Recursive Sets and Relations

Computability and Logic

The Plan

- Eventually, I will show that any Turing-computable* function is a recursive function, thereby closing the 'loop':
  - All Turing-computable* functions are recursive
  - All recursive functions are Abacus-computable* (already shown)
  - All Abacus-computable* are Turing-computable* (already shown)
- Thus, we will have shown that these three sets are exactly the same, providing evidence in favor of the Church-Turing Thesis.
- OK, but to show that any Turing-computable* function is a recursive function, I will need a whole lot more machinery:
  - I need to prove a bunch more functions to be recursive.
  - I will define recursive sets and relations … which will be a great help in showing certain functions to be recursive … and vice versa

Recursive Sets

- The characteristic function c_S of a set $S \subseteq N$ is defined as follows:
  - $c_S(x) = 1$ if $x \in S$
  - $c_S(x) = 0$ if $x \notin S$
- A set S is a recursive set iff its characteristic function c_S is a recursive function
- Examples of recursive sets
  - The empty set ($c_S = z$)
  - The set of all natural numbers ($c_S = const_1$)
  - The set of even numbers ($c_S = ?$)

Recursive Relations

- The characteristic function c_R of a relation $R \subseteq N^k$ is defined as follows:
  - $c_S(x_1, \ldots, x_k) = 1$ if $<x_1, \ldots, x_k> \in S$
  - $c_S(x_1, \ldots, x_k) = 0$ if $<x_1, \ldots, x_k> \notin S$
- A relation R is a recursive set iff its characteristic function c_R is a recursive function
- Examples of recursive relations: <, >, ≤, =

$$c_<(x, y) = sg(y \dot{-} x) \qquad c_>(x, y) = sg(x \dot{-} y)$$

$$c_\leq(x, y) = \overline{sg}(x \dot{-} y)$$

$$c_=(x, y) = \overline{sg}(x \dot{-} y) \times \overline{sg}(y \dot{-} x)$$

Finding new Recursive Functions and Relations

- In the next slides, we’ll go over a bunch of different methods to define new functions and relations (and sets, but they can be seen as 1-place relations) from existing ones.
- In each case, we can show that if the existing functions and relations are recursive, then the resulting functions and relations will be recursive as well.

Processes

- From functions to functions:
  - Composition, Recursion, Minimization (we saw this!)
- From functions and relations to functions:
  - Definition by Cases
- From functions and relations to relations:
  - Substitution
- From functions to relations:
  - Graph
- From relations to relations:
  - Logical operations
- From relations to functions:
  - Bounded Minimization and Maximization

Definition by Cases

- Suppose $f(x_1, \ldots, x_n)$ is defined by:
  - $f(x_1, \ldots, x_n) = g_1(x_1, \ldots, x_n)$ if $R_1(x_1, \ldots, x_n)$
  - …
  - $f(x_1, \ldots, x_n) = g_m(x_1, \ldots, x_n)$ if $R_m(x_1, \ldots, x_n)$
- Where:
  - $R_1 \ldots R_m$ are mutually exclusive
    - i.e. there is no $x_1, \ldots, x_n$, $i \neq j$: $R_i(x_1, \ldots, x_n)$ and $R_j(x_1, \ldots, x_n)$
  - $R_1 \ldots R_m$ are collectively exhaustive
    - i.e. for all $x_1, \ldots, x_n$ there is a i: $R_i(x_1, \ldots, x_n)$
- If:
  - $g_1 \ldots g_m$ are all recursive functions
  - $R_1 \ldots R_m$ are all recursive relations
- Then:
  - f is a recursive function
- Proof:
  - $f(x_1, \ldots, x_n) = g_1(x_1, \ldots, x_n) \times c_{R1}(x_1, \ldots, x_n) + \ldots + g_m(x_1, \ldots, x_n) \times c_{Rm}(x_1, \ldots, x_n)$

Example: min and max

- min(x,y) is a recursive function
- Proof: min(x,y) can be defined by cases:
  - min(x,y) = x if $x \leq y$
  - min(x,y) = y if $x > y$
- max(x,y) is a recursive function as well:
  - max(x,y) = x if $x > y$
  - max(x,y) = y if $x \leq y$

Substitution

- Given:
  - Relation $R(y_1, \ldots, y_m)$
  - Functions $f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$
- We can define relation $R'(x_1, \ldots, x_n)$ as follows:
  - $R'(x_1, \ldots, x_n)$ iff $R(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$
- If:
  - $R(y_1, \ldots, y_m)$ is a recursive relation
  - $f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$ are recursive functions
- Then:
  - R' is a recursive relation
- Proof:
  - $c_{R'}(x_1, \ldots, x_n) = c_R(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$

