Computer Science 513

Encodings — Sequences and Strings

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #5

Goals

Goals for Today:

- Introduce various ***encoding functions*** that allow sequences of integers, and strings of symbols over an alphabet, to be represented by integers.
- This will, effectively, allow input variables, local variables and the output variable of an $\mathcal{S}$-program to represent sequences or strings.
- It will also give us a way to define "primitive recursive", "μ-recursive", and "$\mathcal{S}$-computable" functions of strings of symbols.

Expectations for Students

- Students will be expected to know the definitions of all of the "encoding functions" introduced in this lecture and to be able to compute the values of these functions on (reasonably small or simple) inputs — so that details of later proofs can be understood.
- Students will also be expected to understand definitions, and claims about encoding functions, that are given in these lecture notes.
- Proofs of these claims are, generally, less important — and are included in a supplemental document.

Encoding Pairs of Natural Numbers

First Objective: A way to encode an *ordered pair* of natural numbers using a single natural number

- Consider the ***pairing function*** $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N}$ such that, for $x, y \in \mathbb{N}$,
$$\langle x, y \rangle = 2^x(2y + 1) \dot{-} 1.$$
- Note that, since $2^x \geq 1$ for all $x \in \mathbb{N}$ and $2y + 1 \geq 1$ for all $y \in \mathbb{N}$ as well,
$$2^x(2y + 1) = \langle x, y \rangle + 1 \geq 1,$$
so that
$$\langle x, y \rangle = 2^x(2y + 1) - 1 \in \mathbb{N}$$
for all $x, y \in \mathbb{N}$.

Encoding Pairs of Natural Numbers

- For all $z \in \mathbb{N}$, there exists *exactly* one pair of values $\ell(z) \in \mathbb{N}$ and $r(z) \in \mathbb{N}$ such that $\langle \ell(z), r(z) \rangle = z$:
  - $\ell(z)$ is the largest nonnegative integer $k \in \mathbb{N}$ such that $z + 1$ is divisible by 2^k.
  - Since $z + 1 = \langle \ell(z), r(z) \rangle + 1 = 2^{\ell(z)} \cdot (2r(z) + 1)$,

$$r(z) = \frac{((z+1)/2^{\ell(z)}) - 1}{2}.$$

Note that (by the definition of $\ell(z)$) $(z+1)/2^{\ell(z)}$ is always odd, so that $r(z) \in \mathbb{N}$.

Encoding Pairs of Natural Numbers

Claim #1: The above pairing function $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N}$ is primitive recursive.

Claim #2: The above inverse functions $\ell, r : \mathbb{N} \to \mathbb{N}$ are primitive recursive as well.

These are proved by using the methods to prove that functions are primitive recursive described in Lectures #2–#3, working with the functions proved to be primitive recursive in those lectures. The supplemental material for this lecture includes sketches of the proofs of both of these claims.

Encoding Sequences of Natural Numbers with a Fixed Length

Second Objective: A way to encode a *sequence* of natural numbers — of some fixed length k — using a single natural number

Let k be a positive integer.

- The pairing function and its inverses will be extended — through the use of *composition* — to obtain bijective (one-to-one and onto) mappings

$$\langle \cdot, \cdot, \ldots, \cdot \rangle_k : \mathbb{N}^k \to \mathbb{N}$$

as well as "inverse functions" $\rho_i^k : \mathbb{N} \to \mathbb{N}$ for $1 \leq i \leq n$ such that, for all $x_1, x_2, \ldots, x_k, z \in \mathbb{N}$, if

$$z = \langle x_1, x_2, \ldots, x_k \rangle_k$$

then $x_i = \rho_i^k(z)$ for $1 \leq i \leq k$.

Encoding Sequences of Natural Numbers with a Fixed Length

- If $k = 1$ then $\langle x_1 \rangle_1 = \rho_1^1(x_1) = u_1^1(x_1) = x_1$, so that $\langle \cdot \rangle_1$ and ρ_1^1 are both primitive recursive.
- If $k = 2$ then, for all $x_1, x_2, z \in \mathbb{N}$,

$$\langle x_1, x_2 \rangle_2 = \langle x_1, x_2 \rangle, \quad \rho_1^2(z) = \ell(z) \text{ and } \rho_2^2(z) = r(z),$$

so that $\langle \cdot, \cdot \rangle_2$, ρ_1^2 and ρ_2^2 are all primitive recursive as well.

