Computer Science 513

Modelling Computation of Functions I

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #2

Goals for Today

Goals for Today:

- Describe why attempts to formalize the notion of an "algorithm" — leading to the development of ***computability theory*** — started.
- Describe *one* of the strategies to formalize "algorithm" — by considering computation of ***functions***.
- Introduce ***primitive recursive functions*** — an early (unsuccessful, but still useful) attempt to define computable functions

What Came Before

- ***David Hilbert*** was a German mathematician, and one of the most influential mathematicians of the 19^{th} and early 20^{th} centuries.
- Following a "foundational crisis in mathematics" — a period including the discovery of several *paradoxes* — Hilbert attempted to base all existing mathematical theories on a complete set of axioms, and provide a proof that these axioms were consistent.

What Came Before

- ***Kurt Friedrich Gödel*** was a logician, mathematician and philosopher who dealt a serious blow to Hilbert's program, by publishing two ***incompleteness theorems*** in 1931.
- The first of these theorems states that there is no consistent set of axioms, that can be listed by an ***effective procedure*** (that is, "algorithm") that is capable of proving all true statements about the natural numbers.

What Came Before

- Initially, the notion of an "effective procedure" was only understood informally. Gödel realized, and argued, that a ***formal*** definition of an "algorithm" was needed if his work was to continue.
- Gödel's investigations, and discussions, motivate much of the work that will be described next.
- For additional information about the "foundational crisis in mathematics", Hilbert's program, and Gödel's contribution — along with more information about the beginnings of computability theory — see Part I of Borut Robič's text *The Foundations of Computability Theory* (second edition available online through course web site).

Formalizing the Notion of an "Algorithm"

Attempts to formalize the notion of an "algorithm", beginning in the 1930's, can be grouped into three types:

- Modelling computation of ***functions***.
- Modelling computation by ***humans***.
- Modelling computation of ***languages***.

All three of these will described near the beginning of this course.

Modelling Computations of Functions

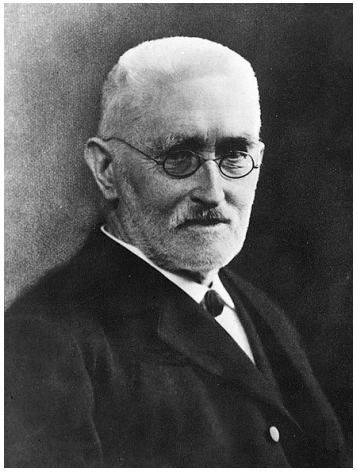
- Let $\mathbb{N}$ be the set of "natural numbers" — defined in this course, to be the set of non-negative integers

$$\mathbb{N} = \{0, 1, 2, \dots\} = \{n \in \mathbb{Z} \mid n \geq 0\}.$$

- Early attempts to define "computable functions" included studies of functions $f : \mathbb{N}^k \to \mathbb{N}$ for each non-negative integer k.

Primitive Recursive Functions

- This included a study of a set of functions that had already been identified.



Julias Wilhelm Richard Dedekind — commonly known as Richard Dedekind — was a German mathematician who made contributions to what is now thought of as “pure mathematics”.

One of the methods to construct functions that will shortly be defined — ***primitive recursion*** — had already been introduced and studied by Dedekind in 1888.

Primitive Recursive Functions

- ***Thoralf Albert Skolem*** was a Norwegian mathematician who worked in mathematical logic and set theory.
- Skolem proposed “primitive recursive arithmetic”, including a definition for the primitive recursive functions, which follows, in 1923 — well before Gödel’s work.

Primitive Recursive Functions: Composition

Definition: Let $f : \mathbb{N}^k \to \mathbb{N}$ and let $g_i : \mathbb{N}^n \to \mathbb{N}$ for $1 \leq i \leq k$. Let

$$h(x_1, x_2, \ldots, x_n) =$$
$$f(g_1(x_1, x_2, \ldots, x_n), g_2(x_1, x_2, \ldots, x_n), \ldots, g_k(x_1, x_2, \ldots, x_n))$$

for all $x_1, x_2, \ldots, x_n \in \mathbb{N}$, so that $h : \mathbb{N}^n \to \mathbb{N}$. Then h is said to be obtained from f and $g_1, g_2, \ldots, g_k$ by ***composition***.

