,…,xn) == 1)

- We also allow other predicates besides testing for one. In fact any predicate that is recursive can be used as the stopping condition.

PREDICATE