Notation: $M_n(x)\downarrow$ means that TM M_n **halts** on input x. Let us assume that if $M_n(x)\downarrow$, then $M_n(x)$ is defined, i.e., the output, $M_n(x)$, is whatever string is left between $\triangleright$ and the first $\sqcup$.

Thus, $\qquad M_n(x)\downarrow \quad \Leftrightarrow \quad M_n(x) \in \mathbf{N} \quad \Leftrightarrow \quad M_n(x) \neq \nearrow$

Fundamental Theorem of r.e. Sets: Let $S \subseteq \mathbf{N}$. T.F.A.E.

1. S is the domain of a partial, recursive function,

   i.e., for some $n \in \mathbf{N}$, $\qquad S \;=\; \{x \in \mathbf{N} \mid M_n(x)\downarrow\}$

2. $S = \emptyset$ or S is the range of a total, recursive function,

   i.e., for some total, recursive $M_m(\cdot)$, $\qquad S = \emptyset$ or $S = M_m(\mathbf{N})$

3. S is the range of a partial, recursive function,

   i.e., for some $r \in \mathbf{N}$, $\qquad S \;=\; M_r(\mathbf{N})$

4. S is r.e.,

   i.e., for some $t \in \mathbf{N}$, $\qquad S = W_t$,

Proof: $S = \{x \mid M_n(x)\downarrow\} \quad \Rightarrow \quad S = \emptyset \;\vee\; \exists m(S = M_m(\mathbf{N}))$

case 1: $S = \emptyset$. Thus S satisfies (2). $\surd$

case 2: $S \neq \emptyset$. let $a_0 \in S$.

Build TM M_m, which on input z does the following:

1. $x := L(z);\; y := R(z)$ $\qquad$ // i.e., $z = P(x, y)$
2. run $M_n(x)$ for y steps
3. **if** it converges **then return**(x)
4. $\qquad$ **else return**(a_0)

Claim: $S = M_m(\mathbf{N})$: $M_m(\mathbf{N}) \subseteq S$ $\surd$

$M_m(\mathbf{N}) \supseteq S$: Suppose $x \in S$.

Thus $M_n(x)$ converges in some number y of steps.

Therefore, $M_m(P(x, y)) = x$. $\surd$

[**Non-computable step in above construction:** no way to tell if we are in **case 1** or **case 2.**]

$S = \emptyset$ or $S = M_m(\mathbf{N}) \quad \Rightarrow \quad \exists r(S = M_r(\mathbf{N}))$

If $S = \emptyset$ then $S = M_0(\mathbf{N})$ where M_0 is a Turing machine that halts on no inputs. $r := 0$

Otherwise, $S = M_m(\mathbf{N})$, i.e., S is the range of the partial, recursive function $M_m(\cdot)$. $r := m$

[Even though $M_m(\cdot)$ is total, it is still considered a **partial, recursive function**. However, of course, $M_m(\cdot)$ is not **strictly partial**.]

$S = M_r(\mathbf{N}) \quad \Rightarrow \quad \exists t(S = W_t)$

Construct TM M_t, which on input x does the following:

1. **for** $i := 1$ to ∞ {
2. run $M_r(0), M_r(1), \ldots, M_r(i)$ for i steps each.
3. **if** any of these output x, **then return**(1)}

[The above construction is called **dove-tailing**.]

Claim: $M_r(\mathbf{N}) = \mathcal{L}(M_t)$.

Suppose $x \in M_r(\mathbf{N})$, i.e., $M_r(j) = x$, for some j,

computation takes k steps, for some k

At round $i = \max(j, k)$, $M_t(x)$ will halt and output "1". $\surd$

Suppose $x \notin M_r(\mathbf{N})$, then $M_t(x)$ will never halt. $\surd$

$S = W_t \quad \Rightarrow \quad \exists n(S = \{x \in \mathbf{N} \mid M_n(x)\downarrow\})$

Construct TM M_n, which on input x does the following:

1. run $M_t(x)$
2. **if** $(M_t(x) = 1)$ **then return**(1)
3. **else** run forever

Recall that, $\quad S = W_t = \mathcal{L}(M_t)$

Thus, $\quad S = \mathrm{dom}(M_n(\cdot)) = \{x \mid M_n(x)\downarrow\}$. $\square$

Reductions = Translations

Def. S is **reducible** to T ($S \leq T$) iff there exists a "very easy to compute" function $f : \mathbf{N} \rightarrow \mathbf{N}$, s.t. $\forall w \in \mathbf{N} \quad (w \in S \quad \Leftrightarrow \quad f(w) \in T)$.

Note: Later we will require $f \in F(\text{DSPACE}[\log n])$.

