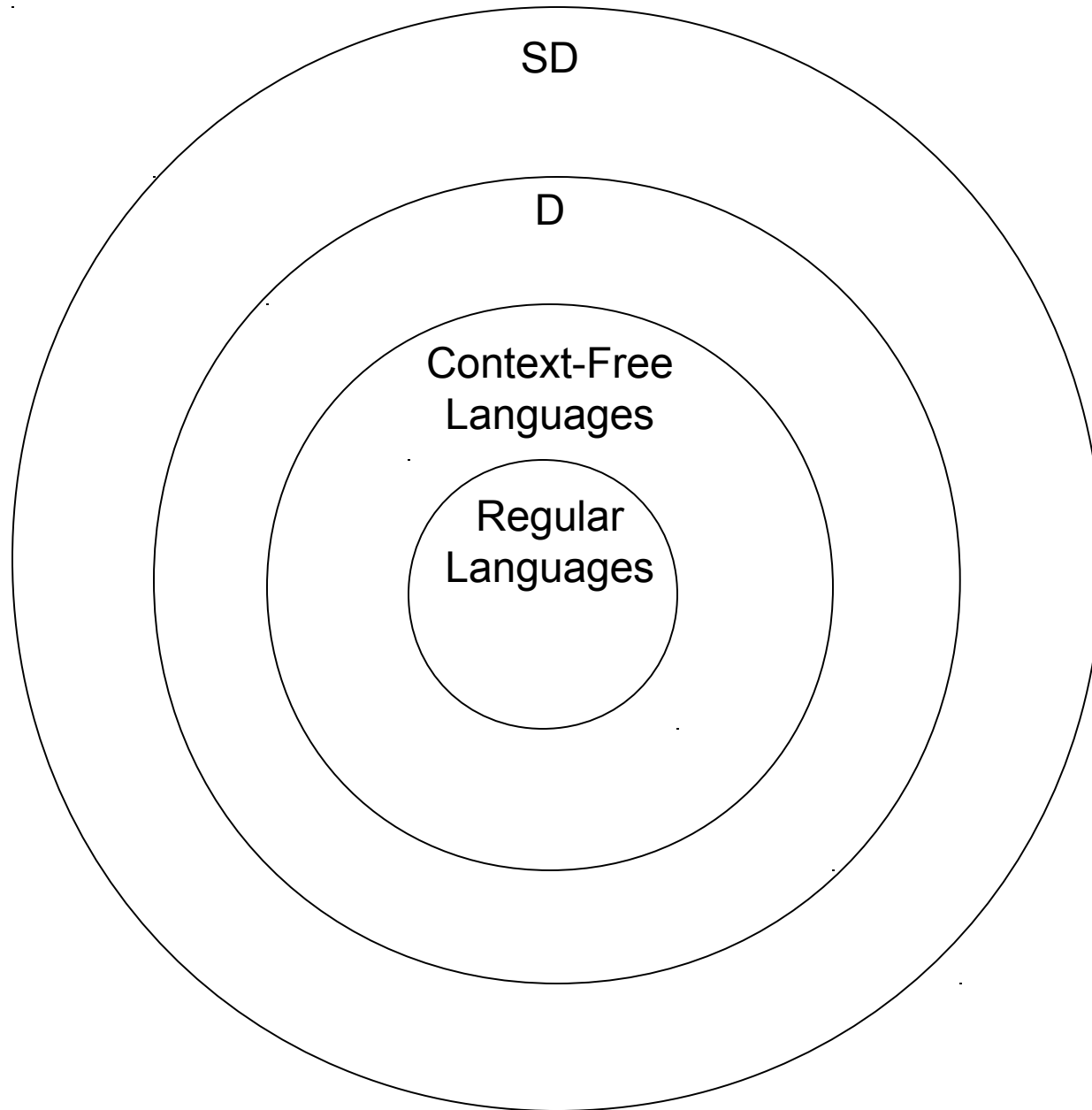




# **Decidable and Semidecidable Languages**

Chapter 20

# D and SD Languages



# Every CF Language is in D

**Theorem:** The set of context-free languages is a *proper* subset of D.

**Proof:**

- Every context-free language is decidable, so the context-free languages are a subset of D.
- There is at least one language,  $A^nB^nC^n$ , that is decidable but not context-free.