Encoding Sequences of Natural Numbers with a Fixed Length

- If $k \geq 2$ then $\langle \cdot, \cdot, \ldots, \cdot \rangle_{k+1}$ can now be defined inductively: For all $x_1, x_2, \ldots, x_{k+1} \in \mathbb{N}$,

$$\langle x_1, x_2, \ldots, x_{k+1} \rangle_{k+1} = \langle \langle x_1, x_2, \ldots, x_k \rangle_k, x_{k+1} \rangle.$$

Then, for $1 \leq i \leq k+1$ and $z \in \mathbb{N}$,

$$\rho_i^{k+1}(z) = \begin{cases} \rho_i^k(\ell(z)) & \text{if } 1 \leq i \leq k, \\ r(z) & \text{if } i = k+1. \end{cases}$$

Encoding Sequences of Natural Numbers with a Fixed Length

- It is easily proved by induction on k that the functions $\langle \cdot, \cdot, \ldots, \cdot \rangle : \mathbb{N}^k \to \mathbb{N}$ and the functions $\rho_i^k : \mathbb{N} \to \mathbb{N}$ are primitive recursive for $1 \leq i \leq k$, and that if

$$z = \langle x_1, x_2, \ldots, x_k \rangle_k$$

for $k \geq 1$ and $z, x_1, x_2, \ldots, x_k \in \mathbb{N}$, then $x_i = \rho_i^k(x)$, for $1 \leq i \leq k$, as claimed.

Encoding Finite Sequences of Natural Numbers

Third Objective: A way to encode any finite sequence of natural numbers — using a single natural number

- Recall the ***Fundamental Theorem of Arithmetic:*** Every natural number $n \geq 2$ has a ***unique*** factorization

$$n = q_1^{e_1} q_2^{e_2} \dots q_k^{e_k}$$

where $k \geq 1$, $q_1, q_2, \dots, q_k$ are distinct ***primes*** — so that without loss of generality (reordering as needed)

$$2 \leq q_1 < q_2 < \dots < q_k,$$

and $e_1, e_2, \dots, e_k$ are positive integers.

Encoding Finite Sequences of Natural Numbers

Gödel Numbering provides a way to map finite sequences of natural numbers to natural numbers — which has some, but not all, of the properties we need.

- Every finite sequence is mapped to a non-negative integer.
- Unfortunately, more than one finite sequence is mapped to the *same* integer.
- Applications of Gödel numbering avoid this problem by treating all the sequences, that are mapped to a given natural number, in the same way.
- The exercises provide a (nonstandard) modification of Gödel numbering that allows every finite sequence to be mapped to a non-negative integer, in such a way that there is exactly one sequence mapped to any given integer — that is, such that the mapping is a bijection.

Encoding Finite Sequences of Natural Numbers

- The ***Gödel number*** of the empty sequence is denoted by $[\,]$ and is equal to 1.
- For every positive integer k, and for $x_1, x_2, \ldots, x_k \in \mathbb{N}$, the ***Gödel number*** of the sequence $x_1, x_2, \ldots, x_k$, with length k, is denoted by

$$[x_1, x_2, \ldots, x_k]$$

and is equal to

$$p_1^{x_1} p_2^{x_2} \ldots p_k^{x_k}$$

where p_i is the i^{th} prime for $1 \leq i \leq k$ — so that $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, and so on.

Encoding Finite Sequences of Natural Numbers

- For every non-negative integer k, the Gödel number, $[x_1, x_2, \ldots, x_k]$ of a sequence $x_1, x_2, \ldots, x_k$, with length k, will also be denoted as

$$[x_1, x_2, \ldots, x_k]_k$$

— so that $[., ., \ldots, .]_k$ effectively denotes a function from $\mathbb{N}^k$ to $\mathbb{N}$ — or the constant 1 when $k = 0$.