Note: If f and $g_1, g_2, \ldots, g_k$ are all *total* functions then so is h. Otherwise, $h(x_1, x_2, \ldots, x_k)$ is defined if and only if $g_i(x_1, x_2, \ldots x_n)$ are all defined, for $1 \leq i \leq n$, so that $g_i(x_1, x_2, \ldots, x_n) = y_i$ for some $y_i \in \mathbb{N}$, and $f(y_1, y_2, \ldots, y_k)$ is defined as well.

Primitive Recursive Functions: Composition

Example: Suppose that $k = 2$, $n = 2$ as well, and that

- $g_1(x_1, x_2) = x_1 - x_2$,
- $g_2(x_1, x_2) = x_2 - x_1$, and
- $f(x_1, x_2) = \max(x_1, x_2)$.

Then, if $h : \mathbb{N}^2 \to \mathbb{N}$ is as defined above, — that is,

$$h(x_1, x_2) = f(g_1(x_1, x_2), g_2(x_1, x_2))$$

for all $x_1, x_2 \in \mathbb{N}$, then h has been obtained from f, g_1 and g_2 by composition, and

$$h(x_1, x_2) = |x_1 - x_2|.$$

Primitive Recursive Functions: Primitive Recursion, Part One

Definition: Let k be any fixed nonnegative integer, and let $g : \mathbb{N}^2 \to \mathbb{N}$ be a ***total*** function. Consider a ***total*** function $h : \mathbb{N} \to \mathbb{N}$ such that

(a) $h(0) = k$, and

(b) $h(t+1) = g(t, h(t))$ for every integer t such that $t \geq 0$.

Then h is said to be obtained from g using ***primitive recursion*** (or just ***recursion***).

Primitive Recursive Functions: Primitive Recursion, Part One

Example: Suppose that $k = 0$ and that $g(x_1, x_2) = x_1 + x_2 + 1$ for all $x_1, x_2 \in \mathbb{N}$, so that $g : \mathbb{N}^2 \to \mathbb{N}$ is a total function. Suppose h is defined from k and g as given above.

- $h(0) = k = 0$.
- $h(1) = g(0, h(0))$
  $= g(0, 0)$
  $= 0 + 0 + 1 = 1$.
- $h(2) = g(1, h(1))$
  $= g(1, 1)$
  $= 1 + 1 + 1 = 3$.

Primitive Recursive Functions
Primitive Recursion, Part One

Indeed

$$\begin{aligned} h(\ell+1) &= g(\ell, h(\ell)) \\ &= \ell + h(\ell) + 1 \\ &= h(\ell) + (\ell+1) \end{aligned}$$

for every integer $\ell \geq 0$. It can be proved, by induction on ℓ, that

$$h(\ell) = \sum_{i=0}^{\ell} i = \frac{\ell(\ell+1)}{2}$$

for every integer $\ell \geq 0$.

Primitive Recursive Functions
Primitive Recursion, Part Two

Definition: Let n be an integer such that $n \geq 1$. Let $f : \mathbb{N}^n \to \mathbb{N}$ and $g : \mathbb{N}^{n+2} \to \mathbb{N}$ be ***total*** functions. Consider a ***total*** function $h : \mathbb{N}^{n+1} \to \mathbb{N}$ such that, for all $x_1, x_2, \ldots, x_n \in \mathbb{N}$,

(a) $h(x_1, x_2, \ldots, x_n, 0) = f(x_1, x_2, \ldots, x_n)$

and, for every integer $t \geq 0$,

(b) $h(x_1, x_2, \ldots, x_n, t+1) = g(t, h(x_1, x_2, \ldots, x_n, t), x_1, x_2, \ldots, x_n)$.

Then h is said to be obtained from f and g using ***primitive recursion*** (or just ***recursion***).

Note: Part one of this definition can be thought of as a special case of part two, where $n = 0$, so that f is the *constant* function k.

