Lecture 18

6.2.3 Language classes with respect to time and space complexity 187

Language hierarchy based on the complexity of accepting the language

Language hierarchy based on the complexity of accepting the language

i. ***Infinity***

- language hierarchy is infinite

Language hierarchy based on the complexity of accepting the language

i. ***Infinity***

- language hierarchy is infinite

ii. ***Continuity*** **for fully space-constructible functions**

- language hierarchy is continuous for fully space-constructible functions

Language hierarchy based on the complexity of accepting the language

i. ***Infinity***

- language hierarchy is infinite

ii. ***Continuity* for fully space-constructible functions**

- language hierarchy is continuous for fully space-constructible functions

iii. ***Continuity* for fully time-constructible functions**

- language hierarchy is continuous for fully time-constructible functions

Language hierarchy based on the complexity of accepting the language

i. ***Infinity***

- language hierarchy is infinite

ii. ***Continuity*** **for fully space-constructible functions**

- language hierarchy is continuous for fully space-constructible functions

iii. ***Continuity*** **for fully time-constructible functions**

- language hierarchy is continuous for fully time-constructible functions

iv. ***Gaps*** **in hierarchy**

- language hierarchy *is not* continuous for the general case of functions which are not space and time constructible

Language hierarchy based on the complexity of accepting the language

i. ***Infinity***

- language hierarchy is infinite

ii. ***Continuity* for fully space-constructible functions**

- language hierarchy is continuous for fully space-constructible functions

iii. ***Continuity* for fully time-constructible functions**

- language hierarchy is continuous for fully time-constructible functions

iv. ***Gaps* in hierarchy**

- language hierarchy *is not* continuous for the general case of functions which are not space and time constructible

v. ***Optimal* TM**

- there is a language for which there is no optimal TM which accepts it in minimal time or minimal space

Language hierarchy based on the complexity of accepting the language

i. ***Infinity***

- language hierarchy is infinite

ii. ***Continuity*** **for fully space-constructible functions**

- language hierarchy is continuous for fully space-constructible functions

iii. ***Continuity*** **for fully time-constructible functions**

- language hierarchy is continuous for fully time-constructible functions

iv. ***Gaps*** **in hierarchy**

- language hierarchy *is not* continuous for the general case of functions which are not space and time constructible

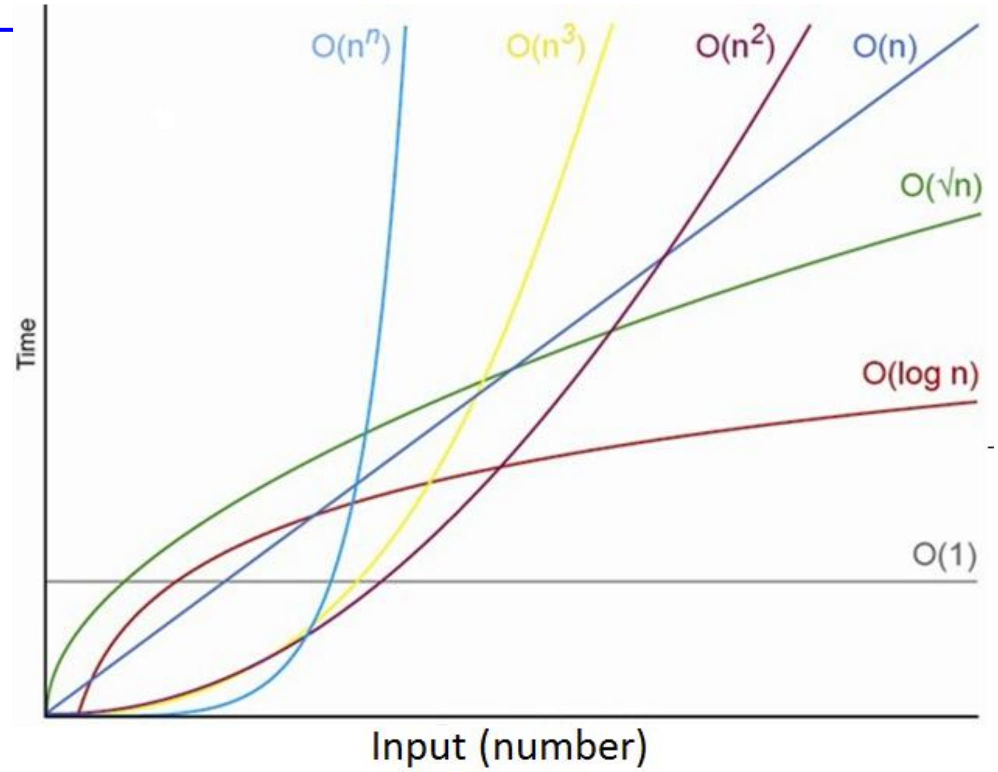
v. ***Optimal*** **TM**

- there is a language for which there is no optimal TM which accepts it in minimal time or minimal space

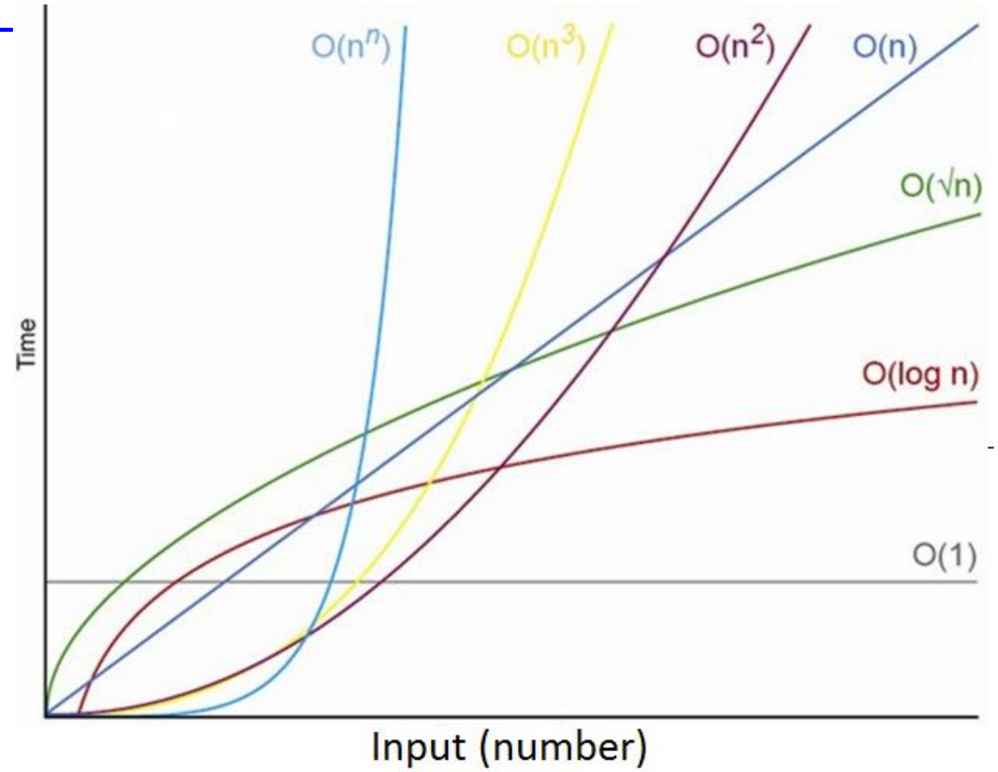
vi. ***Union*** **of language classes**

- there is a complexity function which covers all languages from the union

Infinity

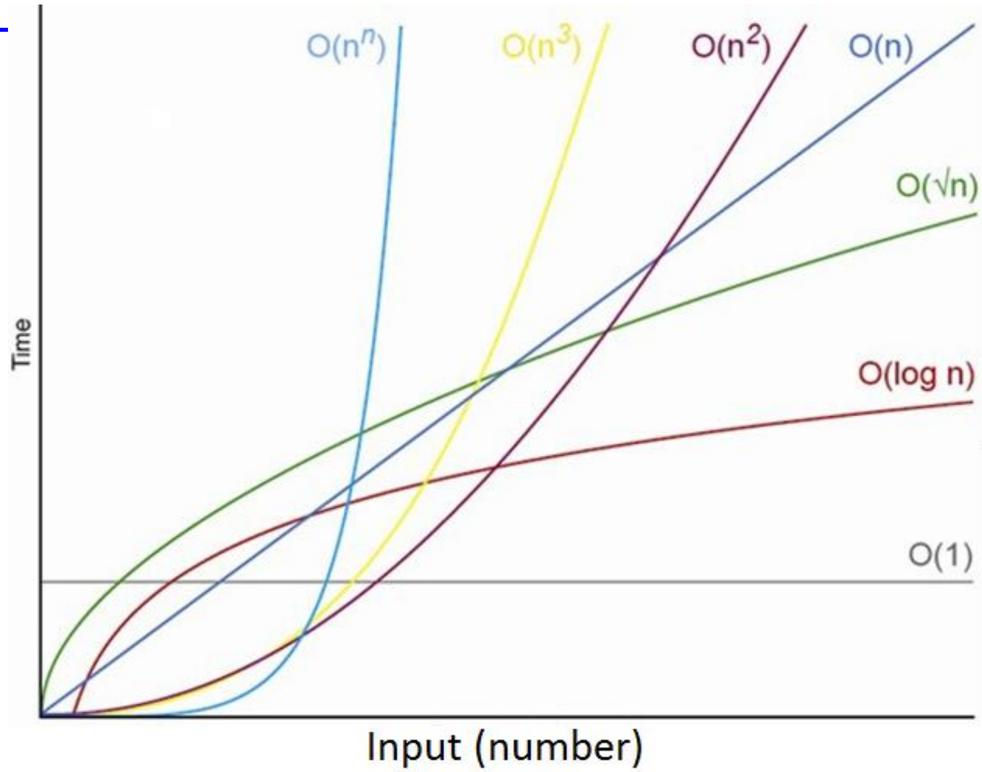


Infinity



DTIME ($f_1(n)$)

