XVIII

Eighteenth Lecture
AUTOMATA & FORMAL LANGUAGES
15 November 2023

Lecture XVII
page 11

§ 4.5 CHURCH'S RECURSIVE FUNCTIONS

4.5 Church's recursive functions

The following operations on partial functions were considered by Alonzo Church (1903–1995). The functions

[illegible] (projection functions) *

[illegible] (constant functions)

[illegible] (where [illegible], the successor function)

are called basic functions. We have already proved that all basic functions are computable.

Suppose $f: W^m \to W$ and $g_0, \ldots, g_{m-1}: W^k \to W$ are partial functions; then the partial function h defined by *

$$h(\vec{w}) := f(g_0(\vec{w}), \ldots, g_{m-1}(\vec{w}))$$

is called the composition of f with $(g_0, \ldots, g_{m-1})$. The notational convention used for operations applies here as well: if one term on the right hand side is undefined, then so is the left hand side.

Suppose $f: W^k \to W$ and $g: W^{k+2} \to W$ are partial functions; then the function h defined by the recursion equations

$$h(\vec{w}, \varepsilon) = f(\vec{w}) \text{ and}$$
$$h(\vec{w}, s(v)) = g(\vec{w}, v, h(\vec{w}, v))$$ *

is called the recursion result of f and g.

Suppose $f: W^{k+1} \to W$ is a partial function; then the partial function h defined by

[illegible]

is called the minimisation result of f.

We say that a class $\mathcal{C}$ of partial functions is closed under composition, recursion, or minimisation if, whenever $f, g, g_0, \ldots, g_{m-1}$ are in $\mathcal{C}$, then the composition of f with $(g_0, \ldots, g_{m-1})$, the recursion result of f and g, or the minimisation result of f, respectively, are in $\mathcal{C}$.

MINIMISATION NEXT TIME

Alonzo Church

Alonzo Church (1903–1995)

Born	June 14, 1903 Washington, D.C., US
Died	August 11, 1995 (aged 92) Hudson, Ohio, US
Citizenship	United States
Alma mater	Princeton University
Known for	Lambda calculus Simply typed lambda calculus Church encoding Church's theorem Church–Kleene ordinal Church–Turing thesis Frege–Church ontology Church–Rosser theorem Intensional logic

Church's recursive fns are defined by closure operations

BASIC FUNCTIONS: projections, constant, successor

COMPOSITION: concatenation

RECURSION: $h(\vec{w}, \varepsilon) = f(\vec{w})$
$h(\vec{w}, s(v)) = g(\vec{w}, v, h(\vec{w}, v))$

COMPOSITION $\vec{w} \in W^k$

$f: W^m \dashrightarrow W$

$g_i: W^k \dashrightarrow W \quad i < m$

Then

$$h(\vec{w}) := f(g_0(\vec{w}), \ldots, g_{m-1}(\vec{w}))$$

RECURSION $\vec{w} \in W^k$

$f: W^k \dashrightarrow W$

$g: W^{k+2} \dashrightarrow W$

Then

$$h(\vec{w}, \varepsilon) = f(\vec{w})$$
$$h(\vec{w}, s(v)) = g(\vec{w}, v, h(\vec{w}, v))$$

Example

STEP 1

STEP 2

STEP 3

STEP 4

STEP 5

$\pi_{1,0}(w) = w$ BASIC

$\pi_{3,2}(w,v,u) = u$ BASIC

s BASIC

$s \circ \pi_{3,2}(w,v,u) = s(u)$ COMPOSITION

$h(w,\varepsilon) = \pi_{1,0}(w)$ RECURSION

$h(w,s(v)) = s \circ \pi_{3,2}(w,v,h(w,v))$

Projection to component 0.

of 1-tople

Projection of 3-tople to comp. 2

Successor

Removing clutter from notation

$$h(w,\varepsilon) = w$$
$$h(w,s(v)) = s(h(w,v))$$

In $\mathbb{N}$-notation:

$$h(n,0) = n$$
$$h(n,m+1) = h(n,m)+1$$

GRASSMANN RECURSION EQUATION FOR ADDITION

$h(n,m) =: n+m$

ALSO:

$$g(n,0) := 0$$
$$g(n,m+1) := g(n,m) + n$$

GRASSMANN EQ. FOR MULTIPLICATION

Similarly for n^m.

Summary

The arithmetical functions $+, \cdot, \exp$ are constructible from the basic functions by composition and recursion.

This is what we mean.

$$(v,w) \longmapsto u \quad \text{iff} \quad \#u = \#v + \#w$$

$$(v,w) \longmapsto u \quad \text{iff} \quad \#u = \#v \cdot \#w$$

$$(v,w) \longmapsto u \quad \text{iff} \quad \#u = \#v^{\#w}$$

Observe The operations COMPOSITION & RECURSION preserve totality of functions.