Encoding Finite Sequences of Natural Numbers

Claim #3: For every positive integer k, the above function $[.,.,\ldots,.]_k : \mathbb{N}^k \to \mathbb{N}$ — such that, for $x_1, x_2, \ldots, x_k \in \mathbb{N}$,

$$[x_1, x_2, \ldots, x_k]_k = \prod_{i=1}^{k} p_i^{x_i}$$

— is a primitive recursive function.

Encoding Finite Sequences of Natural Numbers

Defining "inverses" for these functions is complicated by the following.

- None of the functions $[.,.,\ldots,.]_k : \mathbb{N}^k \to \mathbb{N}$ (for a positive integer k) is ***surjective***. Indeed, the only natural numbers that are equal to $[x_1, x_2, \ldots, x_k]_k$, for any sequence $x_1, x_2, \ldots, x_k$ of natural numbers with length k, are the positive integers that are only divisible by (some of) the primes $p_1, p_2, \ldots, p_k$ — and not by the prime p_j, for any number $j \geq k + 1$.

Encoding Finite Sequences of Natural Numbers

- For every positive integer a, there exists a non-negative integer k and a sequence $x_1, x_2, \ldots, x_k$ of natural numbers such that
$$[x_1, x_2, \ldots, x_k] = a.$$
- However, if $k \geq 1$ and $x_1, x_2, \ldots, x_k \in \mathbb{N}$, then
$$\begin{aligned}[x_1, x_2, \ldots, x_k, 0] &= [x_0, x_1, \ldots, x_k, 0]_{k+1} \\ &= \prod_{i=1}^{k} p_i^{x_i} = [x_1, x_2, \ldots, x_k]_k \\ &= [x_1, x_2, \ldots, x_k].\end{aligned}$$

  Thus extending a sequence, by appending 0 to the of it, does not change its Gödel number — and the function mapping any finite sequence to its Gödel number is also not ***injective***.

Encoding Finite Sequences of Natural Numbers

With that noted, consider functions $(\cdot)_i : \mathbb{N} \to \mathbb{N}$ that are defined as follows.

- $(x)_0 = 1$ for all $x \in \mathbb{N}$.
- For every *positive* integer i, let $(x)_i = 0$ if $x = 0$ and let $(x)_i$ be the largest integer t such that x is divisible by p_i^t, otherwise.

Claim #4:

(a) The function $g : \mathbb{N}^2 \to \mathbb{N}$ such that for all $x, i \in \mathbb{N}$, $g(x, i) = (x)_i$, is primitive recursive.

(b) For every number $i \in \mathbb{N}$ the function from $\mathbb{N}$ to $\mathbb{N}$, mapping x to $(x)_i$, is primitive recursive.

Encoding Finite Sequences of Natural Numbers

Now consider a function $\mathrm{Lt} : \mathbb{N} \to \mathbb{N}$ such that $\mathrm{Lt}(0) = \mathrm{Lt}(1) = 1$ and, for every integer $x \geq 2$, $\mathrm{Lt}(x)$ is the smallest integer n such that

$$[(x)_1, (x)_2, \ldots, (x)_n] = x.$$

A number n with this property always exists, so $\mathrm{Lt} : \mathbb{N} \to \mathbb{N}$ is a well-defined total function.

Claim #5: The function $\mathrm{Lt} : \mathbb{N} \to \mathbb{N}$ is primitive recursive.

Computations on Strings
Encodings

Fourth Objective: A way to encode strings over an alphabet that supports simple computations on strings

Consider some (finite and nonempty) alphabet

$$\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$$

of size $n \geq 1$. (Choose some ordering of the symbols in Σ arbitrarily, if one is not already available.)

- Let $\#(\sigma_i) = i$, for $1 \leq i \leq n$.
- Let us define the encoding $\#(\lambda)$ of the ***empty string*** λ to be 0.