Note: f **translates** membership questions for S to membership questions for T. Thus, **S is no harder than T**.

$$\forall w \in \mathbf{N} \quad \chi_S(w) \quad = \quad \chi_T(f(w))$$
$$\forall w \in \mathbf{N} \quad (w \in S \quad \Leftrightarrow \quad f(w) \in T)$$

Sometimes the "$\Leftrightarrow$" in the definition of reductions makes students think that reductions go both ways, but that is not true, they only go from S to T. The reason for the "$\Leftrightarrow$" is that one arrow tells us that if $f(w) \in T$ then $w \in S$, and the arrow in the other direction tells us that if $f(w) \notin T$ then $w \notin S$. Thus the answer to the question, "Is $f(w) \in T$?", is also the answer to the question, "Is $w \in S$?".

Proposition 3.1 $K \leq A_{0,17} \;=\; \{n \mid M_n(0) = 17\}$

Proof: We want to build an easy-to-compute program translator $f_1 : \mathbf{N} \rightarrow \mathbf{N}$ such that,

Want: $\forall z \in \mathbf{N} \quad (z \in K) \quad \Leftrightarrow \quad (f_1(z) \in A_{0,17})$

Want: $\forall z \in \mathbf{N} \quad (M_z(z) = 1) \quad \Leftrightarrow \quad (M_{f_1(z)}(0) = 17)$.

Define $f_1(z)$ to be the following Turing Machine program, on input x,

1. **if** $x \neq 0$: **return**(34)
2. run $M_z(z)$
3. **if** $(M_z(z) = 1)$: **return**(17)
4. **return**(68)

Recall that we write $M_{f_1(z)}$ for the Turing Machine whose program is $f_1(z)$. Thus,

$$z \in K \;\Leftrightarrow\; M_z(z) = 1 \;\Leftrightarrow\; M_{f_1(z)}(0) = 17 \;\Leftrightarrow\; f_1(z) \in A_{0,17}$$

[In the proof of the above series of equivalences, note that if $M_z(z) = \nearrow$, then $M_{f_1(z)}(0) = \nearrow$.] $\square$

Proposition 3.2 $A_{0,17} \leq K$

Proof: We want to build any easy-to-compute program translator $f_2 : \mathbf{N} \to \mathbf{N}$ such that,

Want: $$\forall z \in \mathbf{N} \quad (z \in A_{0,17}) \quad \Leftrightarrow \quad (f_2(z) \in K)$$

Want: $$\forall z \in \mathbf{N} \quad (M_z(0) = 17) \quad \Leftrightarrow \quad (M_{f_2(z)}(f_2(z)) = 1)$$

Define $f_2(z)$ to be the following Turing Machine program, on input x,

1. run $M_z(0)$
2. **if** $(M_z(0) = 17)$: **return**(1)
3. **return**(0)

Thus,

$$z \in A_{0,17} \;\Leftrightarrow\; M_z(0) = 17 \;\Leftrightarrow\; \forall x \in N\,(M_{f_2(z)}(x) = 1) \;\Leftrightarrow\; M_{f_2(z)}(f_2(z)) = 1 \;\Leftrightarrow\; f_2(z) \in K$$

[In the proof of the above series of equivalences, note that if $M_z(0) = \nearrow$, then $M_{f_2(z)}(x) = \nearrow$ for all inputs x.] $\square$

Def. Let $C \subseteq \mathbf{N}$. C is **r.e.-complete** iff

1. $C \in$ r.e., and
2. $\forall A \in$ r.e. $(A \leq C)$

Intuition: C is a "hardest" r.e. set.

Th: K is r.e. complete.

Proof: We already know that K is r.e.

Let A be an arbitrary r.e. set, i.e., $A = W_i$ for some i.

Wanted: total recursive f, s.t.: $\forall n (n \in A \;\Leftrightarrow\; f(n) \in K)$

Define total, recursive f which on input n computes:

$$M_{f(n)} \quad = \quad \boxed{\text{Erase input}}\,\boxed{\text{Write } n}\,\boxed{M_i}$$

$M_{f(n)}$ ignores its input and instead runs $M_i(n)$.

$$n \in A \;\Leftrightarrow\; M_i(n) = 1 \;\Leftrightarrow\; \forall x (M_{f(n)}(x) = 1)$$

$$\Leftrightarrow\; M_{f(n)}(f(n)) = 1 \;\Leftrightarrow\; f(n) \in K \qquad \surd$$

Prop: Suppose C is r.e.-complete and:

1. $S \in$ r.e., and
2. $C \leq S$

then S is r.e.-complete.

Proof: Show: $\forall A \in$ r.e. $(A \leq S)$

Know: $\forall A \in$ r.e. $(A \leq C)$

Follows by transitivity of $\leq$: $A \leq C \leq S$. $\square$

Cor: $A_{0,17}$ is r.e.-complete.