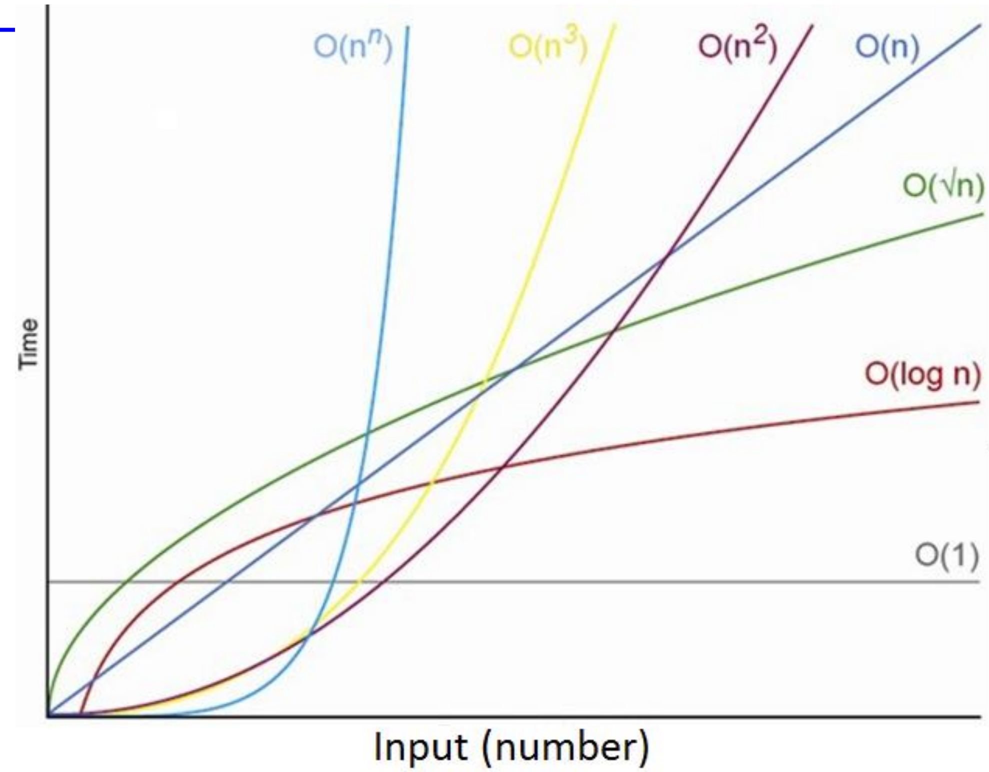
Infinity



$L_1 \notin \mathrm{DTIME}(f_1(n))$

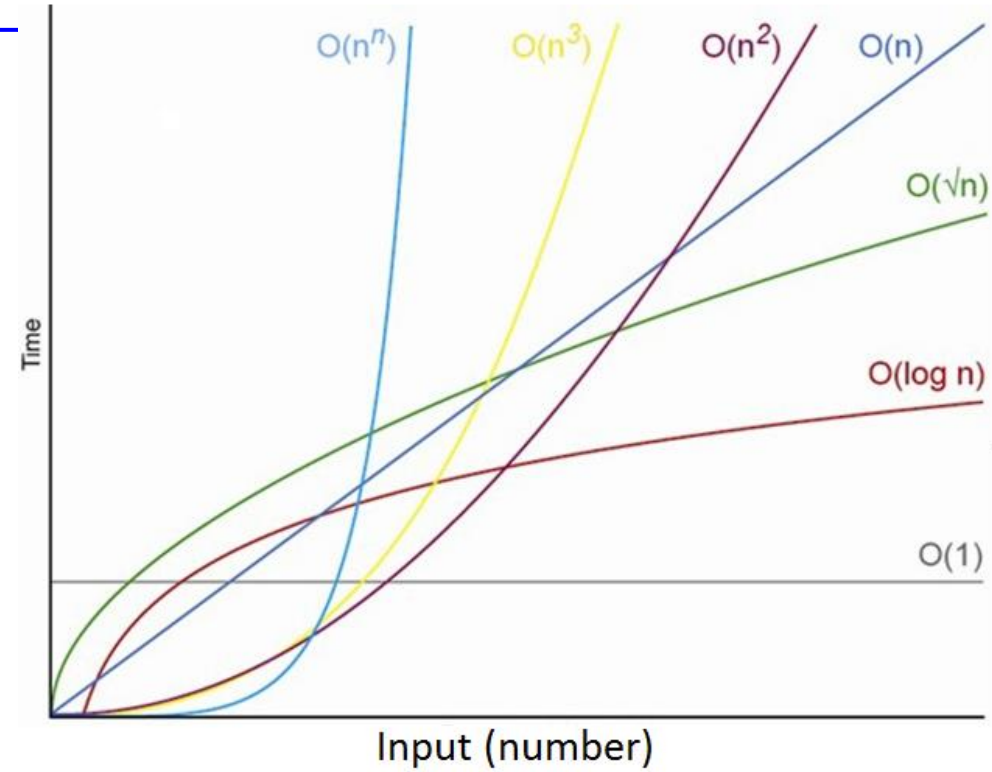
DTIME ($f_1(n)$)

Infinity

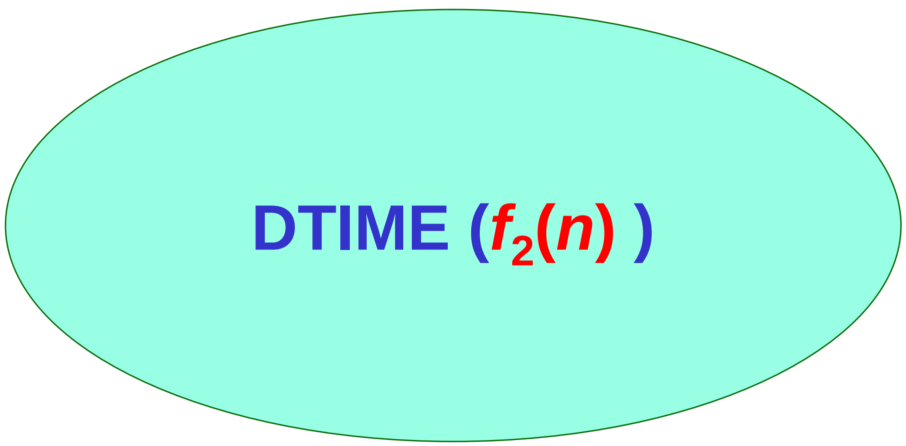


DTIME ($f_2(n)$)