Primitive Recursive Functions
Primitive Recursion, Part Two

Example: Let $n = 1$. Suppose that $f : \mathbb{N} \to \mathbb{N}$ such that $f(x_1) = 1$ for all $x_1 \in \mathbb{N}$. Suppose, as well, that $g : \mathbb{N}^3 \to \mathbb{N}$ such that $g(x_1, x_2, x_3) = x_2 \times x_3$ for all $x_1, x_2, x_3 \in \mathbb{N}$. Then, for all $x_1 \in \mathbb{N}$,

- $h(x_1, 0) = f(x_1) = 1$.
- $\begin{aligned} h(x_1, 1) &= g(0, h(x_1, 0), x_1) \\ &= g(0, 1, x_1) \\ &= 1 \times x_1 = x_1. \end{aligned}$
- $\begin{aligned} h(x_1, 2) &= g(1, h(x_1, 1), x_1) \\ &= g(1, x_1, x_1) \\ &= x_1 \times x_1 = x_1^2. \end{aligned}$

Primitive Recursive Functions
Primitive Recursion, Part Two

Indeed, for every positive integer $\ell \geq 0$,

$$\begin{aligned} h(x_1, \ell + 1) &= g(\ell, h(x_1, \ell), x_1) \\ &= h(x_1, \ell) \times \ell. \end{aligned}$$

This can be used to show, induction on ℓ, that $h(x_1, \ell) = x_1^\ell$ for all integers $x_1, \ell \in \mathbb{N}$ — provided that 0^0 is "defined" to be 1.

Primitive Recursive Functions
Initial Functions

Definition: The ***initial functions*** are as follows.

- *Nullify Function:*[1] The function $n : \mathbb{N} \to \mathbb{N}$ such that $n(x) = 0$ for all $x \in \mathbb{N}$ — also called the function $\zeta : \mathbb{N} \to \mathbb{N}$ in some references.
- *Successor Function:* The function $s : \mathbb{N} \to \mathbb{N}$ such that $s(x) = x + 1$ for all $x \in \mathbb{N}$ — Is called the function $\sigma : \mathbb{N} \to \mathbb{N}$ in some references.

[1] The instructor will probably accidentally call this the "nullity function" sometimes. It will mean the same thing.

Primitive Recursive Functions
Initial Functions

- *Projection Functions:* For integers i and k such that $1 \leq i \leq k$, these include the function $u_i^k : \mathbb{N}^k \to \mathbb{N}$ such that

$$u_i^k(x_1, x_2, \ldots, x_k) = x_i$$

for all $x_1, x_2, \ldots, x_k \in \mathbb{N}$. Also called the function π_i^k in some references.

Note: Each of these functions is a ***total*** function.

Primitive Recursive Functions
Recursive Definition

Definition: A function is ***primitive recursive*** if it can be obtained from the initial functions using a finite number of applications of composition and primitive recursion:

(a) Each of the *initial functions* is a primitive recursive function:

- (i) The nullify function, $n : \mathbb{N} \to \mathbb{N}$, is a primitive recursive function.
- (ii) The successor function, $s : \mathbb{N} \to \mathbb{N}$, is a primitive recursive function.
- (iii) For all integers i and k such that $1 \leq i \leq k$, the projection function u_i^k is a primitive recursive function.

Primitive Recursive Functions
Recursive Definition

(b) If $k, n \geq 1$, $f : \mathbb{N}^k \to \mathbb{N}$ is primitive recursive, and $g_i : \mathbb{N}^n \to \mathbb{N}$ is primitive recursive for $1 \leq i \leq k$, then the function $h : \mathbb{N}^n \to \mathbb{N}$ such that, for $x_{1,2}, \ldots, x_n \in \mathbb{N}$,

$$h(x_1, x_2, \ldots, x_n) = f(g_1(x_1, x_2, \ldots, x_n), \\ g_2(x_1, x_2, \ldots, x_n), \ldots, g_k(x_1, x_2, \ldots, x_n))$$

— obtained by *composition* — is primitive recursive.

Primitive Recursive Functions
Recursive Definition

(c) If k is a fixed nonnegative integer and $g : \mathbb{N}^2 \to \mathbb{N}$ is a primitive recursive function then the function $h : \mathbb{N} \to \mathbb{N}$ such that

- $h(0) = k$ and,
- $h(t+1) = g(t, h(t))$ for every integer $t \geq 0$

— obtained using the first kind of *primitive recursion* — is also primitive recursive.

Primitive Recursive Functions
Recursive Definition

(d) If n is a positive integer, $f : \mathbb{N}^n \to \mathbb{N}$ and $g : \mathbb{N}^{n+2} \to \mathbb{N}$ are both primitive recursive functions, then the function $h : \mathbb{N}^{n+1} \to \mathbb{N}$ such that, for all $x_1, x_2, \ldots, x_n \in \mathbb{N}$,

- $h(x_1, x_2, \ldots, x_n, 0) = f(x_1, x_2, \ldots, x_n)$, and
- $h(x_1, x_2, \ldots, x_n, t+1) = g(t, h(x_1, x_2, \ldots, x_n, t), x_1, x_2, \ldots, x_n)$

for every integer $t \geq 0$

— obtained using the second kind of *primitive recursion* — is primitive recursive too.