Computations on Strings
Encodings

- Suppose now that

$$\omega = \gamma_m \gamma_{m-1} \cdots \gamma_1 \gamma_0$$

is a string in $\Sigma^\star$ with length $m+1 \geq 1$. Define the encoding $\#(\omega)$ of ω to be

$$\#(\gamma_m) \cdot n^m + \#(\gamma_{m-1}) \cdot n^{m-1} + \cdots + \#(\gamma_1) \cdot n + \#(\gamma_0).$$

Computations on Strings
Encodings

- Let $\omega_{k,L} \in \Sigma^\star$ be the string with length k including k copies of σ_1. Then

$$\#(\omega_{k,L}) = n^{k-1} + n^{k-2} + \cdots + n + 1.$$

  Notice, in particular, that $\#(\omega_{1,L}) = 1 = \#(\lambda) + 1$.

- Let $\omega_{k,H} \in \Sigma^\star$ the the string with length k including k copies of σ_n. Then

$$\#(\omega_{k,H}) = n^k + n^{k-1} + \cdots + n^2 + n.$$

  Notice, in particular, that if $k \geq 1$ then $\#(w_{k+1,L}) = \#(w_{k,H}) + 1$.

Computations on Strings
Encodings

- If $\omega, \widehat{\omega} \in \Sigma^\star$ are strings with length k then

$$\#(\omega_{k,L}) \leq \#(\omega), \#(\widehat{\omega}) \leq \#(\omega_{k,H})$$

  and, furthermore, $\#(\omega) = \#(\widehat{\omega})$ if and only if $\omega = \widehat{\omega}$.
- Notice, as well, that $\#(w_k, H) - \#(w_k, L) + 1 = n^k$ — the number of strings in $\Sigma^\star$ with length k.
- Thus if $n \in \mathbb{N}$ then $n = \#(\omega)$ for *exactly one* string $\omega \in \Sigma^\star$.

Computations on Strings
Deciding Membership in Languages

Consider a ***language*** $L \subseteq \Sigma^\star$.

- Let us define a ***predicate*** $p_L : \mathbb{N} \to \{0, 1\}$ that corresponds to L: For all $n \in \mathbb{N}$,

$$p_L(n) = \begin{cases} 1 & \text{if } \omega \in L, \text{ where } \omega \in \Sigma^\star \text{ such that } \#(\omega) = n, \\ 0 & \textit{otherwise}. \end{cases}$$

Computations on Strings
Deciding Membership in Languages

- ***Definition:*** The language L is ***primitive recursive*** if the corresponding predicate, p_L, is a primitive recursive predicate.
- ***Definition:*** The language L is μ***-recursive*** if the corresponding predicate, p_L, is μ-recursive.
- ***Definition:*** The language L is $\mathcal{S}$***-decidable*** if the corresponding predicate, p_L, is an $\mathcal{S}$-computable predicate.

Computations on Strings
Partial Functions of Strings

Consider a ***(partial or total) function*** $f : (\Sigma^\star)^k \to \mathbb{N}$, for a positive integer k.

- Let us define a ***(partial or total) function*** $\widehat{f} : \mathbb{N}^k \to \mathbb{N}$ that corresponds to f: For all $x_1, x_2, \ldots, x_k \in \mathbb{N}$,

$$\widehat{f}(x_1, x_2, \ldots, x_k) = f(\omega_1, \omega_2, \ldots, \omega_k)$$
$$\text{where } \omega_i \in \Sigma^\star \text{ such that } \#(\omega_i) = x_i \text{ for } 1 \leq i \leq k.$$

  In particular — for $x_1, x_2, \ldots, x_k$ and $\omega_1, \omega_2, \ldots, \omega_k$ as above — $\widehat{f}(x_1, x_2, \ldots, x_k)$ is undefined if $f(\omega_1, \omega_2, \ldots, \omega_k)$ is undefined.