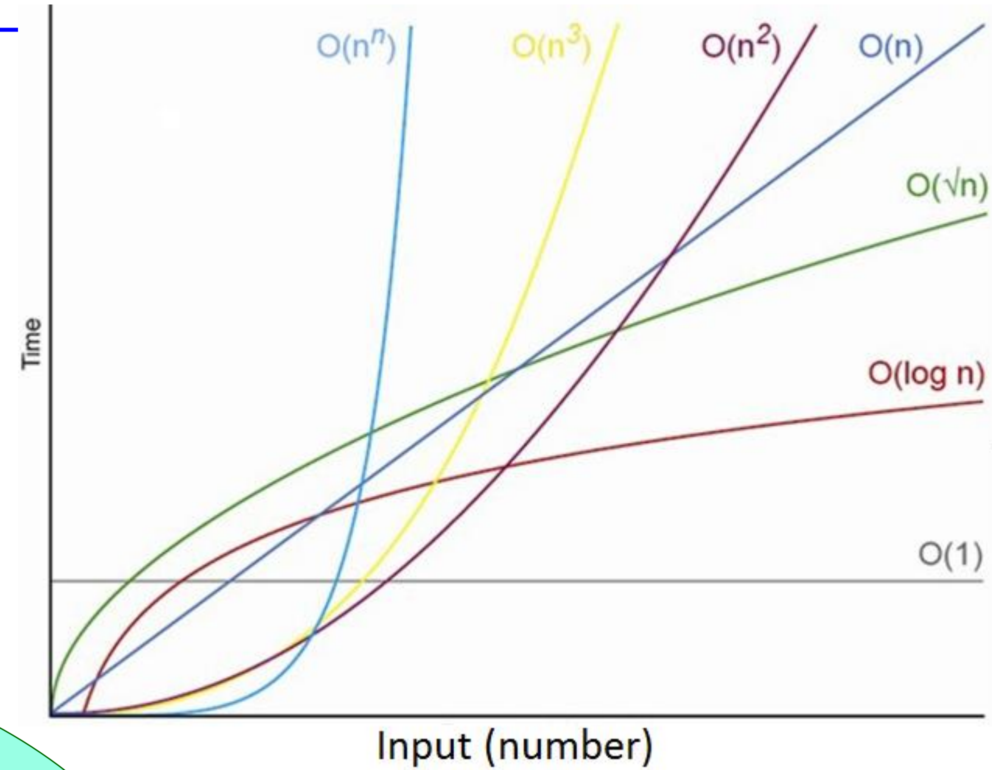
Infinity



$L_2 \notin \mathrm{DTIME}(f_2(n))$

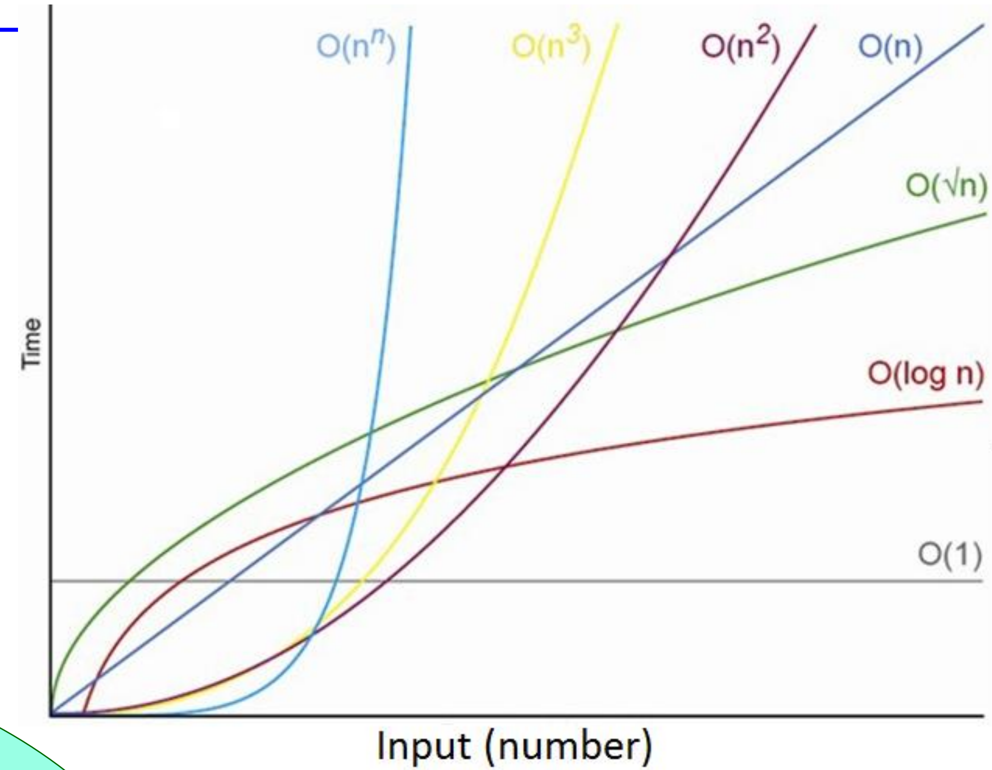


Infinity

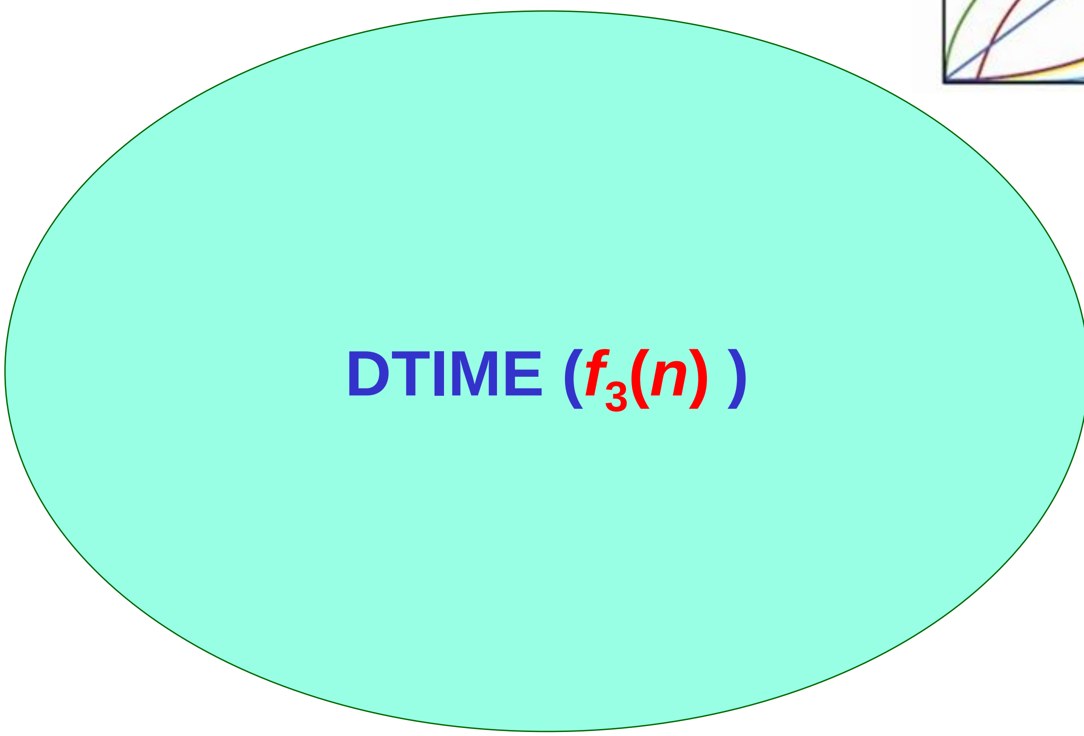


DTIME ($f_3(n)$)

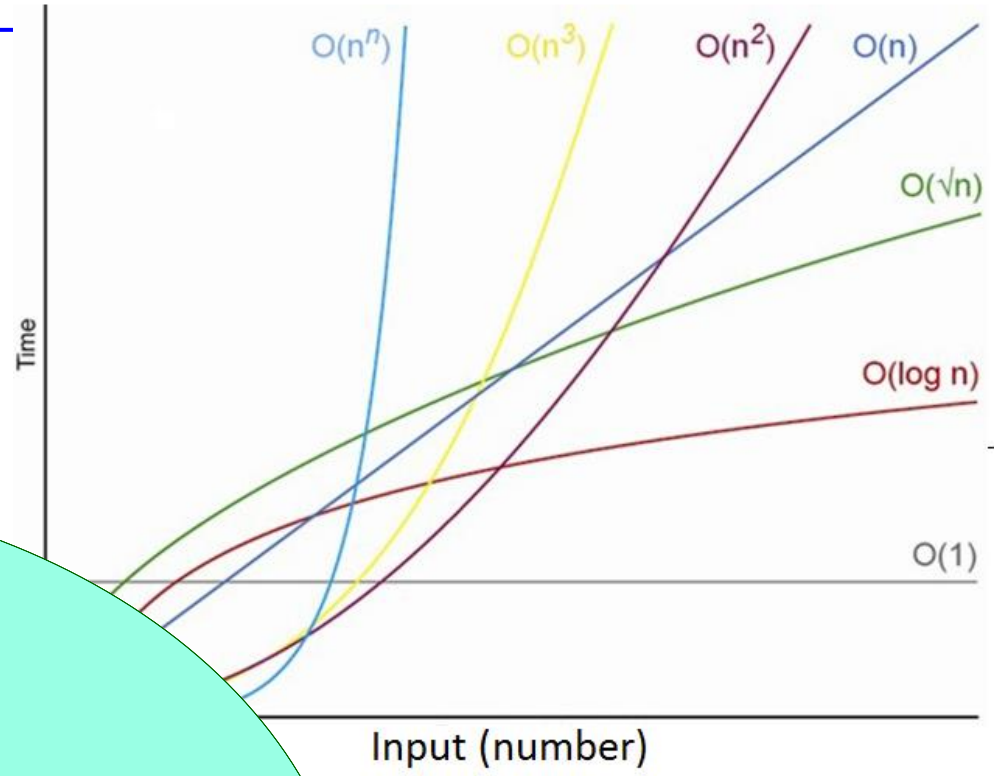
Infinity



$L_3 \notin \mathrm{DTIME}(f_3(n))$

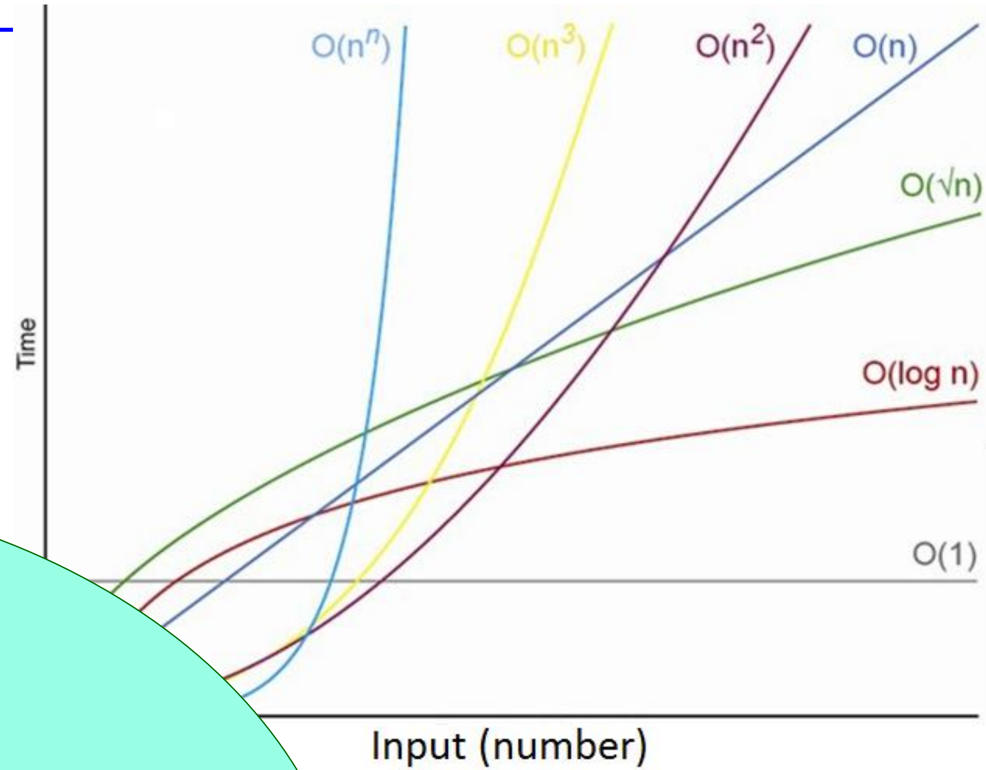


Infinity



DTIME ($f_4(n)$)

Infinity



$L_4 \notin \text{DTIME}(f_4(n))$

DTIME ($f_4(n)$)

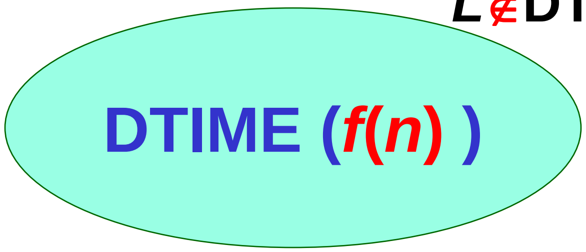
Infinity

$L \notin \text{DTIME}(\ f(n)\ )$

DTIME ($f(n)$)

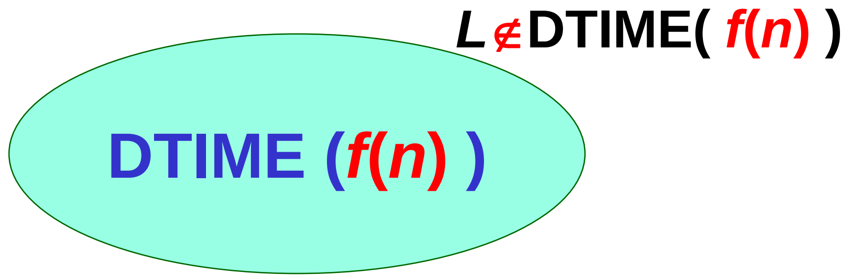
Infinity

$L \notin$ DTIME($f(n)$)



