Theorem 9.4: The busy beaver function, $\sigma(n)$, is eventually larger than any total, recursive function.

Theorem 9.5: There is a *Universal Turing Machine U* such that,

$$U(P(n,m)) \quad = \quad M_n(m)$$

Theorem 9.6: (Unsolvability of Halting Problem) Let,

$$\text{HALT} \quad = \quad \{P(n,m) \mid \text{TM } M_n(m) \text{ eventually halts}\}$$

Then, HALT is r.e. but not recursive.

Listing of all r.e. sets: $W_0, W_1, W_2, \cdots$

$$W_i \quad = \quad \{n \mid M_i(n) = 1\}$$

Corollary 9.8: Let,

$$K = \{n \mid M_n(n) = 1\} \quad = \quad \{n \mid U(P(n,n)) = 1\}$$

$$= \{n \mid n \in W_n\}$$

Then,

$$K \quad \in \quad \textbf{r.e.} - \textbf{Recursive} \ .$$

Notation: $M_n(x)\downarrow$ means that TM M_n **converges** on input x, i.e.,

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

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.,

$$(\exists n)(S \ = \ \text{dom}(M_n(\cdot)) \ = \ \{x \in \mathbf{N} \mid M_n(x)\downarrow\})$$

2. $S = \emptyset$ or S is the range of a total, recursive function, i.e., $S = \emptyset$ or S = range($M_n(\cdot)$) = $M_n(\mathbf{N})$, for some total, recursive function $M_n(\cdot)$.

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

$$S \ = \ M_n(\mathbf{N}), \quad \text{for some } n \in \mathbf{N} \ .$$

4. S is r.e., i.e., $S = W_n$, for some $n \in \mathbf{N}$

Proof: (Please learn this proof!)

$(1 \Rightarrow 2)$: Assume (1), $S = \{x \mid M_n(x)\downarrow\}$.

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

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

From M_n compute M_r, which on input z does the following:

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

Claim: $S = M_r(\mathbf{N}) = \{M_r(x) \mid x \in \mathbf{N}\}$.

$M_r(\mathbf{N}) \subseteq S$

$M_r(\mathbf{N}) \supseteq S$

Suppose $x \in S$.

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

Therefore, $M_r(P(x, y)) = x$.

Note the **non-computable step** in the construction: there is no way to tell whether we are in case 1 or case 2.

$(2) \Rightarrow (3)$: Assume **(2)**. If $S = \emptyset$ then $S = M_0(\mathbf{N})$ where M_0 is a Turing machine that halts on no inputs.

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

Note: Even though $M_n(\cdot)$ is total, it is still considered a "partial, recursive function". However, of course, $M_n(\cdot)$ is not "strictly partial".

$(3) \Rightarrow (4)$: Assume **(3),** $S = M_n(\mathbf{N})$.

From M_n we construct M_d, which on input x does the following:

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

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

Claim: $M_d(\cdot) = p_S(\cdot)$.

If $x \in S$, then $x \in$ range($M_n(\cdot)$).

So for some j and k, $M_n(j) = x$ and the computation takes k steps.

Thus, at round $i = \max(j, k)$, $M_d(x)$ will halt and output "1".

If $x \notin S$, then $M_d(x)$ will never halt.

Thus, $S = W_d = \{x \mid M_d(x) = 1\}$.

$(4) \Rightarrow (1)$: Assume (4), and thus $S = W_n$.

$$S \quad = \quad \{i \mid M_n(i) = 1\}$$

From M_n, construct M_d, which on input x does the following:

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

$$S \quad = \quad \{x \mid M_d(x)\downarrow\}$$

Thus, $S = \text{dom}(M_d(\cdot)) = \{x \mid M_d(x)\downarrow\}$. ♠

This theorem lets us put the “enumerable” in **r.e.**.

A nonempty language A is said to be **Turing enumerable** if there exists a TM that, when started on blank tape, lists the elements of A. The TM will take forever to do so if A is infinite, and it might repeat elements.