MINIMISATION

Suppose $f : \mathbb{W}^{k+1} \dashrightarrow \mathbb{W}$ is a partial function, then the partial function h defined by

$$h(\vec{w}) := \begin{cases} v & \text{if for all } u \leq v, \text{ we have that } f(\vec{w}, u)\downarrow \text{ and} \\ & \quad v \text{ is } <\text{-minimal such that } f(\vec{w}, v) = \varepsilon \text{ or} \\ \uparrow & \text{otherwise} \end{cases}$$

is called the *minimisation result of* f.

$f : \mathbb{W}^{k+1} \dashrightarrow \mathbb{W}$

$$h(\vec{w}) := \begin{cases} v & \text{if f.a. } u \leq v \text{ we have} \\ & f(\vec{w}, u)\downarrow \text{ and} \\ & v \text{ is } <\text{-minimal s.t.} \\ & f(\vec{w}, v) = \varepsilon \\ \uparrow & \text{o/w} \end{cases}$$

either some $u \leq v$ has $f(\vec{w}, u)\uparrow$ or $\forall v$ $f(\vec{w}, v)\downarrow \neq \varepsilon$

Observation h can be a partial function, even if f was total.

Definition ① The class of **primitive recursive functions** is the smallest class containing the basic fn, closed under composition & recursion.

② The class of **recursive functions** is the smallest class containing the basic function & closed under composition, recursion, & minimisation.

RECURSION TREES

Very similarly to the parse trees of Chapter 3, we are now looking at RECURSION TREES.

Label	Arity	Branching number	Interpretation
$\mathsf{B}^{\pi}_{k,i}$	k	0	Projection
B^{c}_{k}	k	0	Constant
B^{s}	1	0	Successor
$\mathsf{C}_{n,k}$	k	$n+1$	Composition
R_{k}	$k+1$	2	Recursion
M_{k}	k	1	Minimisation

Table 2: Labels for recursion trees and their arities and branching numbers

Definition A RECURSION TREE $\mathbb{T} = (T, \ell)$ is a labelled tree where the labels come from Table 2

(i) for every $\alpha \in T$, $|\mathrm{succ}_T(\alpha)|$ is the branching number of $\ell(\alpha)$;

(ii) if $\ell(\alpha) = \mathsf{C}_{n,k}$, then the first successor of α has a label with arity n and all other successors of α have labels of arity k;

(iii) if $\ell(\alpha) = \mathsf{R}_{k}$, then the first successor of α has a label of arity k and the second successor has a label or arity $k+2$;

(iv) if $\ell(\alpha) = \mathsf{M}_{k}$, then the unique successor of α has a label of arity $k+1$.

Assign a function f_π to a recursive tree π by recursively reading the labels.

$$l(\varepsilon) = B^{\pi}_{k,i} \longrightarrow f_\pi := \pi_{k,i}$$

$$l(\varepsilon) = B^{c}_{k} \longrightarrow f_\pi := c_{k,\varepsilon}$$

$$l(\varepsilon) = B^{s} \longrightarrow f_\pi := s_{\pi'}$$

Suppose already know $f_{\pi'}$ for all proper subtrees of π, then if

$l(\varepsilon) = C_{n,k}$ $\longrightarrow$ if $\pi_0, \dots, \pi_n$ are the subtrees at successors of the root, then

$$f_\pi(\vec{\omega}) = f_{\pi_0}(f_{\pi_1}(\vec{\omega}), \dots, f_{\pi_n}(\vec{\omega}))$$

$l(\varepsilon) = R_k$ $\longrightarrow$ if π_0, π_1 are the subtrees at successors.

then f_π is the recursion result of f_{π_0}, f_{π_1}.