Computations on Strings
Partial Functions of Strings

- ***Definition:*** For $f : (\Sigma_1^\star)^k \to \mathbb{N}$ as above, the function f is ***primitive recursive*** if the corresponding function $\widehat{f} : \mathbb{N}^k \to \mathbb{N}$ is primitive recursive.
- ***Definition:*** For $f : (\Sigma_1^\star)^k \to \mathbb{N}$ as above, the function f is ***μ-recursive*** if the corresponding function $\widehat{f} : \mathbb{N}^k \to \mathbb{N}$ is μ-recursive.
- ***Definition:*** For $f : (\Sigma_1^\star)^k \to \mathbb{N}$ as above, the function f is ***$\mathcal{S}$-computable*** if the corresponding function $\widehat{f} : \mathbb{N}^k \to \mathbb{N}$ is $\mathcal{S}$-computable.

Computations on Strings
Functions of Strings

For alphabets Σ_1 and Σ_2, consider a ***(partial or total) function*** $f : (\Sigma_1^\star)^k \to \Sigma_2^\star$ for a positive integer k.

- Let us define a ***(partial or total) function*** $\widehat{f} : \mathbb{N}^k \to \mathbb{N}$ that corresponds to f: For all $x_1, x_2, \ldots, x_k \in \mathbb{N}$,

$$\widehat{f}(x_1, x_2, \ldots, x_k) = \#(f(\omega_1, \omega_2, \ldots, \omega_k))$$
$$\text{where } \omega_1, \omega_2, \ldots, \omega_k \in \Sigma_1^\star$$
$$\text{such that } \#(\omega_i) = x_i \text{ for } 1 \leq i \leq k.$$

  In particular — for $x_1, x_2, \ldots, x_k$ and $\omega_1, \omega_2, \ldots, \omega_k$ as above — $\widehat{f}(x_1, x_2, \ldots, x_k)$ is undefined if $f(\omega_1, \omega_2, \ldots, \omega_k)$ is undefined.

Computations on Strings
Functions of Strings

- ***Definition:*** For $f : (\Sigma_1^\star)^k \to \Sigma_2^\star$ as above, the function f is ***primitive recursive*** if the corresponding function $\widehat{f} : \mathbb{N}^k \to \mathbb{N}$ is primitive recursive.
- ***Definition:*** For $f : (\Sigma_1^\star)^k \to \Sigma_2^\star$ as above, the function f is μ***-recursive*** if the corresponding function $\widehat{f} : \mathbb{N}^k \to \mathbb{N}$ is μ-recursive.
- ***Definition:*** For $f : (\Sigma_1^\star)^k \to \Sigma_2^\star$ as above, the function f is $\mathcal{S}$***-computable*** if the corresponding function $\widehat{f} : \mathbb{N} \to \mathbb{N}$ is $\mathcal{S}$-computable.

Computations of Strings
Functions of Strings

One can define "primitive recursive", "μ-recursive", and "$\mathcal{S}$-computable" functions, for other kinds of functions of strings (for example, for functions whose inputs are a *pair* of strings over different alphabets) in the same way.

Computations on Strings
Primitive Recursive Functions of Strings

The following functions of strings are primitive recursive. Proofs of this are sketched in the supplemental document for this lecture.

(a) *Length of a String:* The function $f : \Sigma^\star \to \mathbb{N}$ such that, for all $\omega \in \Sigma^\star$, $f(\omega) = |\omega|$.

(b) *Concatenation of Strings:* The function

$$\text{concat} : \Sigma^\star \times \Sigma^\star \to \Sigma^\star$$

such that, for all $\alpha, \beta \in \Sigma^\star$, $\text{concat}(\alpha, \beta) = \alpha \cdot \beta$, the *concatenation* of the input strings.

Computations on Strings
Primitive Recursive Functions of Strings

(c) *Rightmost Symbol of a String:* The function

$$\text{rtend} : \Sigma^\star \to \Sigma^\star$$

such that $\text{rtend}(\lambda) = \lambda$ and such that if $\omega = \sigma_1\sigma_2 \ldots \sigma_k$ is a nonempty string in $\Sigma^\star$ with positive length k, then $\text{rtend}(\omega)$ is the rightmost symbol, σ_k, in ω.