It should be pretty clear that for nonempty sets “Turing enumerable” means *exactly* “the range of a total recursive function”. So except for $\emptyset$, “Turing enumerable” means exactly “**r.e.**.”

An infinite set of numbers is *Turing enumerable in increasing order* if it is Turing enumerable by a machine that lists i before j whenever $i < j$.

It's pretty easy to see that an infinite set is Turing enumerable in increasing order iff it is recursive:

- $\Rightarrow$: Keep running the TM until you hit the target or pass it.
- $\Leftarrow$: Run through all numbers in increasing order and test each one, listing the ones that are in the language.

Definition 7.1 Let S and T be sets of numbers. We say that S is *reducible* to T $(S \leq T)$ iff there exists a total, recursive $f : \mathbf{N} \rightarrow \mathbf{N}$ such that:

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

♠

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

The notation "$S \leq T$" is meant to suggest "S is no more difficult than T". To use this notation, we should be confident that "$\leq$" is **reflexive** and **transitive** (You'll check this on HW#3.) The notation suggests as well that it is **anti-symmetric**, but it is not. It is quite possible to have $S \leq T$, $T \leq S$, and $S \neq T$ all be simultaneously true. In this case we say S and T are **equivalent**.

This kind of reduction is called a **many-one reduction**. Later we'll see another kind called a **Turing reduction**.

An Example:

$$A_{0,17} \quad = \quad \{n \mid M_n(0) = 17\}$$

Claim: $K \leq A_{0,17}$.

Proof: Define $f(n)$ as follows:

$M_{f(n)}$ =

erase input; write n	M_n	if 1 then write 17 else loop

$$n \in K \;\Leftrightarrow\; M_n(n) = 1 \;\Leftrightarrow\; M_{f(n)}(0) = 17 \;\Leftrightarrow\; f(n) \in A_{0,17}$$

$\spadesuit$

If $K \leq A_{0,17}$ really means "K is no harder than $A_{0,17}$" or equivalently "$A_{0,17}$ is no easier than K", then we should be able to conclude that $A_{0,17}$ is not recursive because K is not recursive. The next theorem will let us do this in general.

Fundamental Theorem of Reductions:

If $S \leq T$ are languages then:

1. If T is **r.e.**, then S is **r.e.**.
2. If T is co-**r.e.**, then S is co-**r.e.**.
3. If T is **Recursive**, then S is **Recursive**.

Moral: Suppose $S \leq T$. Then,

- If T is easy, then so is S.
- If S is hard, then so is T.

Another way to phrase this is that **r.e.**, co-**r.e.**, and **Recursive** are each **downward closed under reductions**.

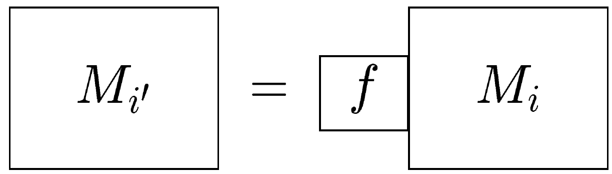
Proof: Let $f : S \leq T$, i.e., $(\forall x)(x \in S \ \Leftrightarrow \ f(x) \in T)$

1. Suppose $T \ = \ W_i \ = \ \{x \mid M_i(x) = 1\}$.

   From M_i compute the TM $M_{i'}$ which on input x does the following:

   (a) compute $f(x)$

   (b) run $M_i(f(x))$



   Then

   $$(x \in S) \ \Leftrightarrow \ (f(x) \in T) \ \Leftrightarrow (M_i(f(x)) = 1) \ \Leftrightarrow \ (M_{i'}(x) = 1)$$

   Therefore, $S = W_{i'}$, and we have shown that $S \in$ **r.e.,** as desired.