$l(\varepsilon) = M_k$ $\longrightarrow$ if π' is the subtree at successor, then

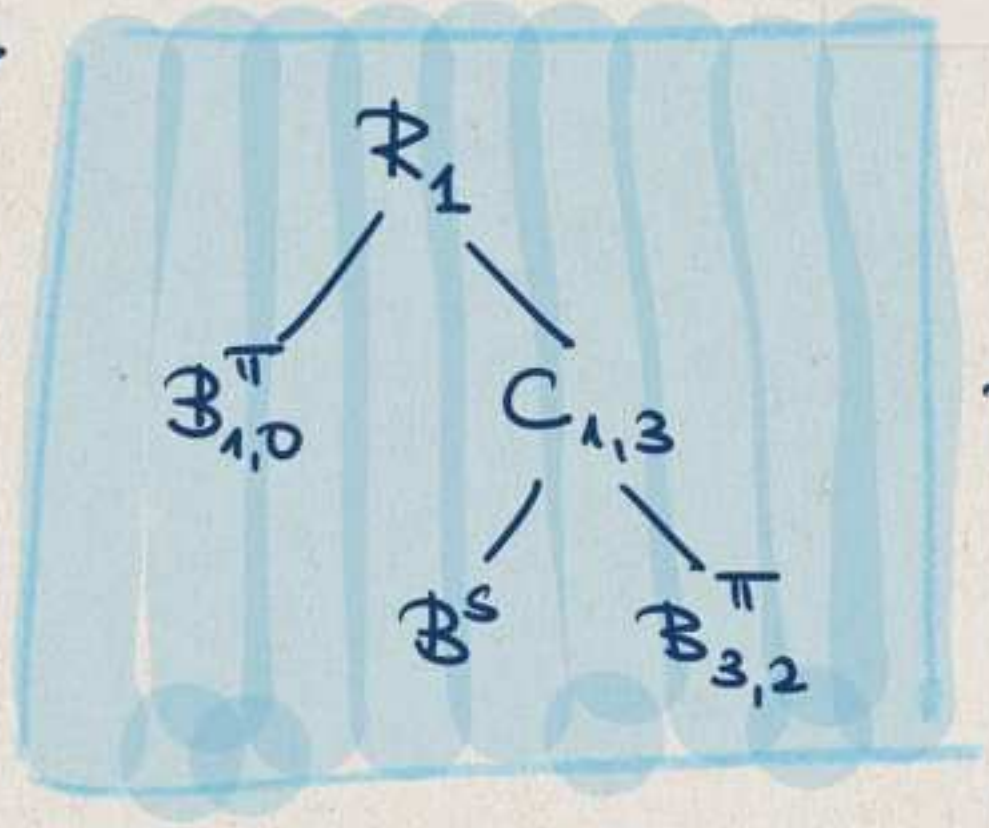
f_π is the minimisation result of $f_{\pi'}$

Example

Example

STEP 1	$\pi_{1,0}(w) = w$	BASIC
STEP 2	$\pi_{3,2}(w,v,u) = u$	BASIC
STEP 3	s	BASIC
STEP 4	$s \circ \pi_{3,2}(w,v,u) = s(u)$	COMPOSITION
STEP 5	$h(w,\varepsilon) = \pi_{1,0}(w)$ $h(w,s(v)) = s \circ \pi_{3,2}(w, v, h(w,v))$	RECURSION

T



Grassmann definition of addition

This is the recursion tree of addition

$f_\pi(w,v) = u$ iff $\#u = \#w \dotplus \#v$.

Proposition f is recursive iff $\exists T$ s.t. $f = f_T$.

[f is primitive recursive iff $\exists T$ s.t. ℓ has no label of the form M_k and $f = f_T$].

Proof. $\mathcal{R} = \{f ; f \text{ recursive}\}$ $\mathcal{R}' = \{f ; \exists T\ f = f_T\}$.

Clearly $\mathcal{R}'$ contains basic functions and is closed under CRM, so by minimality $\mathcal{R} \subseteq \mathcal{R}'$.

If $\mathcal{R}' \neq \mathcal{R}$, then there is T s.t. f_T is not recursive.

Take T of minimal height and obtain contradiction to the construction of f_T. q.e.d.

Remark This Proposition allows us to prove results about recursive or prim. rec. functions by ind. on height of recursion tree.

Theorem Every recursive function is computable.
[The converse is also true; discussed later.]

Prof. By above remarks, we just prove this by induction:

(1) Observe that all basic functions are computable.

(2) Observe that if $f, g_1, \dots, g_n$ are computable, then so is
$$h(\vec{w}) := f(g_1(\vec{w}), \dots, g_n(\vec{w}))$$
[Substitution Lemma]

(3) Remains:
Show that computable fns are closed under recursion & minimization.

NEXT PAGE

Lecture XIX

Recursion Fix f, g computable

$$h(\vec{w}, \varepsilon) = f(\vec{w})$$
$$h(\vec{w}, s(v)) = g(\vec{w}, v, h(\vec{w}, v))$$

Show h is computable.

Fix $\vec{w}, v$ & describe a RM that computes $h(\vec{w}, v)$.

Two scratch registers k & l.

Compute $f(\vec{w})$ and write into register l.

Check if $v = \varepsilon$. —YES→ Output register l.

↓ NO

REPEAT SUBROUTINE

UNTIL v is equal to register k.

SUBROUTINE

- Compute $g(\vec{w}, v_0, v_1)$ where v_0 is the content of reg. k & v_1 is the content of reg. l
- Write result into register l.
- Apply s to register k.

Output register l.

Minimization: will be done in Lecture XIX