Example

- Consider relation $R(x,y,z)$ defined as follows:
  - $R(x,y,z)$ iff $y \times z \leq x$
- We see that R is the result of substituting the recursive function $\times$ into recursive relation $\leq$
- Thus, R is recursive
- (Technically, R is the result of substituting the functions $f_1(x,y,z) = y \times z$ and $f_2(x,y,z) = x$ into $\leq$, and we need to show that $f_1(x,y,z) = y \times z$ and $f_2(x,y,z) = x$ are recursive … but that’s trivial using the identity functions)

Graph

- Remember that any function $f:X \rightarrow Y$ can be seen as a relation defined over $X \times Y$
- The Graph operation will obtain a relationship from a function in exactly this manner.
  - Given function $f(x_1, \ldots, x_n)$
  - Define $R_f(x_1, \ldots, x_n, y)$ iff $f(x_1, \ldots, x_n) = y$
- If f is recursive, then R_f is recursive.
- Proof: R_f is the result of substituting recursive function f into recursive relation =

Logical Operations

- Given n-place relations R, R_1, and R_2 we can define:
  - $\neg R(x_1, \ldots, x_n)$ iff not $R(x_1, \ldots, x_n)$ (i.e. $\langle x_1, \ldots, x_n \rangle \notin R$)
  - $R_1 \wedge R_2(x_1, \ldots, x_n)$ iff $R_1(x_1, \ldots, x_n)$ and $R_1(x_1, \ldots, x_n)$
  - $R_1 \vee R_2(x_1, \ldots, x_n)$ iff $R_1(x_1, \ldots, x_n)$ or $R_1(x_1, \ldots, x_n)$
- If R, R_1, and R_2 are recursive, then $\neg R$, $R_1 \wedge R_2$, and $R_1 \vee R_2$ are recursive:
  - $c_{\neg R} = 1 - c_R$
  - $c_{R1 \wedge R2} = c_{R1} \times c_{R2}$ (or: $c_{R1 \wedge R2} = \min(c_{R1} , c_{R2})$)
  - $c_{R1 \vee R2} = sg(c_{R1} + c_{R2})$ (or: $c_{R1 \vee R2} = \max(c_{R1} , c_{R2})$)

Bounded Quantification

- Given n+1-place relation $R(x_1, \ldots, x_n, y)$, we can define:
  - $\exists v \leq u[R](x_1, \ldots, x_n, u)$ iff there exists some $v \leq u$ such that $R(x_1, \ldots, x_n, v)$
    – We'll simply write this as $\exists v \leq u\ R(x_1, \ldots, x_n, v)$
  - $\forall v \leq u[R](x_1, \ldots, x_n, u)$ iff for all $v \leq u$: $R(x_1, \ldots, x_n, v)$
    – We'll simply write this as $\forall v \leq u\ R(x_1, \ldots, x_n, v)$

- If R is recursive, then $\exists v \leq u[R]$ and $\forall v \leq u[R]$ are recursive as well:

$$c_{\exists v \leq u[R]}(x_1, \ldots, x_n, u) = sg(\sum_{v=0}^{u} c_R(x_1, \ldots, x_n, v))$$

$$c_{\forall v \leq u[R]}(x_1, \ldots, x_n, u) = \prod_{v=0}^{u} c_R(x_1, \ldots, x_n, v)$$

Using Strict Bounds

$$c_{\exists v<u[R]}(x_1,\ldots,x_n,u)=\begin{cases} c_{\exists v\le u[R]}(x_1,\ldots,x_n,pred(u)) & if\, 0<u \\ 0 & if\, 0=u \end{cases}$$

$$c_{\forall v<u[R]}(x_1,\ldots,x_n,u)=\begin{cases} c_{\forall v\le u[R]}(x_1,\ldots,x_n,pred(u)) & if\, 0<u \\ 1 & if\, 0=u \end{cases}$$

Example: Prime

- Consider the 1-place relation P(x) where P(x) iff x is prime (alternatively, consider the set P of all primes)
- P(x) is recursive (P is recursive) since:
  - $P(x)$ iff $1<x \wedge \neg\exists y<x\ \exists z<x\ y \times z = x$
  - That is: P(x) can be defined by applying the processes of logical operators ($\neg$, $\wedge$, and bounded quantification), substitution, and composition to other recursive functions ($const_1$ and $\times$) and recursive relations ($<$ and $=$).