Recall our hypothesis for this proof:

$$f : S \leq T, \qquad \text{i.e.,} \qquad (\forall x)(x \in S \Leftrightarrow f(x) \in T)$$

The last two parts of the theorem follow directly from the first:

2. **Observation:** $S \leq T \quad \Leftrightarrow \quad \overline{S} \leq \overline{T}$.

$$T \in \text{co-}\textbf{r.e.} \Leftrightarrow \overline{T} \in \textbf{r.e.}, \overline{S} \in \textbf{r.e.} \Leftrightarrow S \in \text{co-}\textbf{r.e.}$$

3. $T \in$ **Recursive** $\quad \Rightarrow \quad (T \in \textbf{r.e.} \wedge T \in \text{co-}\textbf{r.e.}) \quad \Rightarrow$

$$(S \in \textbf{r.e.} \wedge S \in \text{co-}\textbf{r.e.}) \quad \Rightarrow \quad S \in \textbf{Recursive}$$

♠

Definition 7.2 Let $C \subseteq \mathbf{N}$. C is r.e.-*complete* iff

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

Intuition: C is a "hardest" r.e. set. In the "$\leq$" ordering, in that it is above all other r.e. sets. ♠

If you have seen a definition of "**NP**-complete", this definition should look familiar. **NP**-completeness was explicitly modeled on this historically earlier concept.

It is perhaps odd that there are any **r.e.**-complete sets at all – the definition doesn't suggest why there should be. But in fact we've already seen one.

Theorem 7.3 *K is r.e. complete.*

Proof: Let $A \in$ **r.e.** be arbitrary, so we know that $A = W_i$ for some i.

We want: $$(\forall n)(n \in A \quad \Leftrightarrow \quad f(n) \in K)$$

Note the implicit types here. The number $f(n)$ is going to be interpreted as the number of a TM.

Define the recursive function f which on input n computes *this particular* TM:

$M_{f(n)} =$	Erase input	Write n	M_i

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

$$\Leftrightarrow M_{f(n)}(f(n)) = 1 \quad \Leftrightarrow \quad f(n) \in K$$

♠

Get used to numbers being treated as machines! Lots of our standard languages are of the form $\{n : M_n$ is a TM such that$\ldots\}$.

Proposition 7.4 *Suppose that C is r.e.-complete and the following hold:*

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

then S is r.e.-complete.

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

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

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

Corollary 7.5 *$A_{0,17}$ is r.e.-complete.*
Every r.e.-complete set is r.e. and not recursive.

$$\text{HALT} \quad = \quad \{P(n,m) \mid \text{TM } M_n(m) \text{ eventually halts}\}$$

Proposition 7.6 HALT *is r.e.-complete.*

Proof: We have already seen that HALT is r.e. It thus suffices to show that $K \leq$ HALT.