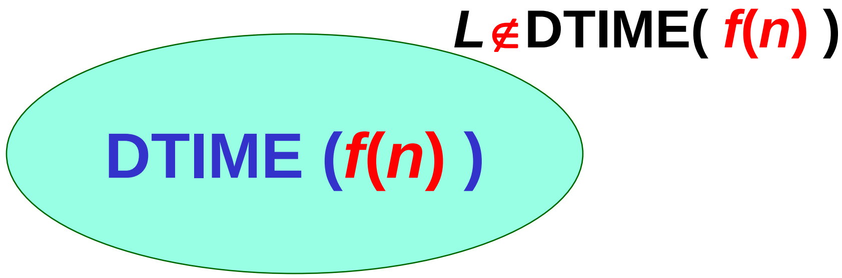
- If

Infinity



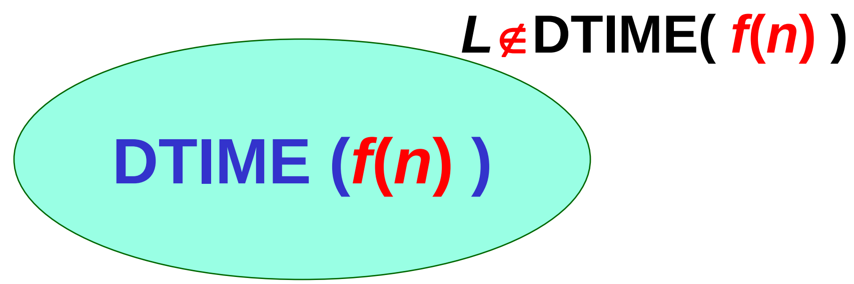
- **If**
  - **$f(n)$ is a total recursive function**

Infinity



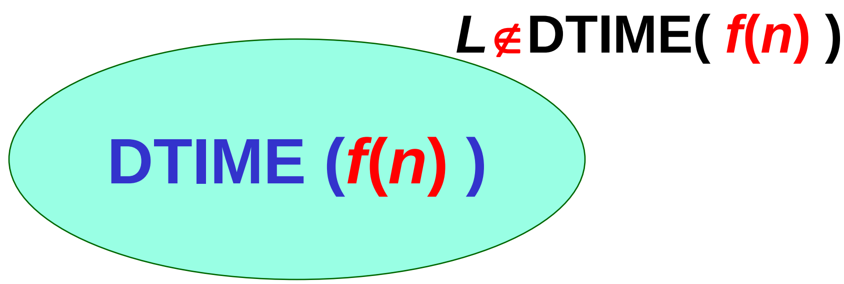
- **If**
  - **$f(n)$ is a total recursive function**
- **Then there is a language L**

Infinity



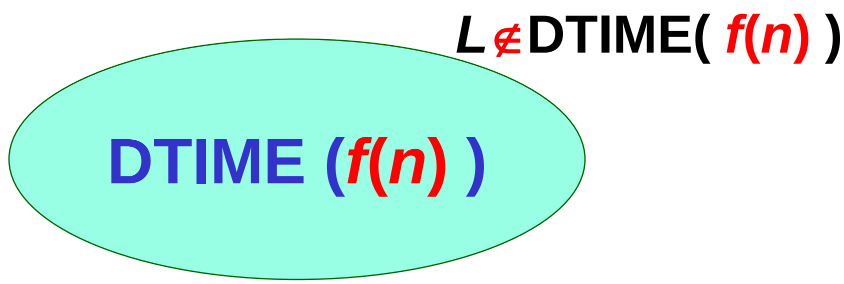
- **If**
  - **$f(n)$ is a total recursive function**
- **Then there is a language L**
  - **$L \notin \text{DTIME}(f(n))$**

Infinity



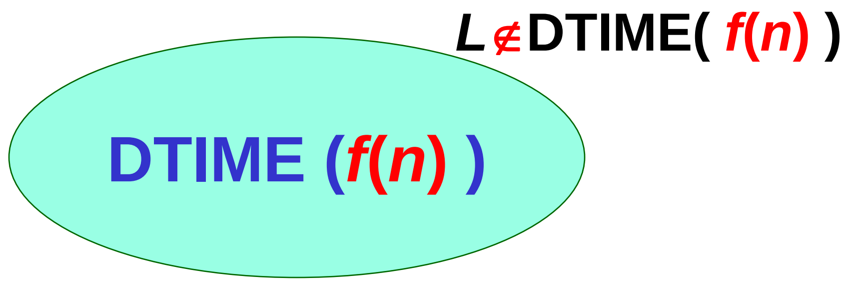
- **If**
  - **$f(n)$ is a total recursive function**
- **Then there is a language L**
  - **$L \notin \mathrm{DTIME}(f(n))$**
  - **$L \notin \mathrm{NTIME}(f(n))$**

Infinity



- If
  - $f(n)$ is a total recursive function
- Then there is a language L
  - $L \notin \text{DTIME}(f(n))$
  - $L \notin \text{NTIME}(f(n))$
  - $L \notin \text{DSPACE}(f(n))$

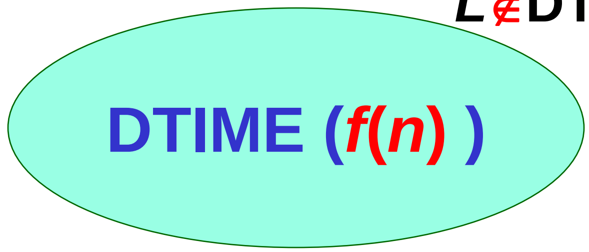
Infinity



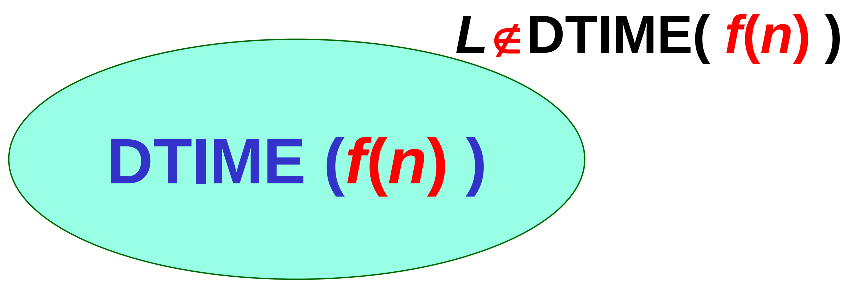
- If
  - $f(n)$ is a total recursive function
- Then there is a language L
  - $L \notin \text{DTIME}(f(n))$
  - $L \notin \text{NTIME}(f(n))$
  - $L \notin \text{DSPACE}(f(n))$
  - $L \notin \text{NSPACE}(f(n))$

Infinity

$L \notin \mathrm{DTIME}(\ f(n)\ )$

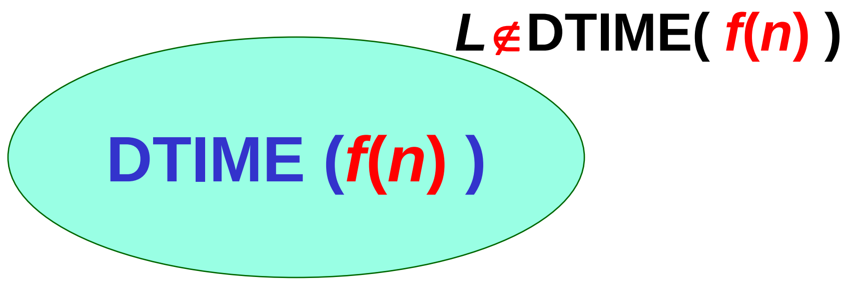


Infinity



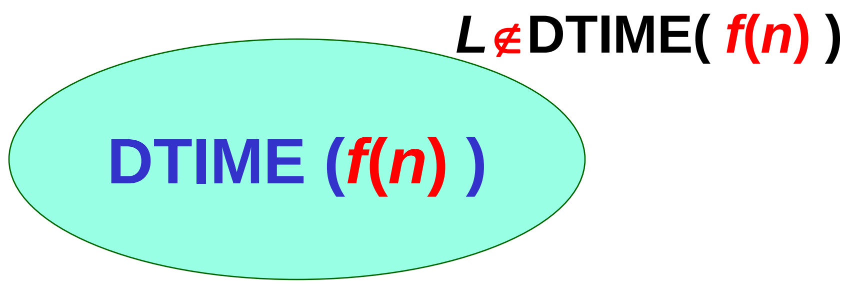
- **Encoding of TM M**
  - **tape symbols: $\{0, 1, B, X_4, X_5, \ldots X_m\}$**
  - **symbol X_k is encoded by the string of zeros 0^k**

Infinity



- Encoding of TM M
  - tape symbols: $\{0, 1, B, X_4, X_5, \ldots X_m\}$
  - symbol X_k is encoded by the string of zeros 0^k
- String w_i
  - string w_i is the i-th string in a canonical sequence of all strings

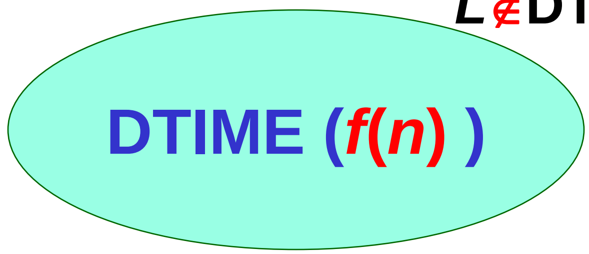
Infinity



- **Encoding of TM *M***
  - **tape symbols: $\{0, 1, B, X_4, X_5, \ldots X_m\}$**
  - **symbol X_k is encoded by the string of zeros 0^k**
- **String w_i**
  - **string w_i is the i-th string in a canonical sequence of all strings**
- **TM M_i**
  - **index value i in TM M_i equals to the integer value of its binary encoding**