Primitive Recursive Functions
Example: Constant Functions

Consider the ***constant function*** $c_k(x_1) = k$ for any integer $k \geq 0$.

Claim #1: The above function $c_k : \mathbb{N} \to \mathbb{N}$ is a primitive recursive function for every non-negative integer k.

Proof: By induction on k. The standard form of mathematical induction will be used.

Basis: If $k = 0$ then it suffices to notice that $c_k = f_0$ is one of the initial functions, namely, the nullify function:

$$c_k(x_1) = c_1(x_1) = n(x_1) = 0$$

for all $x_1 \in \mathbb{N}$. It follows that $c_k = c_0$ is primitive recursive, as claimed.

Primitive Recursive Functions
Example: Constant Functions

Inductive Step: Let h be an integer such that $h \geq 0$. It is necessary and sufficient to use the following

> Inductive Hypothesis: The function $c_h : \mathbb{N} \to \mathbb{N}$ is primitive recursive.

to prove the following

> Inductive Claim: The function $c_{k+1} : \mathbb{N} \to \mathbb{N}$ is primitive recursive.

(a) With that noted, consider the ***successor function*** $s : \mathbb{N} \to \mathbb{N}$ such that $s(x_1) = x_1 + 1$. Since it is an initial function, this function is primitive recursive.

(b) It follows by the inductive hypothesis that the function $c_k : \mathbb{N} \to \mathbb{N}$ is primitive recursive.

Primitive Recursive Functions
Example: Constant Functions

(c) Note next that the function obtained from s and c_k by ***composition*** is c_{k+1}: For every integer $x_1 \geq 0$,

$$s(c_k(x_1)) = s(k) = k + 1 = c_{k+1}(x_1).$$

It follows that the function c_{k+1} is primitive recursive as well.

This establishes the inductive claim and completes the inductive step.

It now follows by induction that the constant function $c_k : \mathbb{N} \to \mathbb{N}$ such that $c_k(x_1) = k$ for all $x_1 \in \mathbb{N}$ is primitive recursive, as claimed. □

Primitive Recursive Functions
Example: Addition

Consider the addition function $plus : \mathbb{N}^2 \to \mathbb{N}$ such that

$$plus(x_1, x_2) = x_1 + x_2$$

for all $x_1, x_2 \in \mathbb{N}$.

(a) Let $f_0 : \mathbb{N} \to \mathbb{N}$ such that, for all $x_1 \in \mathbb{N}$, $f_0(x_1) = u_1^1(x_1) = x_1$. Then f_0 is primitive recursive because f_0 is one of the *initial functions* (namely, one of the projection functions).

(b) Let $f_1 : \mathbb{N}^3 \to \mathbb{N}$ such that, for all $t, y, x \in \mathbb{N}$, $f_1(t, y, x) = u_2^3(t, y, x) = y$. Then f_1 is primitive recursive because f_1 is one of the *initial functions* (namely, one of the projection functions).

Primitive Recursive Functions
Example: Addition

(c) Let $f_2 : \mathbb{N} \to \mathbb{N}$ such that, for all $x \in \mathbb{N}$, $f_2(x) = s(x) = x + 1$. Then f_2 is primitive recursive because f_2 is one of the *initial functions* (namely, the successor function).

(d) Let $f_3 : \mathbb{N}^3 \to \mathbb{N}$ such that, for all $t, y, x \in \mathbb{N}$,

$$\begin{aligned} f_3(t, y, x) &= f_2(f_1(t, y, x)) \\ &= f_2(y) && \text{(by the definition of } f_1\text{)} \\ &= y + 1 && \text{(by the definition of } f_2\text{).} \end{aligned}$$

Then f_3 is primitive recursive because f_3 has been produced from the primitive recursive functions f_2 and f_1 by *composition*.