We want to build a total, recursive f such that for all $w \in \mathbf{N}$,

$$w \in K \quad \Leftrightarrow \quad f(w) \in \text{HALT}$$

$$M_w(w) = 1 \quad \Leftrightarrow \quad M_{L(f(w))}(R(f(w))) \text{ halts}$$

That is, we want,

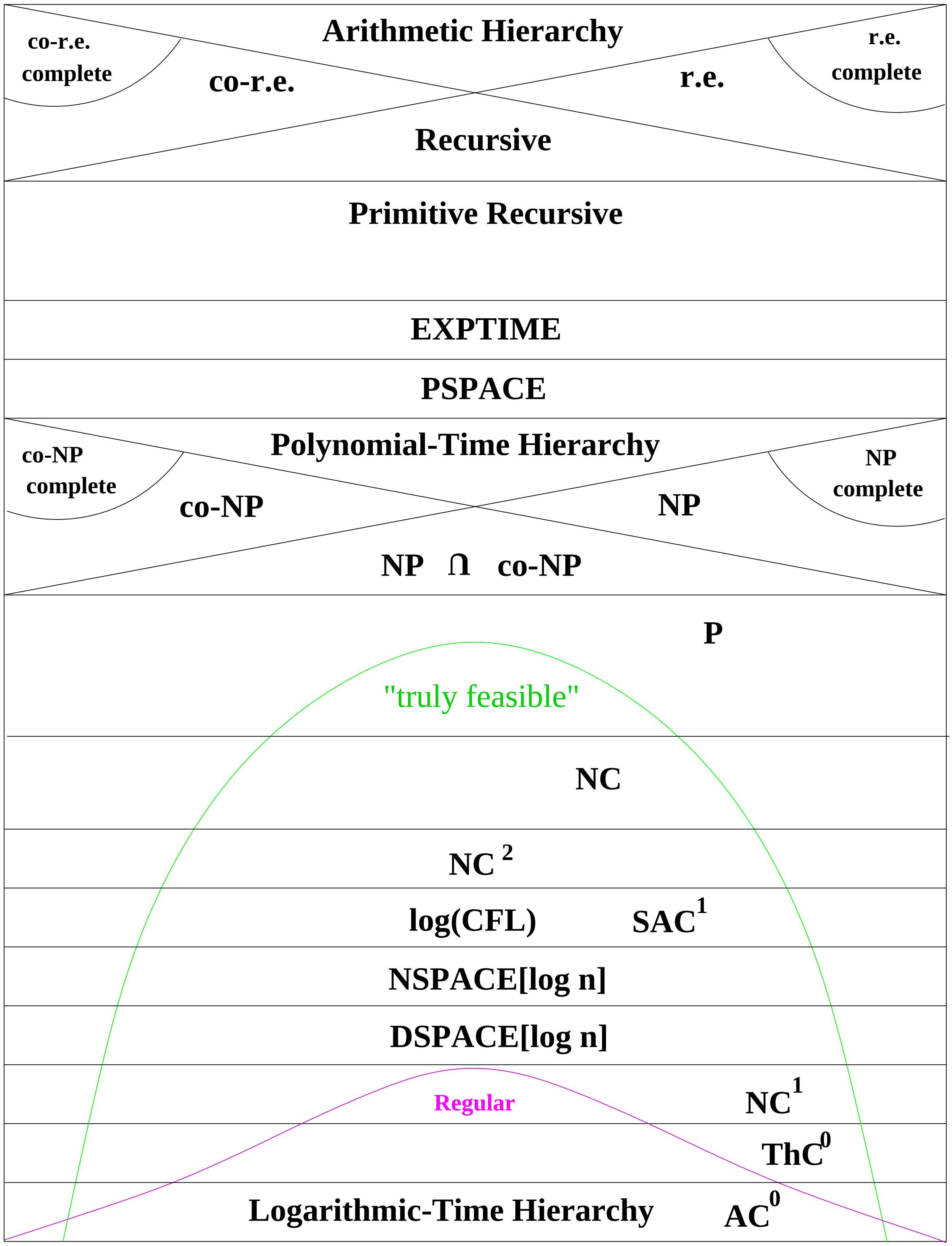
$$M_w(w) = 1 \quad \Leftrightarrow \quad M_\ell(r) \text{ halts}, \quad \text{where } f(w) = P(\ell, r)$$

Given w, let, $M_{\ell(w)} \ =$

Erase input	Write w	M_w	if 1 then halt else diverge

Letting $f(w) = P(\ell(w), 0)$, we have that

$$M_w(w) = 1 \quad \Leftrightarrow \quad M_{\ell(w)}(0) \text{ halts} \quad \Leftrightarrow \quad f(w) \in \text{HALT} \spadesuit$$


Arithmetic Hierarchy
co-r.e. complete
co-r.e.
r.e.
r.e. complete
Recursive
Primitive Recursive
EXPTIME
PSPACE
Polynomial-Time Hierarchy
co-NP complete
co-NP
NP
NP complete
NP ∩ co-NP
P
"truly feasible"
NC
NC 2
log(CFL)
SAC1
NSPACE[log n]
DSPACE[log n]
Regular
NC1
ThC0
Logarithmic-Time Hierarchy
AC0