Infinity

$L \notin \mathrm{DTIME}(\, f(n)\,)$



Infinity

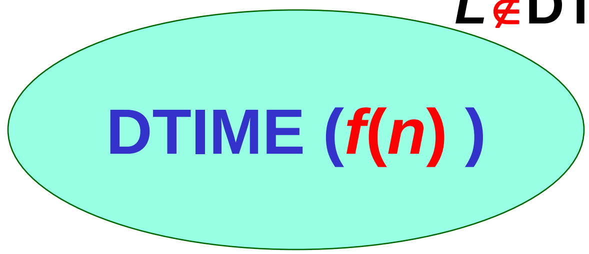
$L \notin \text{DTIME}(f(n))$

$\text{DTIME}(f(n))$

- **Language definition is based on diagonalization**

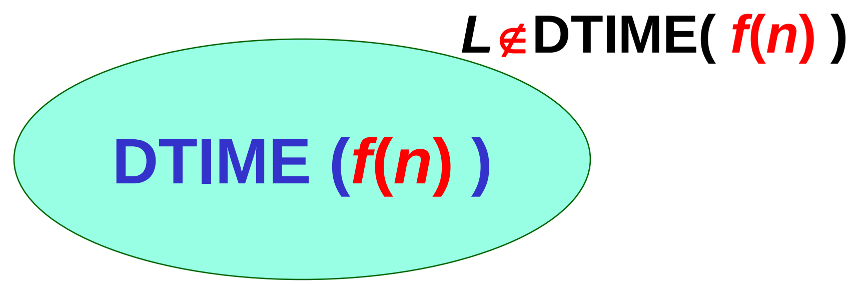
Infinity

$L \notin \mathrm{DTIME}(\,f(n)\,)$



- **Language definition is based on diagonalization**
  - $L = \{\; w_i \;|$

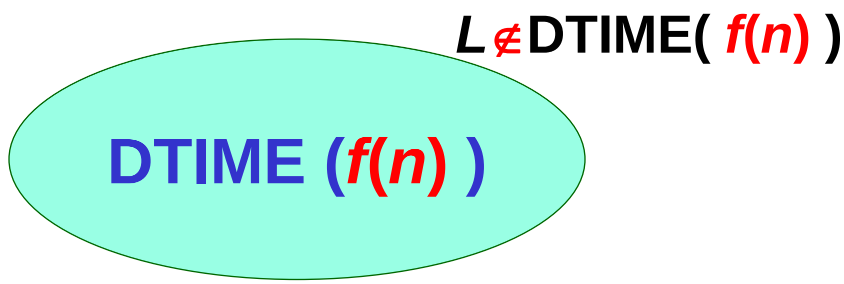
Infinity



- Language definition is based on diagonalization

  - $L = \{\ w_i \mid$
    TM M_i does not accept w_i

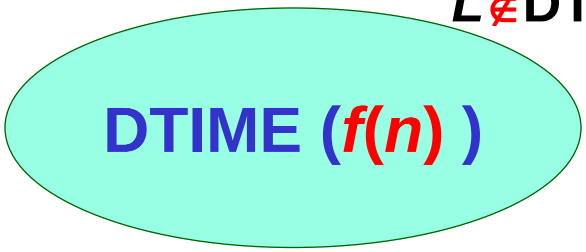
Infinity



- **Language definition is based on diagonalization**
  - **$L = \{\ w_i\ |$ TM M_i does not accept w_i in less than $f(\ |w_i|\ )$ head moves$\}$**

Infinity

$L \notin \mathrm{DTIME}(f(n))$



Infinity

$L \notin \mathrm{DTIME}(f(n))$

$\mathrm{DTIME}(f(n))$

- **Language *L* is recursive**

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

B	*B*	*B*	*B*	*B*	*B*

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- **Language L is recursive**

TM M accepts L and always halts

TM M computes the input length $n = \|w\|$	B	B

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- **Language L is recursive**

TM M accepts L and always halts

TM M computes the input length $n = \|w\|$	*B*	*B*

B	*B*	*B*	*B*	*B*	*B*

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

We calculate $f(n)$

$f(n)$ is a total recursive function

TM M accepts L and always halts

TM M computes the input length $n = \|w\|$				*B*	*B*	
B	*B*	*B*	*B*	*B*	*B*	

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = \|w\|$	B	B	
TM N computes the value $f(n)$ and always halts		B	

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = \|w\|$	B	B	

TM N computes the value $f(n)$ and always halts	B	

B	B	B	B	B	B	

Infinity

$L \notin \mathrm{DTIME}(\, f(n)\,)$

DTIME $(f(n))$

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = \lvert w \rvert$	B	B	

TM N computes the value $f(n)$ and always halts	B	

Head moves counter $f(n)$		B	B	B	

Infinity

$L \notin \text{DTIME}(\ f(n)\ )$

DTIME ($f(n)$)

- **Language *L* is recursive**

TM *M* accepts *L* and always halts

TM *M* computes the input length $n = \|w\|$	*B*	*B*	
TM *N* computes the value $f(n)$ and always halts	*B*		

B	*B*	*B*	*B*	*B*	*B*	

Head moves counter $f(n)$		*B*	*B*	*B*	

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = \|w\|$	B	B	
TM N computes the value $f(n)$ and always halts	B		
TM M finds the index i of string w_i in the canonical sequence TM M finds the binary encoding of i which is also the encoding of TM M_i			
Head moves counter $f(n)$	B	B	B

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = |w|$ | B | B

TM N computes the value $f(n)$ and always halts | B

TM M finds the index i of string w_i in the canonical sequence
TM M finds the binary encoding of i which is also the encoding of TM M_i

B | B | B | B | B | B

Head moves counter $f(n)$ | B | B | B

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = \|w\|$	B	B
TM N computes the value $f(n)$ and always halts		B
TM M finds the index i of string w_i in the canonical sequence TM M finds the binary encoding of i which is also the encoding of TM M_i		
Invalid code - TM M_i has no defined transitions		
Head moves counter $f(n)$	B	B B

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = \|w\|$	B	B	
TM N computes the value $f(n)$ and always halts	B		
TM M finds the index i of string w_i in the canonical sequence TM M finds the binary encoding of i which is also the encoding of TM M_i			
Invalid code - TM M_i has no defined transitions			
Head moves counter $f(n)$	B	B	B

YES

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = |w|$ *B* *B*

TM N computes the value $f(n)$ and always halts *B*

TM M finds the index i of string w_i in the canonical sequence
TM M finds the binary encoding of i which is also the encoding of TM M_i

TM M_i halts and does not accept the string

Head moves counter $f(n)$ *B* *B* *B*

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- **Language L is recursive**

TM M accepts L and always halts

TM M computes the input length $n = \|w\|$	B	B
TM N computes the value $f(n)$ and always halts		B

TM M finds the index i of string w_i in the canonical sequence
TM M finds the binary encoding of i which is also the encoding of TM M_i

TM M_i halts and does not accept the string

YES

Head moves counter $f(n)$ B B B

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- **Language L is recursive**

TM M accepts L and always halts

TM M computes the input length $n = |w|$ *B* *B*

TM N computes the value $f(n)$ and always halts *B*

TM M finds the index i of string w_i in the canonical sequence

TM M finds the binary encoding of i which is also the encoding of TM M_i

TM M_i halts and accepts the string

Head moves counter $f(n)$ *B* *B* *B*

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = \|w\|$	B	B	
TM N computes the value $f(n)$ and always halts	B		
TM M finds the index i of string w_i in the canonical sequence TM M finds the binary encoding of i which is also the encoding of TM M_i			
TM M_i halts and accepts the string			
Head moves counter $f(n)$	B	B	B

NO

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = |w|$ | B | B

TM N computes the value $f(n)$ and always halts | B

TM M finds the index i of string w_i in the canonical sequence

TM M finds the binary encoding of i which is also the encoding of TM M_i

TM M_i does not accept the string, head moves > $f(n)$ times

Head moves counter $f(n)$ | B | B | B

Infinity

$L \notin$ DTIME($f(n)$)

DTIME ($f(n)$)

- Language L is recursive

TM M accepts L and always halts

TM M computes the input length $n = |w|$ | B | B

TM N computes the value $f(n)$ and always halts | B

TM M finds the index i of string w_i in the canonical sequence
TM M finds the binary encoding of i which is also the encoding of TM M_i

TM M_i does not accept the string, head moves > $f(n)$ times → YES

Head moves counter $f(n)$ | B | B | B

Infinity

$L \notin$ DTIME($f(n)$)

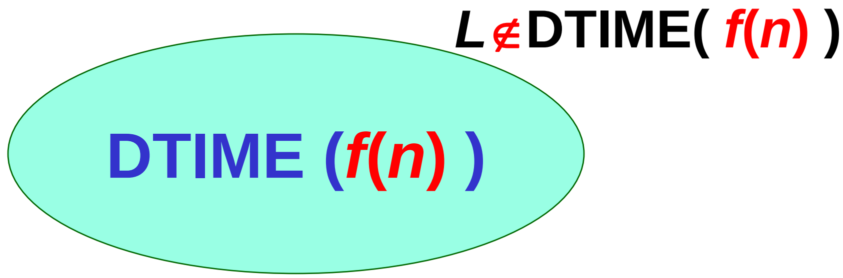
DTIME ($f(n)$)

- **Language L is recursive**

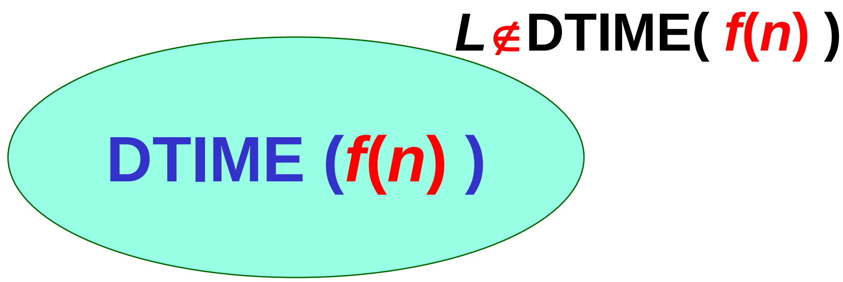
TM M accepts L and always halts

TM M computes the input length $n = \|w\|$	B	B
TM N computes the value $f(n)$ and always halts		B
TM M finds the index i of string w_i in the canonical sequence TM M finds the binary encoding of i which is also the encoding of TM M_i		
TM M_i does not accept the string, head moves > $f(n)$ times		
Head moves counter $f(n)$	B	B B