So the context-free languages are a *proper* subset of D.



# Decidable and Semidecidable Languages

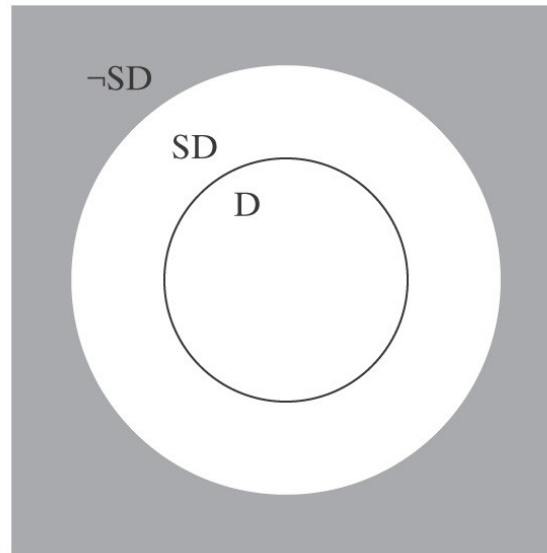
Almost every obvious language that is in SD is also in D:

- $A^n B^n C^n = \{a^n b^n c^n, n \geq 0\}$
- $\{w_c w, w \in \{a, b\}^*\}$
- $\{ww, w \in \{a, b\}^*\}$
- $\{w = x \star y = z : x, y, z \in \{0, 1\}^* \text{ and, when } x, y, \text{ and } z \text{ are viewed as binary numbers, } xy = z\}$

But there are languages that are in SD but not in D:

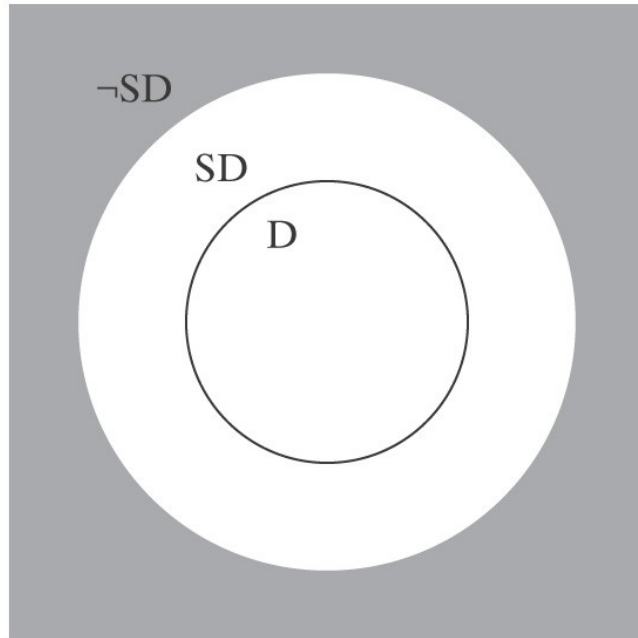
- $H = \{ \langle M, w \rangle : M \text{ halts on input } w \}$

# D and SD



1. D is a subset of SD. In other words, every decidable language is also semidecidable.
2. There exists at least one language that is in SD/D, the donut in the picture.
3. There exist languages that are not in SD. In other words, the gray area of the figure is not empty.

# Subset Relationships between D and SD



✓ 1. There exists at least one SD language that is not D.

2. Every language that is in D is also in SD: If  $L$  is in D, then there is a Turing machine  $M$  that decides it (by definition).

But  $M$  also semidecides it.