(d) *Prefix Obtained by Deleting the Rightmost Symbol:* The function

$$\text{rtrunc} : \Sigma^\star \to \Sigma^\star$$

such that $\text{rtrunc}(\lambda) = \lambda$ and such that if $\omega = \sigma_1\sigma_2 \ldots \sigma_k$ is a nonempty string in $\Sigma^\star$ with positive length k, then $\text{rtrunc}(\omega)$ is the string $\sigma_1\sigma_2 \ldots \sigma_{k-1}$ with length $k - 1$ obtained by deleting the rightmost symbol of ω.

Computations on Strings
Primitive Recursive Functions of Strings

(e) *Leftmost Symbol of a String:* The function

$$\text{ltend} : \Sigma^\star \to \Sigma^\star$$

such that $\text{ltend}(\lambda) = \lambda$ and such that if $\omega = \sigma_1\sigma_2 \ldots \sigma_k$ is a nonempty string in $\Sigma^\star$ with positive length k, then $\text{ltend}(\omega)$ is the leftmost symbol, σ_1, in ω.

(f) *Suffix Obtained by Deleting the Leftmost Symbol:* The function

$$\text{ltrunc} : \Sigma^\star \to \Sigma^\star$$

such that $\text{ltrunc}(\lambda) = \lambda$ and such that if $\omega = \sigma_1\sigma_2 \ldots \sigma_k$ is a nonempty string in $\Sigma^\star$ with positive length k, then $\text{ltrunc}(\omega)$ is the string $\sigma_2\sigma_3 \ldots \sigma_k$ with length $k - 1$ obtained by deleting the leftmost symbol of ω.

Computations on Strings
Primitive Recursive Functions of Strings

When considering machines that perform computations on strings it will be useful to consider two alphabets at the same time.

- Suppose that Σ_1 and Σ_2 are alphabets such that

$$|\Sigma_1| = n_1 < n_2 = |\Sigma_2|.$$

  Suppose, in particular, that $\Sigma_1 \subset \Sigma_2$ and that every symbol in Σ_1 has the same encoding, as an element of Σ_2, as it does as an element of Σ_1.

- Consider the function $\text{upchange}_{n_1,n_2} : \mathbb{N} \to \mathbb{N}$ such that, for all $\omega \in \Sigma_1^\star$, if $\ell \in \mathbb{N}$ is the encoding of ω as a string in $\Sigma_1^\star$, then $\text{upchange}_{n_1,n_2}(\ell)$ is the encoding of ω as a string in $\Sigma_2^\star$.

Computations on Strings
Primitive Recursive Functions of Strings

- For each string $\omega \in \Sigma_2^\star$, let $\phi_{n_2,n_1}(\omega)$ be the string in $\Sigma_1^\star$ obtained by removing, from ω, all the symbols that are not in Σ_1.

  This function could defined inductively as follows: $\phi_{n_2,n_1}(\lambda) = \lambda$ and, for all $\omega \in \Sigma_2^\star$ and $\sigma \in \Sigma_2$,

$$\phi_{n_2,n_1}(\omega \cdot \sigma) = \begin{cases} \phi_{n_2,n_1}(\omega) \cdot \sigma & \text{if } \sigma \in \Sigma_1 \text{ (that is, } \#(\sigma) \leq n_1\text{),} \\ \phi_{n_2,n_1}(\omega) & \text{otherwise.} \end{cases}$$

Computations on Strings
Primitive Recursive Functions of Strings

- Let

$$\text{downchange}_{n_2,n_1} : \mathbb{N} \to \mathbb{N}$$

such that, for all $s \in \mathbb{N}$, if s is the encoding of a string $\omega \in \Sigma_2^\star$ and $\widehat{\omega} = \phi_{n_2,n_1}(\omega)$, then $\text{downchange}_{n_2,n_1}(s)$ is the encoding of $\widehat{\omega}$, when considered as a string in $\Sigma_1^\star$.

- It is shown in the supplemental document that the functions

$$\text{upchange}_{n_1,n_2} : \mathbb{N} \to \mathbb{N}$$

and

$$\text{downchange}_{n_2,n_1} : \mathbb{N} \to \mathbb{N}$$

are both primitive recursive.