Infinity

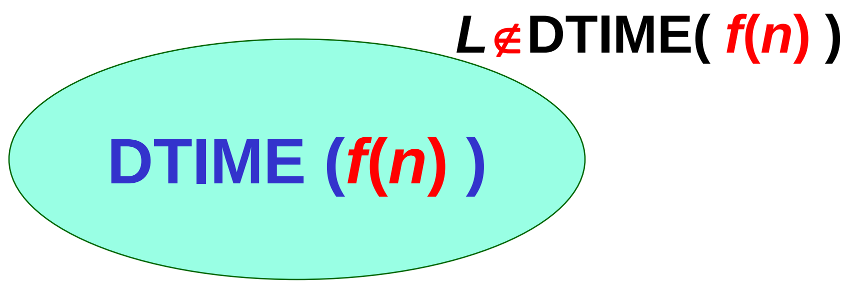


Infinity



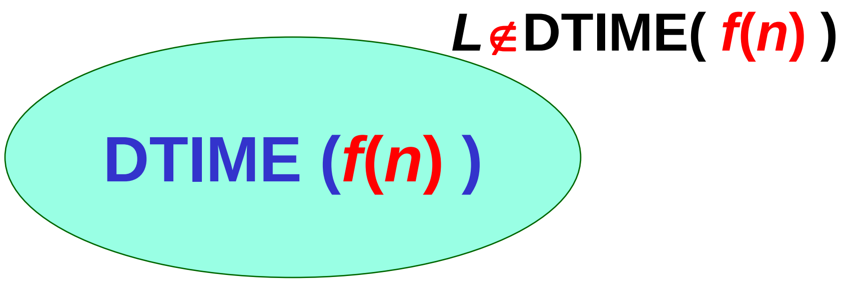
- A proof that $L \notin \text{DTIME}(f(n))$

Infinity



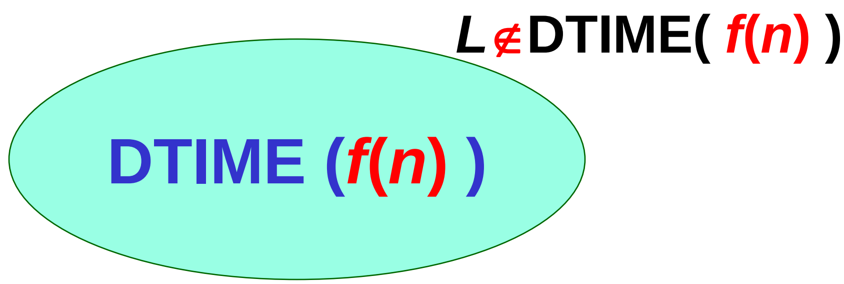
- A proof that $L \notin \mathrm{DTIME}(f(n))$
  - Assumption: $L=L(M_i)$

Infinity



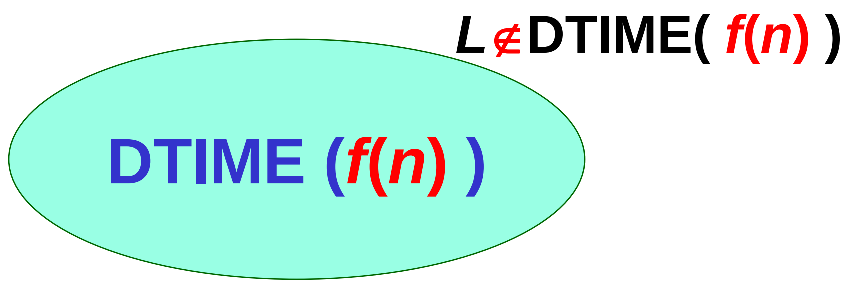
- A proof that $L \notin$ DTIME($f(n)$)
  - Assumption: $L = L(M_i)$
    - —language L is accepted by TM M_i of time complexity $f(n)$

Infinity



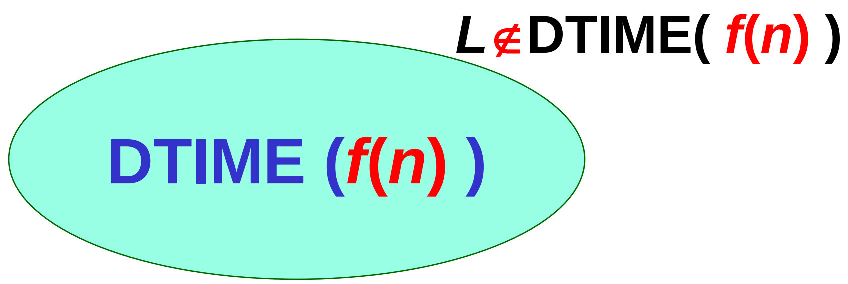
- A proof that $L \notin \mathrm{DTIME}(f(n))$
  - Assumption: $L = L(M_i)$
    - —language L is accepted by TM M_i of time complexity $f(n)$
  - $|w_i| = n$

Infinity



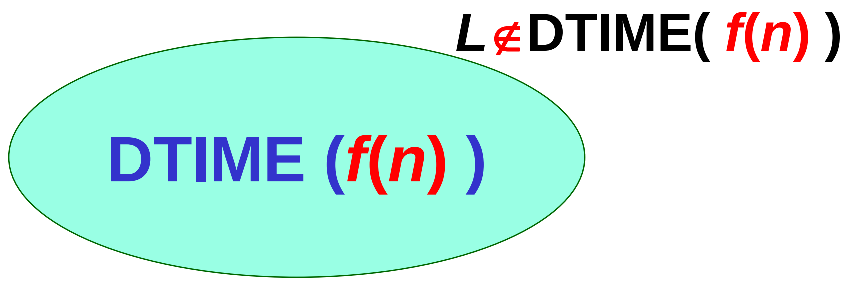
- A proof that $L \notin \text{DTIME}(f(n))$
  - Assumption: $L = L(M_i)$
    - —language L is accepted by TM M_i of time complexity $f(n)$
  - $|w_i| = n$
  - Assumption : $w_i \in L(M_i)$

Infinity



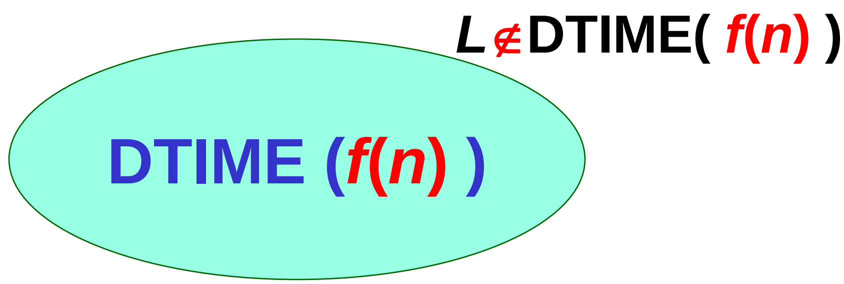
- A proof that $L \notin \mathrm{DTIME}(f(n))$
  - Assumption: $L=L(M_i)$
    - —language L is accepted by TM M_i of time complexity $f(n)$
  - $|w_i| = n$
  - Assumption : $w_i \in L(M_i)$
    - $\Rightarrow$ TM M_i accepts the string w_i in less than $f(n)$ head moves

Infinity



- A proof that $L \notin \mathrm{DTIME}(f(n))$
  - Assumption: $L=L(M_i)$
    - —language L is accepted by TM M_i of time complexity $f(n)$
  - $|w_i| = n$
  - Assumption : $w_i \in L(M_i)$
    - $\Rightarrow$ TM M_i accepts the string w_i in less than $f(n)$ head moves
    - $\Rightarrow$ $w_i \notin L$, because $L=\{w_i \mid M_i$ does not accept w_i in less than $f(|w_i|)$ head moves$\}$

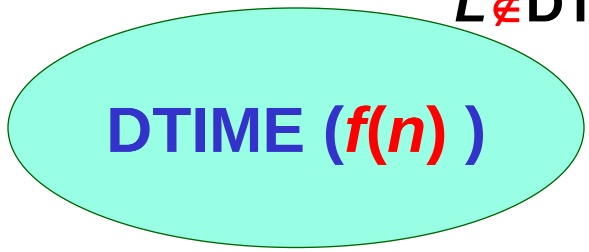
Infinity



- A proof that $L \notin \mathrm{DTIME}(f(n))$
  - **Assumption: $L=L(M_i)$**
    - —language L is accepted by TM M_i of time complexity $f(n)$
  - $|w_i| = n$
  - **Assumption : $w_i \in L(M_i)$**
    - $\Rightarrow$ TM M_i accepts the string w_i in less than $f(n)$ head moves
    - $\Rightarrow$ $w_i \notin L$, because $L=\{w_i \mid M_i$ does not accept w_i in less than $f(|w_i|)$ head moves$\}$
    - $\Rightarrow$ $w_i \notin L(M_i)$, because we assume that $L=L(M_i)$

Infinity

$L \notin \mathrm{DTIME}(\ f(n)\ )$



- A proof that $L \notin \mathrm{DTIME}(f(n))$
  - Assumption: $L=L(M_i)$
    - —language L is accepted by TM M_i of time complexity $f(n)$
  - $|w_i| = n$
  - Assumption : $w_i \in L(M_i)$
    - $\Rightarrow$ TM M_i accepts the string w_i in less than $f(n)$ head moves
    - $\Rightarrow$ $w_i \notin L$, because $L=\{w_i \mid M_i$ does not accept w_i in less than $f(|w_i|)$ head moves$\}$
    - $\Rightarrow$ $w_i \notin L(M_i)$, because we assume that $L=L(M_i)$
    - $\Rightarrow$ contradiction