# Languages That Are Not in SD

**Theorem:** 3. There are languages that are not in SD.

**Proof:** Assume any nonempty alphabet  $\Sigma$ .

- There is a countably infinite number of SD languages over  $\Sigma$ .
- There is an uncountably infinite number of languages over  $\Sigma$ .

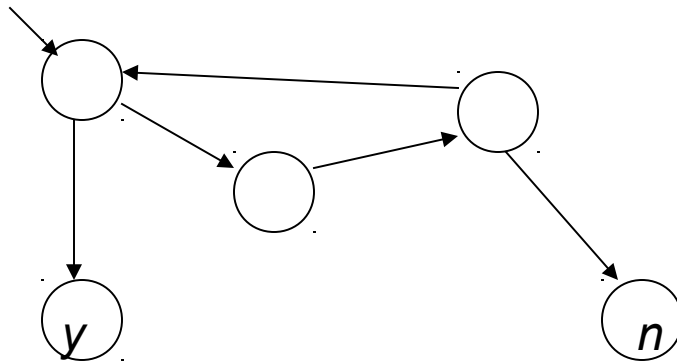
So there are more languages than there are languages in SD.  
Thus there must exist at least one language that is in  $\neg$ SD.

# Closure of D Under Complement

**Theorem:** The set  $D$  is closed under complement.

**Proof:** (by construction) If  $L$  is in  $D$ , then there is a deterministic Turing machine  $M$  that decides it.

$M$ :



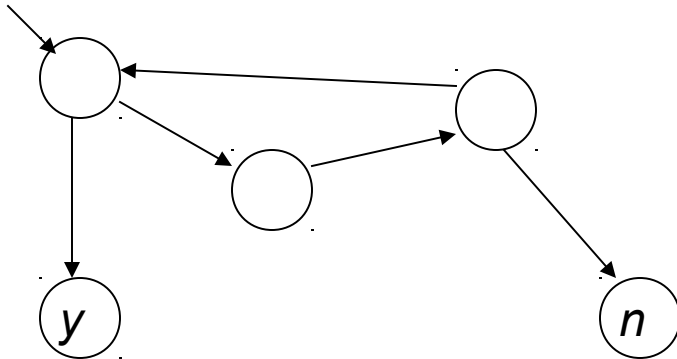
From  $M$ , we construct  $M'$  to decide  $\neg L$ :



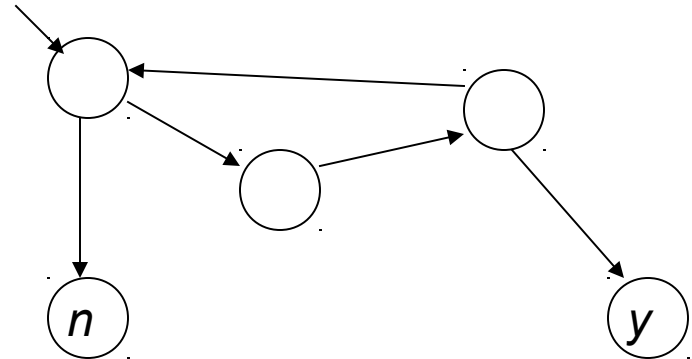
# Closure of D Under Complement

**Proof:** (by construction)

$M$ :



$M'$ :



This works because, by definition,  $M$  is:

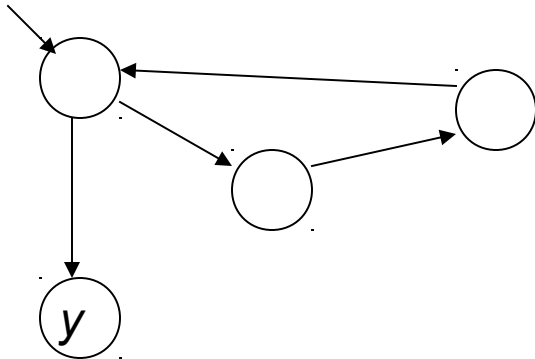
- deterministic
- complete

Since  $M'$  decides  $\neg L$ ,  $\neg L$  is in D.

