The trick of switching $\mathsf{P}$ and $\mathsf{PSPACE}$ does not affect running time or space, so it also shows that $\mathsf{Exp. PSPACE, P, L}$ are closed under $\sim$.

- Does not work for NTMs: Same as with an NFA.

$$L = \{ x : x \text{ ends in } 0 \} \quad \text{but swapping}$$

$$\mathsf{acc} \quad \hat{L} = \{ x : x = 3 \text{ or } x \text{ ends in } 1 \} \quad \text{do } \hat{L}.$$  

So does not show that $\mathsf{NP} = \mathsf{co-NP}$. Also does not show $\mathsf{NL} = \mathsf{co-NL}$, but this is a famous theorem (1979).

- Does not work if $M$ does not halt for all inputs.

**Proof:** Take DTMs $M_1$ and $M_2$ such that $L(M_1) = A$ and $L(M_2) = \hat{A}$. Build $M_3$ via flowcharts as follows.

But we can show

**Theorem:** if $A$ and $\hat{A}$ are both c.e., then $A$ is decidable.

\[ \text{Input } X \]

\[ \text{Did } M_1 \text{ accept } X \text{ in that step?} \]

\[ \text{Did } M_1 \text{ accept } X \text{ in that step?} \]

\[ \text{Execute one move step of the computation } M_1 \text{ (X)} \]

\[ \text{Execute one more step of } M_2 \text{ (X), if possible) \]

\[ \text{Yes} \]

\[ \text{No} \]

\[ \text{RE accept } \quad \text{ or } \text{ RE = REC.} \]
Theorem: We can define a language $D \in \mathcal{L} \setminus \mathcal{L}_{\mathcal{R}} \setminus \mathcal{L}_{\mathcal{R}}$ so those classes are different and neither equals $\mathcal{L}_{\mathcal{R}}$.

Proof: Recall every DTM $M$ has a unique string code $\langle M \rangle$.

Define $D = \{ \langle M \rangle : M$ does not accept $\langle M \rangle \}$. 

Suppose $D$ were c.e. Then there would be a DTM $Q$ s.t. $L(Q) = D$. $Q$ accepts $\langle Q \rangle$ $\iff$ $\langle Q \rangle \notin D$ by $L(Q) = D$.

But then $\langle Q \rangle \in D$ $\iff$ $\langle Q \rangle \notin D$ by def of $D$.

$\vdash \langle Q \rangle \in D \iff \langle Q \rangle \notin D$.

A logical statemt can never be equivalent to its negation, else the Universe explodes. So $D$ cannot be c.e.

Analogy: Any 1-1 function $f : A \to P(A)$ cannot be onto $P(A)$. $D_f = \{ x : x \notin \text{the set } f(x) \}$.

If $D$ is in the range of $f$, then it would equal $f(q)$ for some $q \in A$. But then $q \in D$ $\iff$ $q$ is in the set $f(q)$ by def of $D$.

$q \in D$ $\iff$ $q$ is not in the set $f(q)$, by def of $D$. Same contradiction. Hence $P(\mathcal{Z})$ is uncountable.
But $D$ is co-c.e. The complement of $D$ is (essentially)

$$K_{TM} = \{ \langle M \rangle : \text{M does accept } \langle M \rangle \}.$$ 

$K_{TM}$ is (e.g.: it is a "central slice" of the Asm language

$$A_{TM} = \{ \langle M, x \rangle : \text{M accepts } x \},$$

and the UTM $M_U$ s.t. $L(M_U) = A_{TM}$ can be modified to accept $K_{TM}$ with an initial check that $x = \langle M \rangle$.

Loose end: what if a given input $x$ is not the code of any TM $M$? Several tech. approaches:

- Consider such $x$ to be a code for $M_0 = \emptyset$.
- Use Gödel numbers:
  $$x = \text{number } i \xrightarrow{f} \text{machine } M_i.$$  

**Historical Note:** The latter gives us a computable enumeration $M_1, M_2, M_3, \ldots, M_i, \ldots$ of machines. We could build a "TM compiler" that would not only test whether a TM code is valid, it could list out all the valid codes ad-infinitem. Any Turing machine can be considered to generate such a list by making it try to generate all TM's and accepting computation. Then it enumerates its language, hence the term "r.e."