Primitive Recursive Functions
Example: Addition

(e) Let $n = 1$ and let $f_4 : \mathbb{N}^2 \to \mathbb{N}$ be the function obtained from the above functions $f_0 : \mathbb{N} \to \mathbb{N}$ and $f_3 : \mathbb{N}^3 \to \mathbb{N}$ using primitive recursion. Then

$$f_4(x_1, 0) = f_0(x_1) = x_1$$

for all $x_1 \in \mathbb{N}$ and, for all $x_1, t \in \mathbb{N}$,

$$f_4(x_1, t+1) = f_3(t, f_4(x_1, t), x_1) = f_4(x_1, t) + 1.$$

Then f_4 is primitive recursive because f_4 has been produced from the primitive recursive functions f_1 and f_3 by the second form of *primitive recursion* (with $n = 1$).

Primitive Recursive Functions
Example: Addition

Claim #2: If the function $f_4 : \mathbb{N}^2 \to \mathbb{N}$ is as given above then

$$f_4(x_1, x_2) = plus(x_1, x_2) = x_1 + x_2$$

for all $x_1, x_2 \in \mathbb{N}$.

Proof: By induction on x_2. The standard form of mathematical induction will be used.

Basis: If $x_2 = 0$ then

$$\begin{aligned} f_4(x_1, x_2) &= f_4(x_1, x_2) \\ &= x_1 && \text{as shown in the definition of } f_4\text{, above} \\ &= x_1 + 0 \\ &= x_1 + x_2 = plus(x_1, x_2) \end{aligned}$$

as required.

Primitive Recursive Functions
Example: Addition

Inductive Step: Let k be an integer such that $k \geq 0$. It is necessary and sufficient to use the following

> Inductive Hypothesis: $f_4(x_1, k) = x_1 + k = plus(x_1, k)$ *for every integer* $x_1 \in \mathbb{N}$.

to prove the following

> Inductive Claim: $f_4(x_1, k+1) = x_1 + k + 1 = plus(x_1, k+1)$ *for every integer* $x_1 \in \mathbb{N}$.

Primitive Recursive Functions
Example: Addition

With that noted, let $x_1 \in \mathbb{N}$. Then

$$\begin{aligned} f_4(x_1, k+1) &= f_4(x_1, k) + 1 && \text{(by the definition of } f_4\text{, above)} \\ &= (x_1 + k) + 1 && \text{(by the inductive hypothesis)} \\ &= x_4 + (k+1) = \mathit{plus}(x_4, k+1) \end{aligned}$$

as required to establish the inductive claim and complete the inductive step.

It now follows, by induction on x_2, that

$$f_4(x_1, x_2) = x_1 + x_2 = \mathit{plus}(x_1, x_2)$$

for all $x_1, x_2 \in \mathbb{N}$. □

Thus the ***addition*** function is a primitive recursive function.

Primitive Recursive Functions
Example: Addition

Things to Note:

- The function f_4 was developed in a step-by-step way, using the inductive definition of a "primitive recursive function", in order to make it clear that this function really *is* primitive recursive.
- Since the form of this did not match the form of the "addition"" function, *plus*, a proof that the primitive recursive function, that had been generated, was *equal* to the desired addition function, was also given.
- This can be used as an example of how a proof that a function is primitive recursive can be organized.

Primitive Recursive Functions
Additional Examples

Additional, useful, functions that can be shown to be primitive recursive, using this technique, include the following:

- A "subtraction" function for the natural numbers: The function $monus : \mathbb{N}^2 \to \mathbb{N}$ such that, for all $x_1, x_2 \in \mathbb{N}$,

$$monus(x_1, x_2) = x_1 \dot{-} x_2 = \begin{cases} x_1 - x_2 & \text{if } x_1 \geq x_2, \\ 0 & \text{if } x_1 < x_2. \end{cases}$$

- A "multiplication" function: The function $mult : \mathbb{N}^2 \to \mathbb{N}$ such that, for all $x_1, x_2 \in \mathbb{N}$

$$mult(x_1, x_2) = x_1 \times x_2.$$

A supplement for this lecture includes proofs that these functions are primitive recursive.

Primitive Recursive Predicates

Definition A ***predicate*** is a total function $p : \mathbb{N}^n \to \{0, 1\}$ for some integer $n \geq 1$.