# SD is Not Closed Under Complement

Can we use the same technique?

$M$ :



$M'$ :

# SD is Not Closed Under Complement

Suppose we had:

$M_L$ :

$M_{\neg L}$ :

Then we could decide  $L$ . How?

So every language in SD would also be in D.

But we know that there is at least one language ( $H$ ) that is in SD but not in D. Contradiction.

# D and SD Languages

**Theorem:** A language is in D iff both the language and its complement are in SD.

**Proof:**

- $L$  in D implies  $L$  and  $\neg L$  are in SD:
  - $L$  is in SD because  $D \subset SD$ .
  - D is closed under complement
  - So  $\neg L$  is also in D and thus in SD.
- $L$  and  $\neg L$  are in SD implies  $L$  is in D:
  - $M_1$  semidecides  $L$ .
  - $M_2$  semidecides  $\neg L$ .
  - To decide  $L$ :
    - Run  $M_1$  and  $M_2$  in parallel on  $w$ .
    - Exactly one of them will eventually accept.



# A Language that is Not in SD

**Theorem:** The language  $\neg H =$

$\{ \langle M, w \rangle : \text{TM } M \text{ does not halt on input string } w \}$

is not in SD.

**Proof:**

- $H$  is in SD.
- If  $\neg H$  were also in SD then  $H$  would be in D.
- But  $H$  is not in D.
- So  $\neg H$  is not in SD.

# Enumeration

Enumerate means list. We look at Turing Machines as generators.

We say that Turing machine  $M$  **enumerates** the language  $L$  iff, for some fixed, non-halting, state  $p$  of  $M$ :

$$L = \{w : (s, \varepsilon) \vdash_M^* (p, w)\}.$$

Whenever the machine enters  $p$ , the string on the tape is enumerated.

If  $L$  is finite, then  $M$  eventually halts.

A language is **Turing-enumerable** iff there is a Turing machine that enumerates it.

# Example of Enumeration

Consider a **printing** subroutine:  $P$  be a Turing machine that enters state  $p$  and then halts:

Let  $L = a^*$ .  $M_1$  and  $M_2$  both enumerate  $L$ :

$M_1$ :

$\downarrow$   
 $>PaR$

$M_2$ :

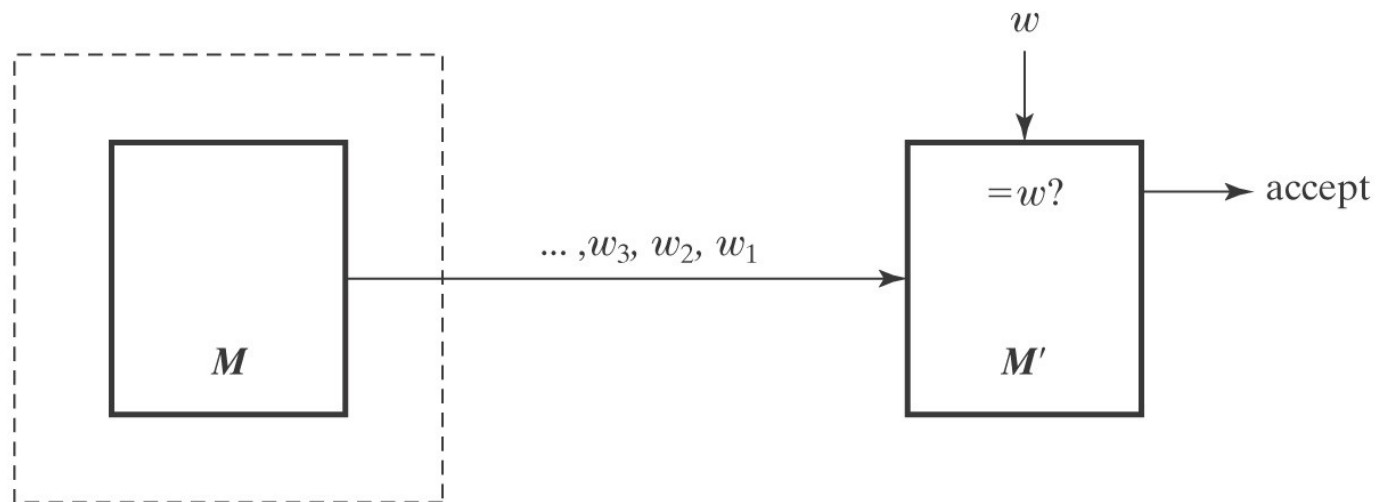
$\downarrow$   
 $>PaP\Box RaRaRaP\Box P$