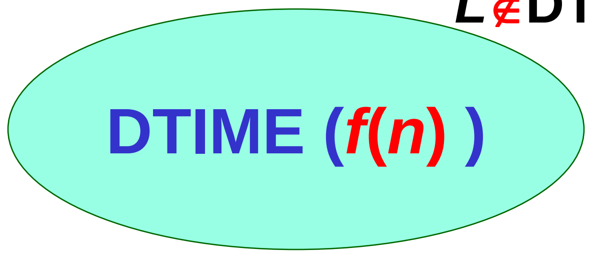
Infinity

$L \notin \mathrm{DTIME}(\, f(n) \,)$

$\mathrm{DTIME}\ (f(n))$

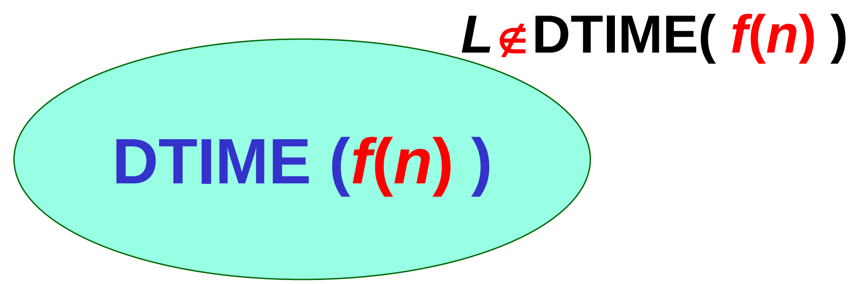
Infinity

$L \notin \mathrm{DTIME}(\, f(n) \,)$



- **Assumption:** $w_i \notin L(M_i)$

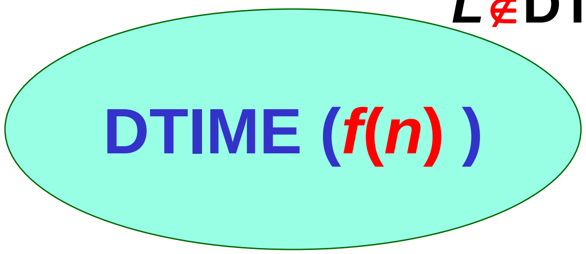
Infinity



- **Assumption:** $w_i \notin L(M_i)$
  - $\Rightarrow$ **TM M_i does not accept the string w_i in less than $f(n)$ head moves**

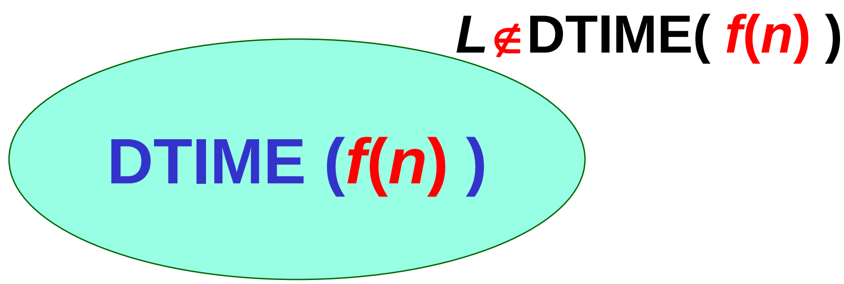
Infinity

$L \notin \mathrm{DTIME}(\, f(n)\,)$



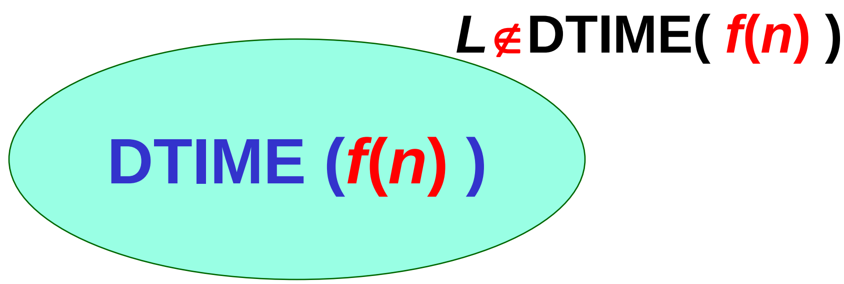
- **Assumption:** $w_i \notin L(M_i)$
  - $\Rightarrow$ **TM M_i does not accept the string w_i in less than $f(n)$ head moves**
  - $\Rightarrow$ $w_i \in L$, **because $L=\{w_i \mid M_i$ does not accept w_i in less than $f(|w_i|)$ head moves$\}$**

Infinity



- **Assumption:** $w_i \notin L(M_i)$
  - $\Rightarrow$ **TM M_i does not accept the string w_i in less than $f(n)$ head moves**
  - $\Rightarrow$ $w_i \in L$, **because $L=\{w_i \mid M_i$ does not accept w_i in less than $f(|w_i|)$ head moves$\}$**
  - $\Rightarrow$ $w_i \in L(M_i)$, **because we assume that $L=L(M_i)$**

Infinity



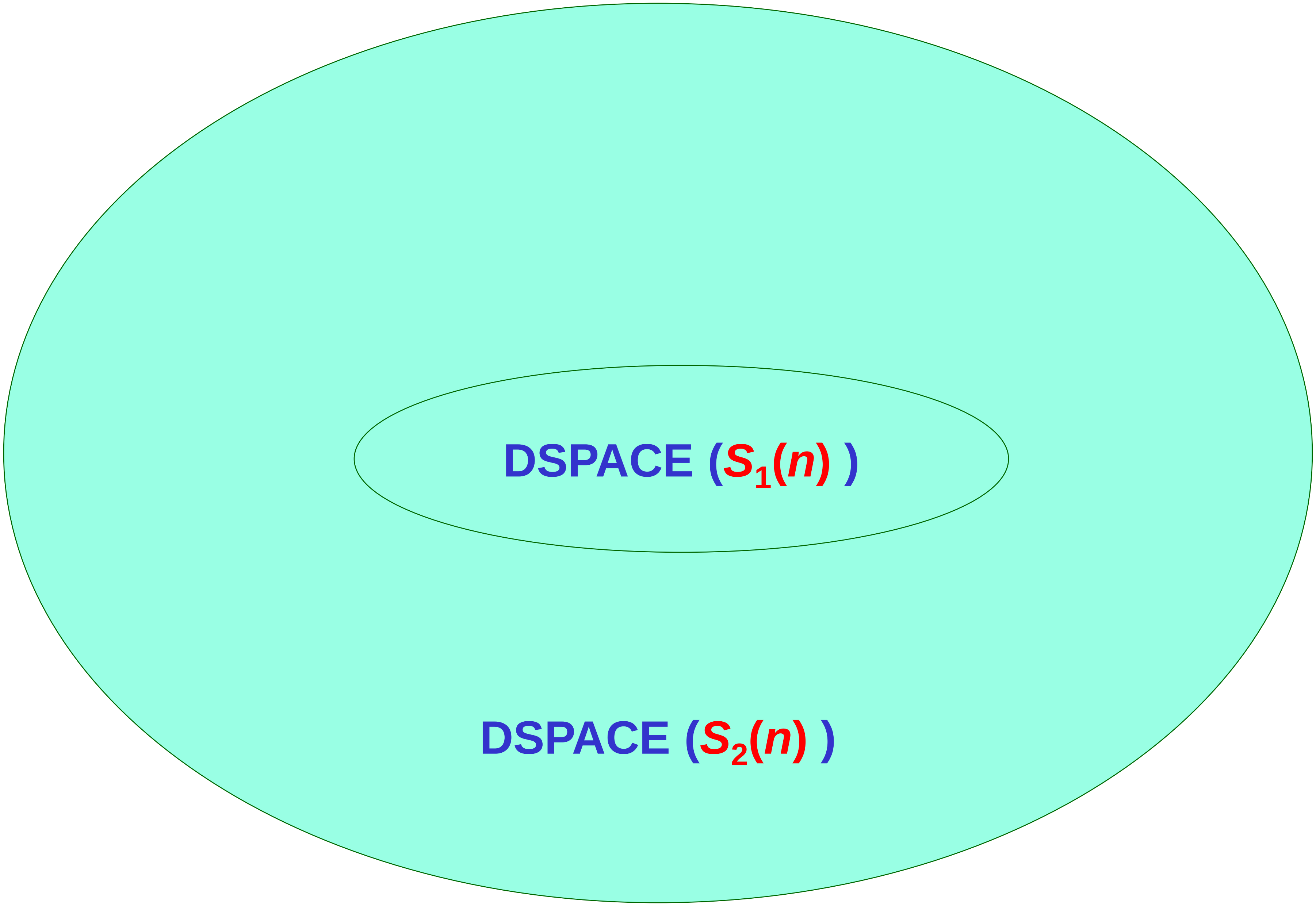
- **Assumption:** $w_i \notin L(M_i)$
  - $\Rightarrow$ **TM M_i does not accept the string w_i in less than $f(n)$ head moves**
  - $\Rightarrow$ $w_i \in L$, **because $L=\{w_i \mid M_i$ does not accept w_i in less than $f(|w_i|)$ head moves$\}$**
  - $\Rightarrow$ $w_i \in L(M_i)$, **because we assume that $L=L(M_i)$**
  - $\Rightarrow$ **contradiction**

