Recall what decidable meant:

Algorithm $M$ - input $w$

- $M(w)$ accepts iff $w \in R$ \quad (R-relaton)
- $M(w)$ rejects iff $w \notin R$

So $M(w)$ halts on all inputs $w$.

Now allow $M$ may run forever and not produce an answer.

Defn let $R$ be a $k$-ary relation. Then $R$ is semidecidable if there is an algorithm $M$ s.t.

for all $w_1, \ldots, w_k \in \Sigma^*$,

- $M(w_1, \ldots, w_k)$ accepts iff $(w_1, \ldots, w_k) \in R$ (i.e. $R(w_1, \ldots, w_k)$)

For other $w_1, \ldots, w_k \notin R$, $M(w_1, \ldots, w_k)$ might run forever.
Example: $R = \exists \langle n,m \rangle : \exists i \leq n \text{ s.t. } i \text{ and } i+m \text{ are both prime}.$

$R \leq N^2.$

An algorithm that semi-decides is:

**Input:** $n,m$

**Algorithm**

For $i = n, n+1, n+2, \ldots$

If $i$ and $i+m$ are prime, accept (and halt)

End-for

Theorem: If $R$ is decidable, then $R$ is semidecidable.

**Pf:** Obvious.
Allow a machine algorithm to output successively arbitrary many things.

Called an "enumerator".

Outputs may be k-tuples \( v_1, v_2, v_3, \ldots \) for \((\Sigma^*)^k\).

Could be finitely many \( v_i \)'s, infinitely many, even zero.

Algorithms does not take any input.

**Definition.** Let \( R \) be a k-ary relation. \( R \) is computably enumerable iff there is an enumerator algorithm that for all \( v \in (\Sigma^*)^k \),

\[ v \in R \text{ iff } v \text{ is enumerated as one of the outputs } v_1, v_2, \ldots \]

of the enumerator.

We don't require any order of enumeration.
Theorem: $\mathcal{P}$ is computably enumerable if and only if $\mathcal{P}$ is semi-decidable.

Example $\{<n,m>: \exists i > n, i \text{ and } \sigma^i_m \text{ are prime}\}$

Algorithm to enumerate this relation:

For $k = 2, 3, 4, \ldots$
   For $l = 2, 3, 4, 5, \ldots$  \text{(so } 2 \leq k\text{)}
      If $k$ and $l$ are prime,
         For $n = 0, 1, 2, \ldots, l$
            Output $<n, k-l>$ (and keep running)
         End if
   End for
End for

OK to output same thing more than once.
Proof: Suppose $R$ is computably enumerable by algorithm $M$. Then $R$ is semidecided by:

Input $v \in (\Sigma^*)^k$

**Algorithm**
- Run $M$, watch what it outputs.
- If $v$ is ever output, accept (and halt).

Suppose $R$ is semidecidable by algorithm $N$.
Here is an algorithm to computably enumerate $R$:

**Algorithm:**
- Let $v_1, v_2, v_3, v_4, \ldots$ enumerate all members of $(\Sigma^*)^k$ in order of increasing length.
- $v_i$ - $i$-th member of $(\Sigma^*)^k$

For $j = 1, 2, 3, \ldots$

For $i = 1, 2, 3, \ldots$

Let $u_i$ be the $i$-th member of $(\Sigma^*)^k$.

Run $N(u_i)$ for up to $j$ steps.
- If $N(u_i)$ accepts in $\leq j$ steps, then output $u_i$.

**Proof:** Suppose $R$ is computably enumerable by algorithm $M$. Then $R$ is semidecided by:

Input $v \in (\Sigma^*)^k$

**Algorithm**
- Run $M$, watch what it outputs.
- If $v$ is ever output, accept (and halt).

Suppose $R$ is semidecidable by algorithm $N$.
Here is an algorithm to computably enumerate $R$:

**Algorithm:**
- Let $v_1, v_2, v_3, v_4, \ldots$ enumerate all members of $(\Sigma^*)^k$ in order of increasing length.
- $v_i$ - $i$-th member of $(\Sigma^*)^k$

For $j = 1, 2, 3, \ldots$

For $i = 1, 2, 3, \ldots$

Let $u_i$ be the $i$-th member of $(\Sigma^*)^k$.

Run $N(u_i)$ for up to $j$ steps.
- If $N(u_i)$ accepts in $\leq j$ steps, then output $u_i$.

"Dovetailing algorithm"
How to enumerate $\Sigma^*$: Let $\Sigma = \{a_1, a_2, \ldots, a_s\}$

- Length $0$, 1 way of length 1
- Length 1, 2 ways of length 2
- Length 2, 3 ways of length 3

To enumerate $(\Sigma^*)^k$: If $\langle w_1, w_k \rangle \in (\Sigma^*)^k$,

let $1 \langle w_1, w_k \rangle | = 0 \cdot | w_1 |

Enumerate member of $(\Sigma^*)^k$ in increasing order of

total length $\Sigma_i | w_i |

Thus $R$ is decidable iff $R$ and $\overline{R}$ are computably

enumerate.

Proof: Already showed: $R$ decidable $\Rightarrow$ $R$ is semidecidable
- $R$ decidable $\Rightarrow$ $\overline{R}$ is decidable

$R = (\Sigma^*)^k \setminus R$
Proof - continued

Suppose $R$ is computably enumerable by algorithm $M$.

Here is an algorithm that decides $R$:

**Input** $v \in (\Sigma^*)^k$

**Algorithm**

Run $M$ and $N$ in parallel.

If $M$ outputs $\omega$, accept.
If $N$ outputs $\omega$, reject.

One of these two possibilities eventually happens. So the algorithm always halts.

**Terminology** computable enumerable $v$ also called "c.e."

A relation is often called a "set".
A unary relation is often called a "language".
A language $L$ is a subset of $\Sigma^*$.

"M semidecides $R$", often instead, say
"M recognizes $R$" or
"M accepts $R$".

**Definition:** A partial computable function $f$ is a function with domain a subset of $(\Sigma^*)^k$ and range $\Sigma^*$, and such there is an algorithm $M$ that on input $v \in (\Sigma^*)^k$:

- $M(v) = w$ if $f(v) = w$.
- $M(v)$ never halts if $f(v)$ is undefined.

Notation:
- $f(v) \downarrow$, $f(v)$ converges if $v \in \text{domain}(f)$
- $f(v) \uparrow$, $f(v)$ diverges if $v \notin \text{dom}(f)$
\[ f(n, m) = \begin{cases} \text{the least } i, n, \text{ s.t. } i \text{ and } i + m \text{ are prime} \\ \text{undefined if no such } i \text{ exists} \end{cases} \]

\[ f(n, m) = \mu(i) \left[ \begin{array}{c} i \geq n \text{ and } i \text{ and } i + m \text{ are prime} \\ \text{the least } i \text{ such that } \ldots \end{array} \right] \]