# SD and Turing Enumerable

**Theorem:** A language is in SD iff it is Turing-enumerable.

**Proof that Turing-enumerable implies SD:** Let  $M$  be the Turing machine that enumerates  $L$ . We convert  $M$  to a machine  $M'$  that semidecides  $L$ :

1. Save input  $w$ .
2. Begin enumerating  $L$ . Each time an element of  $L$  is enumerated, compare it to  $w$ . If they match, accept.





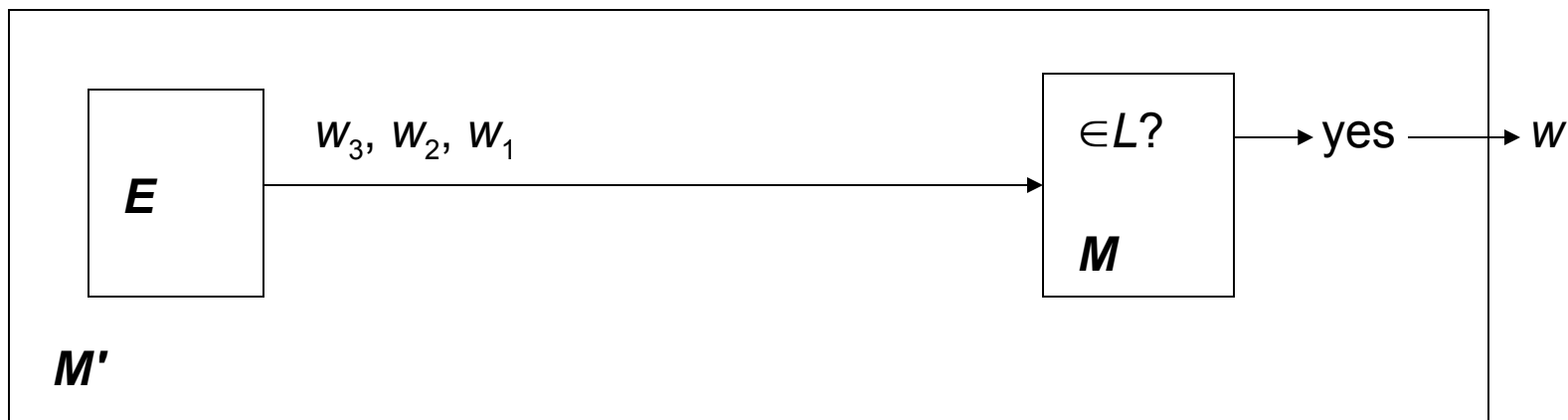
# The Other Way

***Proof that SD implies Turing-enumerable:***

If  $L \subseteq \Sigma^*$  is in SD, then there is a Turing machine  $M$  that semidecides  $L$ .

A procedure  $E$  to enumerate all elements of  $L$ :

1. Enumerate all  $w \in \Sigma^*$  lexicographically.  
e.g.,  $\epsilon$ ,  $a$ ,  $b$ ,  $aa$ ,  $ab$ ,  $ba$ ,  $bb$ , ...
2. As each is enumerated, use  $M$  to check it.



Does this work?