- If $x_1, x_2, \ldots, x_n \in \mathbb{N}$ then we generally think of this predicate as being ***true*** for the sequence of inputs $x_1, x_2, \ldots, x_n$ if

$$p(x_1, x_2, \ldots, x_n) = 1.$$

  We think of the predicate as being ***false*** for this sequence of inputs if

$$p(x_1, x_2, \ldots, x_n) = 0.$$

Primitive Recursive Predicates
Example: Checking Whether $x_1 > 0$

Consider the predicate $\mathit{nonzero} : \mathbb{N} \to \{0, 1\}$ such that, for all $x_1 \in \mathbb{N}$,

$$\mathit{nonzero}(x_1) = \begin{cases} 0 & \text{if } x_1 = 0, \\ 1 & \text{if } x_1 > 0. \end{cases}$$

This can be thought of as being "true"' if and only if $x_1 > 0$. Since $x_1 \in \mathbb{N}$, this is equivalent to the condition that x_1 is "nonzero".

- This predicate can be shown to be primitive recursive using the same kind of process as was used for the above examples. The supplement for this lecture includes a proof that this predicate is primitive recursive.

Primitive Recursive Predicates
Example: Checking Whether $x_1 > x_2$

Next consider the predicate $gt : \mathbb{N}^2 \to \{0, 1\}$ such that, for all $x_1, x_2 \in \mathbb{N}$,

$$gt(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 > x_2 \\ 0 & \text{if } x_1 \leq x_2. \end{cases}$$

- Note that $x_1 > x_2$ if and only if $x_1 \dot{-} x_2 > 0$.
- ***Easy Exercise:*** Use the above information, along with primitive recursive functions that have already been established, to prove that the above predicate $gt : \mathbb{N}^2 \to \{0, 1\}$ is primitive recursive.

Closure Properties
Closure Under Logical Operations

The following claims can sometimes be used to prove that functions (and predicates) more easily than by using the process that has been applied, so far.

Claim #3: Let n be a positive integer and let $p, q : \mathbb{N}^n \to \{0, 1\}$ be primitive recursive predicates.

(a) $\neg p$ is a primitive recursive predicate.

(b) $p \wedge q$ is a primitive recursive predicate.

(c) $p \vee q$ is a primitive recursive predicate.

Closure Property: Definition by Cases

Claim #4: Suppose that $f, g : \mathbb{N}^n \to \mathbb{N}$ are primitive recursive functions and suppose that $p : \mathbb{N}^n \to \{0, 1\}$ is a primitive recursive predicate. Consider the function $h : \mathbb{N}^n \to \mathbb{N}$ such that, for all $x_1, x_2, \ldots, x_n \in \mathbb{N}$,

$$h(x_1, x_2, \ldots, x_n) = \begin{cases} f(x_1, x_2, \ldots, x_n) & \text{if } p(x_1, x_2, \ldots, x_n) = 1, \\ g(x_1, x_2, \ldots, x_n) & \text{if } p(x_1, x_2, \ldots, x_n) = 0. \end{cases}$$

This function is also primitive recursive.

Application of Closure Properties
A Remainder Function

- Recall that if $x_1, x_2 \in \mathbb{N}$, and $x_2 \geq 1$, then

$$x_1 \bmod x_2$$

  is the unique integer, r, such that $0 \leq r \leq x_2 - 1$ and $x_1 - r$ is divisible by x_2. Thus $x_1 \bmod 1 = 0$ for all $x_1 \in \mathbb{N}$.
- Let $rem : \mathbb{N}^2 \to \mathbb{N}$ such that, for all $x_1, x_2 \in \mathbb{N}$,

$$rem(x_1, x_2) = \begin{cases} x_1 \bmod x_2 & \text{if } x_2 \geq 1, \\ 0 & \text{if } x_2 = 0. \end{cases}$$

Application of Closure Properties
A Remainder Function

- Note that — when $x_1, x_2 \in \mathbb{N}$ and $x_2 > 0$ —

$$rem(x_1 + 1, x_2) = \begin{cases} rem(x_1, x_2) + 1 & \text{if } rem(x_1, x_2) < x_2 - 1, \\ 0 & \text{otherwise.} \end{cases}$$

- This observation can be used, along with Claim #4, to prove that the "*rem*" function is also primitive recursive. Once again, the supplemental document for this lecture includes a proof, to show how this claim can be used.

To Be Continued

- It will be useful, for later proofs, to be able to show that functions are primitive recursive.
- Additional ways to do this, and additional examples of useful primitive recursive functions, will be included in the next lecture.