Bounded Minimization and Maximization

- Given n+1-place relation $R(x_1, \ldots, x_n, y)$ define n+1-place functions Min[R] and Max[R]:
  - $Min[R](x_1, \ldots, x_n, w)$ = smallest $y \le w$ for which $R(x_1, \ldots, x_n, y)$ if such a y exists
  - $Min[R](x_1, \ldots, x_n, w) = w + 1$ if no such y exists
  - $Max[R](x_1, \ldots, x_n, w)$ = largest $y \le w$ for which $R(x_1, \ldots, x_n, y)$ if such a y exists
  - $Max[R](x_1, \ldots, x_n, w) = 0$ if no such y exists

Proof that Min[R] is Recursive if R is Recursive

If R is recursive, then Min[R] is recursive as well:

$$Min[R](x_1,\ldots,x_n,w)=\sum_{i=0}^{w}c_S(x_1,\ldots,x_n,i)$$

where c_S is the characteristic function of the relation S defined as $\forall t \leq i[\neg R](x_1, \ldots, x_n, i)$

Why This Works

Suppose we want to know Min[R](x,w), where R is defined as below:

w	R(x,w) (e.g.)	$\neg$R(x,w)	S(x,w) = $\forall t \leq w[\neg R](x,w)$ = $\forall t \leq w\ \neg R(x,t)$	$c_S(x,w)$	$\sum_{i=0}^{w} c_S(x,i)$
0	F	T	T	1	1
1	F	T	T	1	2
2	T	F	F	0	2
3	F	T	F	0	2
4	T	F	F	0	2
5	F	T	F	0	2

Verify that this is indeed Min[R](x,w)

Max[R] is Recursive too

- Left as HW question
  - Make sure to demonstrate that your function works by providing a similar table (and note that for the specific R relation as defined in that table, the Max column entries should be 0,0,2,2,4,4)

Example: quo and rem are Recursive

- Define quo(tient) and rem(ainder) functions as follows:
  - $quo(x,y)$ = the largest $z \leq x$ such that $z*y \leq x$ if $y > 0$
  - $quo(x,y) = 0$ if $y = 0$
  - $rem(x,y) = x - y*quo(x,y)$
- quo is recursive since it is a definition by cases where one of the cases uses the bounded maximization of a recursive relation (and every other function and relation used is recursive)
- rem is recursive since -, *, and quo are recursive.

Example: The Next Prime

- Let $\pi'(x)$ = the least y such that $x < y$ and y is prime
- $\pi'(x)$ is recursive, since it can be defined as the bounded minimization of a recursive relation:
  - $\pi'(x) = \mathrm{Min}[x<y \wedge \mathrm{Prime}(y)](x, x!+1)$
  - Explanation: $x!+1$ is not divisible by any number $\leq x$, so either $x!+1$ is prime itself or it has a prime factor greater than x … in either case, there exists a prime number greater than x but smaller or equal to $x!+1$

Example: Modified Logarithms

- Consider the following two modified logarithm functions lo(x,y) and lg(x,y):
  - lo(x,y) = the largest z such that y^z divides x if x and y > 1 where ‘x divides y’ iff for some z: z * x = y
  - lo(x,y) = 0 otherwise
  - lg(x,y) = the largest z such that $y^z \leq x$ if x > 1 and y > 1
  - lg(x,y) = 0 otherwise
- lo and lg are recursive, since they can be defined using bounded maximization (use x as upper bound) and other ‘recursive’ operations over recursive functions and relations (divides can be defined as bounded existential quantification (again, use x as bound))

Prime Coding and Decoding Functions

- A sequence $x_1, \ldots, x_k$ can be encoded using the following 'prime coding': $code(x_1, \ldots, x_n) = 2^n * 3^{x1} * 5^{x2} * \ldots \pi(n)^{xn}$ where $\pi(n)$ is the 'n-th' prime and where 2 is the '0-th' prime.
  - $\pi(n)$ is recursive, since $\pi(0) = 2$ and $\pi(n+1) = \pi'(\pi(n))$
  - $code(x_1, \ldots, x_n)$ is therefore recursive as well
- Given some code number s, the sequence can be decoded using the following (recursive) function:
  - $ent(s,i)$ = the i-th entry (the 0-th entry gives the length) = $lo(s, \pi(i))$