# Dovetailing

$\varepsilon$  [1]

$\varepsilon$  [2]

$\varepsilon$  [3]

$\varepsilon$  [4]

$\varepsilon$  [5]

$\varepsilon$  [6]

a [1]

a [2]

a [3]

a [4]

a [5]

b [1]

b [2]

b [3]

aa [1]

aa [2]

aa [3]

ab [1]

ab [2]

ba [1]

# The Other Way

## ***Proof that SD implies Turing-enumerable:***

If  $L \subseteq \Sigma^*$  is in SD, then there is a Turing machine  $M$  that semidecides  $L$ .

A procedure to enumerate all elements of  $L$ :

1. Enumerate all  $w \in \Sigma^*$  lexicographically.
2. As each string  $w_i$  is enumerated:
  1. Start up a copy of  $M$  with  $w_i$  as its input.
  2. Execute one step of each  $M_i$  initiated so far, excluding only those that have previously halted.
3. Whenever an  $M_i$  accepts, output  $w_i$ .



# Lexicographic Enumeration

$M$  **lexicographically enumerates**  $L$  iff  $M$  enumerates the elements of  $L$  in lexicographic order.

A language  $L$  is **lexicographically Turing-enumerable** iff there is a Turing machine that lexicographically enumerates it.

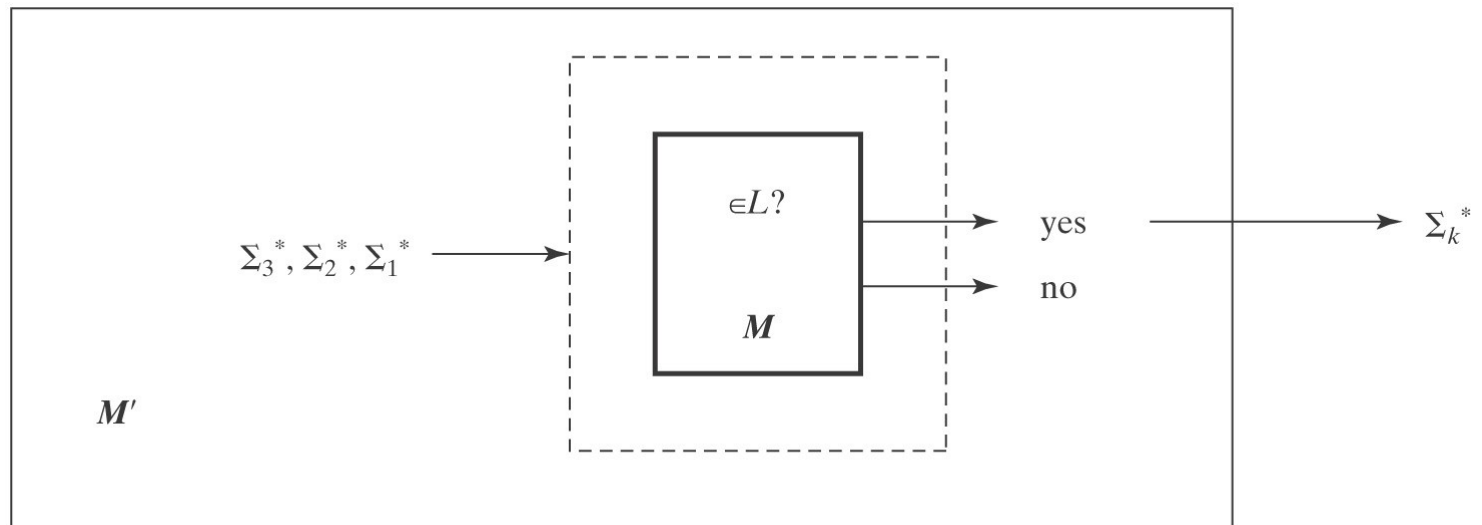
Example:  $A^n B^n C^n = \{a^n b^n c^n : n \geq 0\}$

Lexicographic enumeration:

# Lexicographically Enumerable = D

**Theorem:** A language is in D iff it is lexicographically Turing-enumerable.

**Proof that D implies lexicographically TE:** Let  $M$  be a Turing machine that decides  $L$ . Then  $M'$  lexicographically generates the strings in  $\Sigma^*$  and tests each using  $M$ . It outputs those that are accepted by  $M$ . Thus  $M'$  lexicographically enumerates  $L$ .

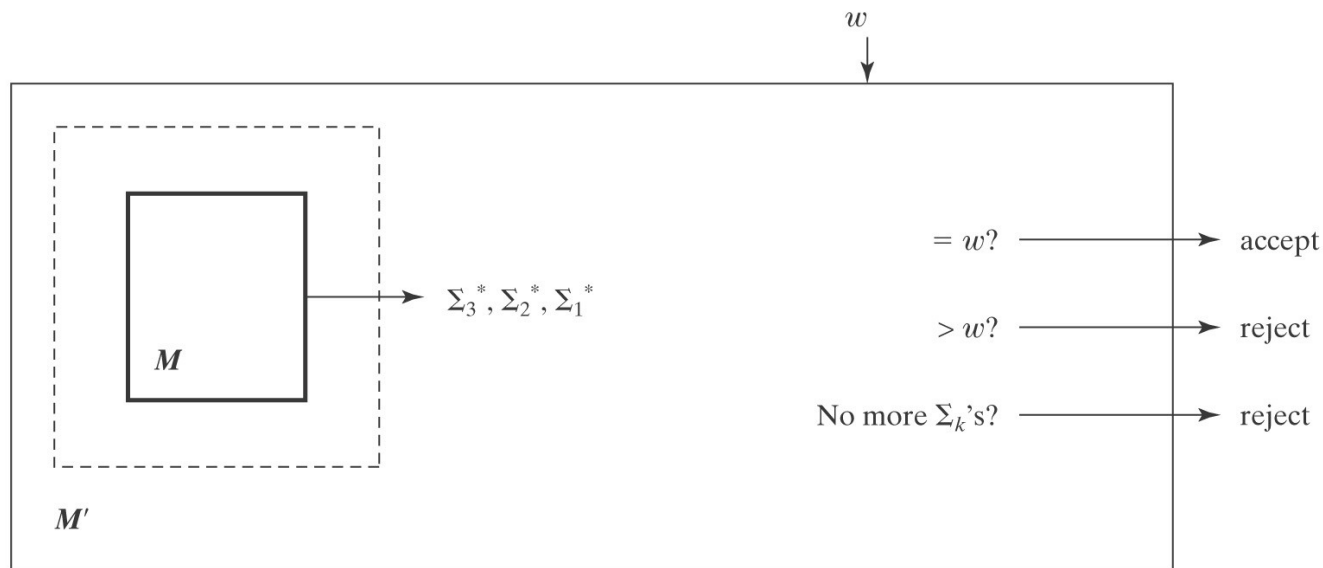


# Proof, Continued

***Proof that lexicographically TE implies D:*** Let  $M$  be a Turing machine that lexicographically enumerates  $L$ . Then, on input  $w$ ,  $M'$  starts up  $M$  and waits until:

- $M$  generates  $w$  (so  $M'$  accepts),
- $M$  generates a string that comes after  $w$  (so  $M'$  rejects), or
- $M$  halts (so  $M'$  rejects).

Thus  $M'$  decides  $L$ .



# Language Summary

**IN**

Semideciding TM  
Enumerable  
Unrestricted grammar

Deciding TM  
Lexicographic enum.  
 $L$  and  $\neg L$  in SD

CF grammar  
PDA  
Closure

Regular Expression  
FSM

**SD**

$H$

**D**

$A^n B^n C^n$

**Context-Free**

$A^n B^n$

**Regular**

$a^* b^*$

**OUT**

Reduction

Diagonalize  
Reduction

Pumping  
Closure

Pumping  
Closure