Continuity of the hierarchy for fully space-constructible functions

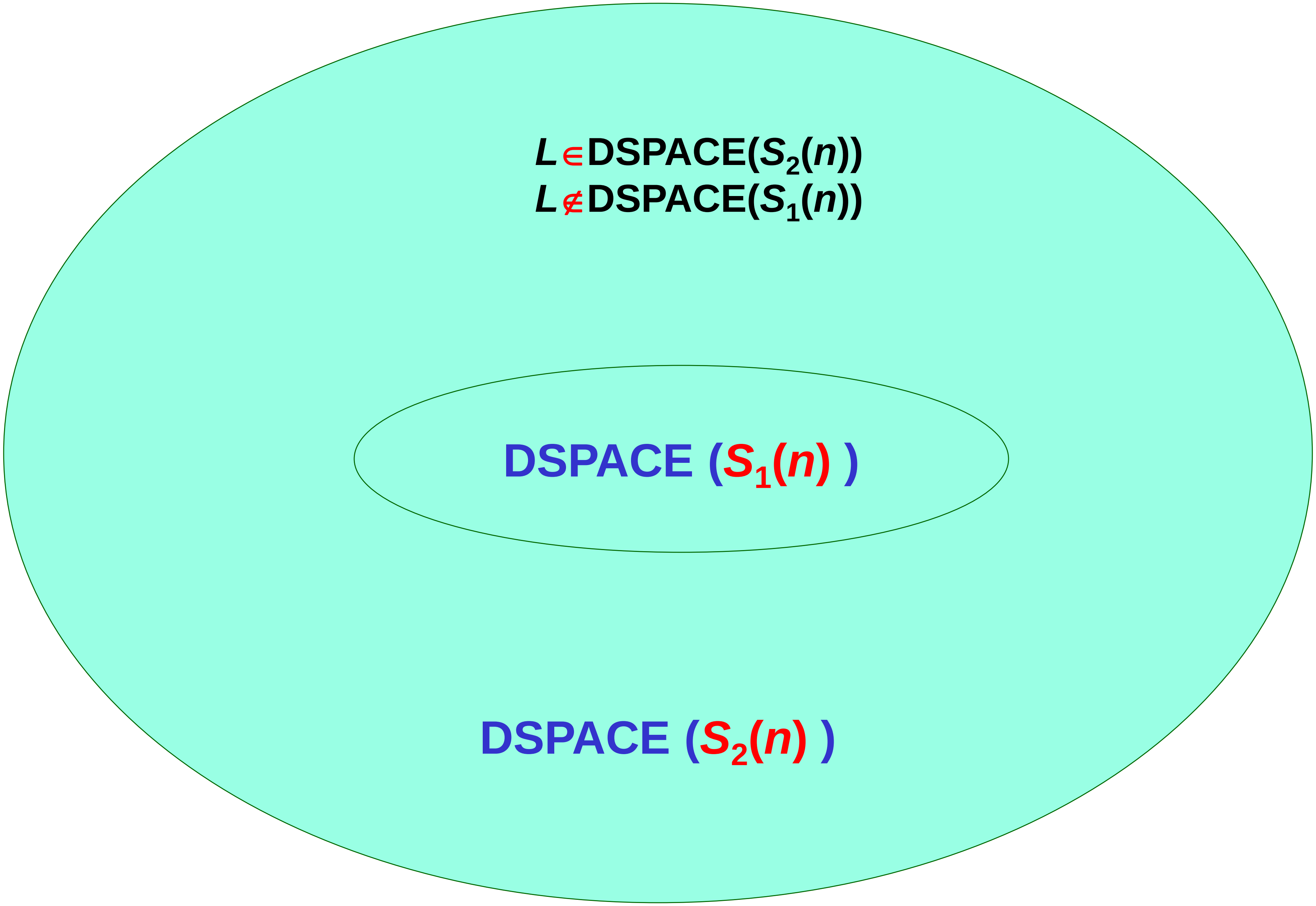
Continuity of the hierarchy for fully space-constructible functions

DSPACE ($S_1(n)$)

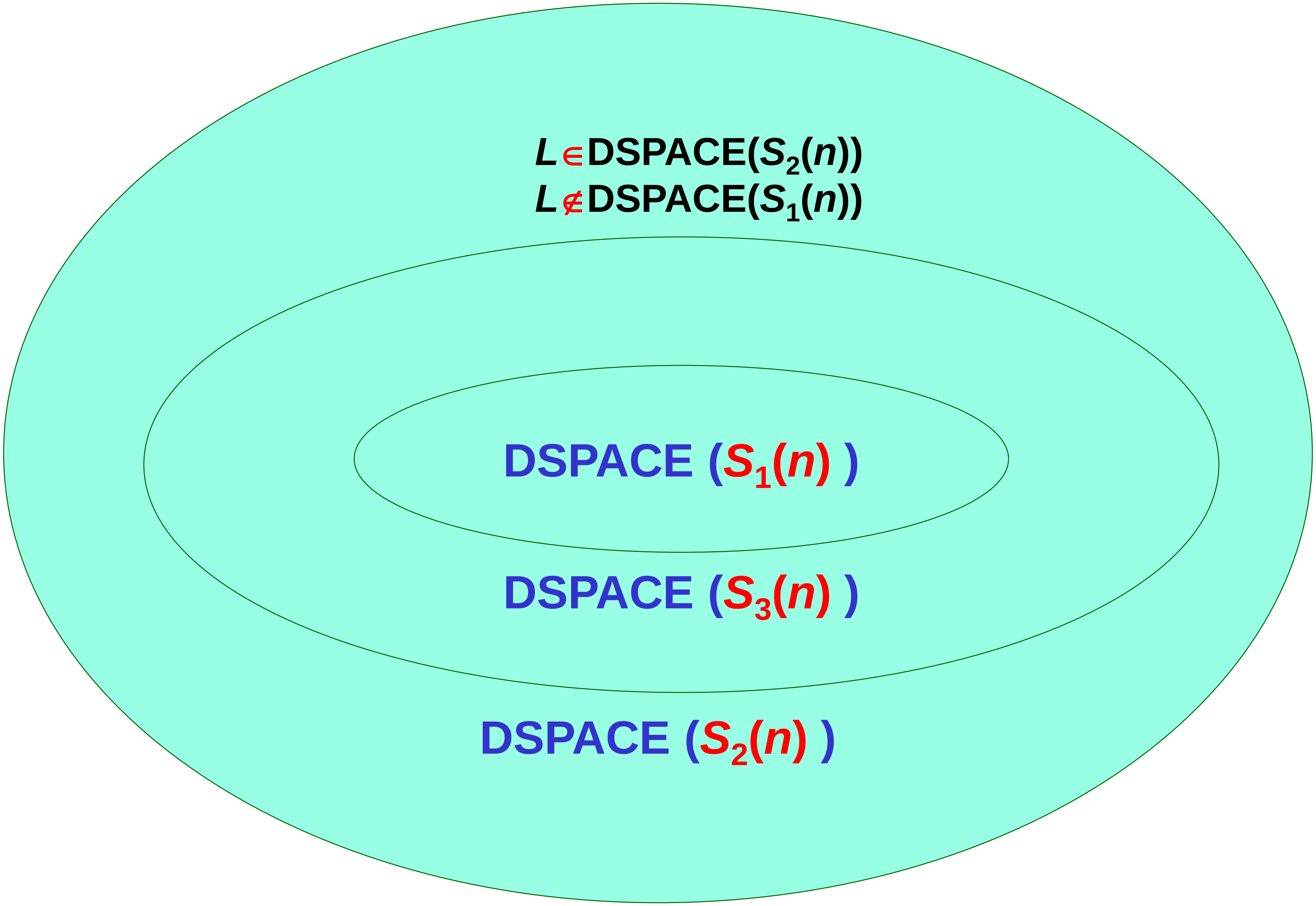
Continuity of the hierarchy for fully space-constructible functions



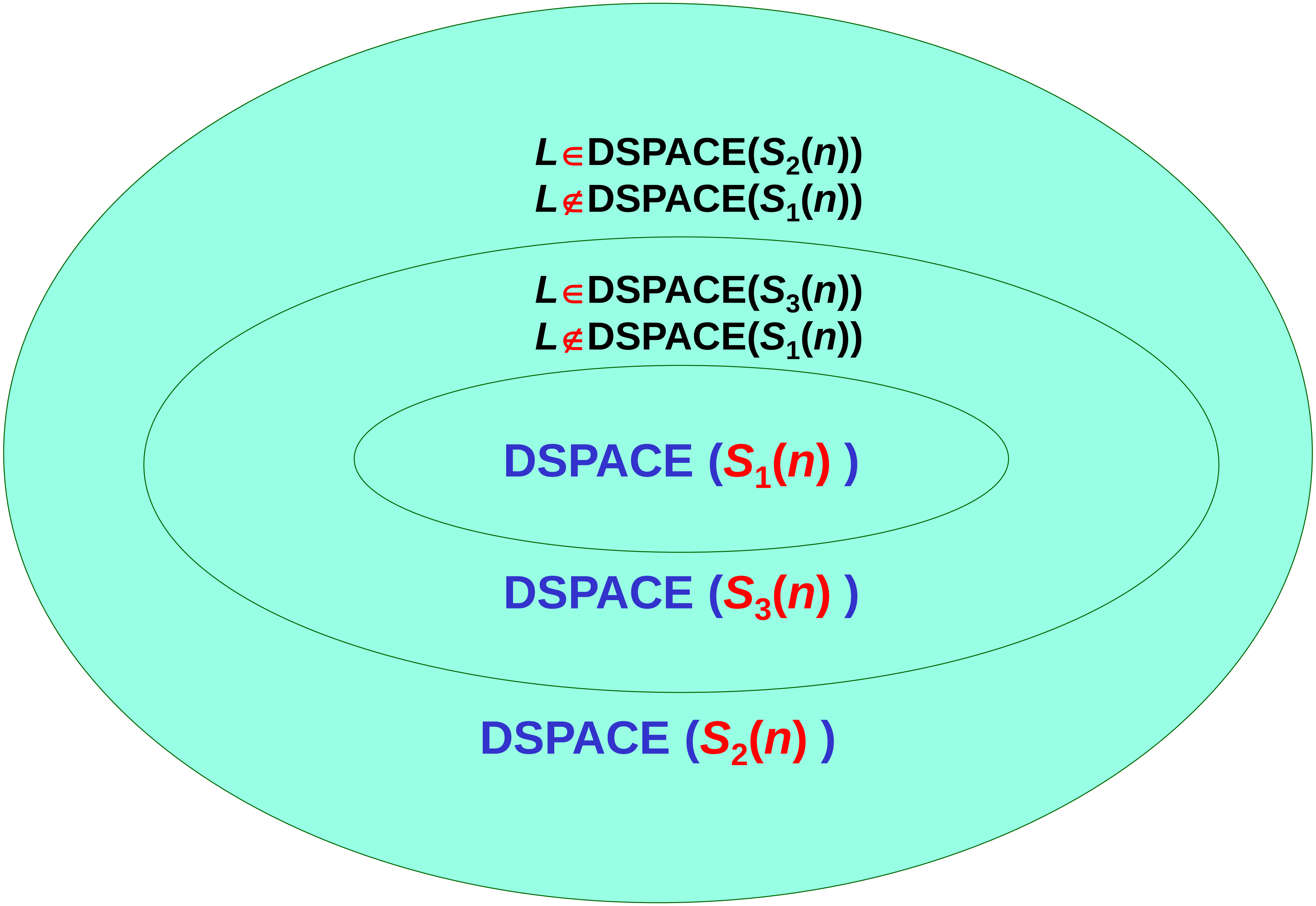
Continuity of the hierarchy for fully space-constructible functions



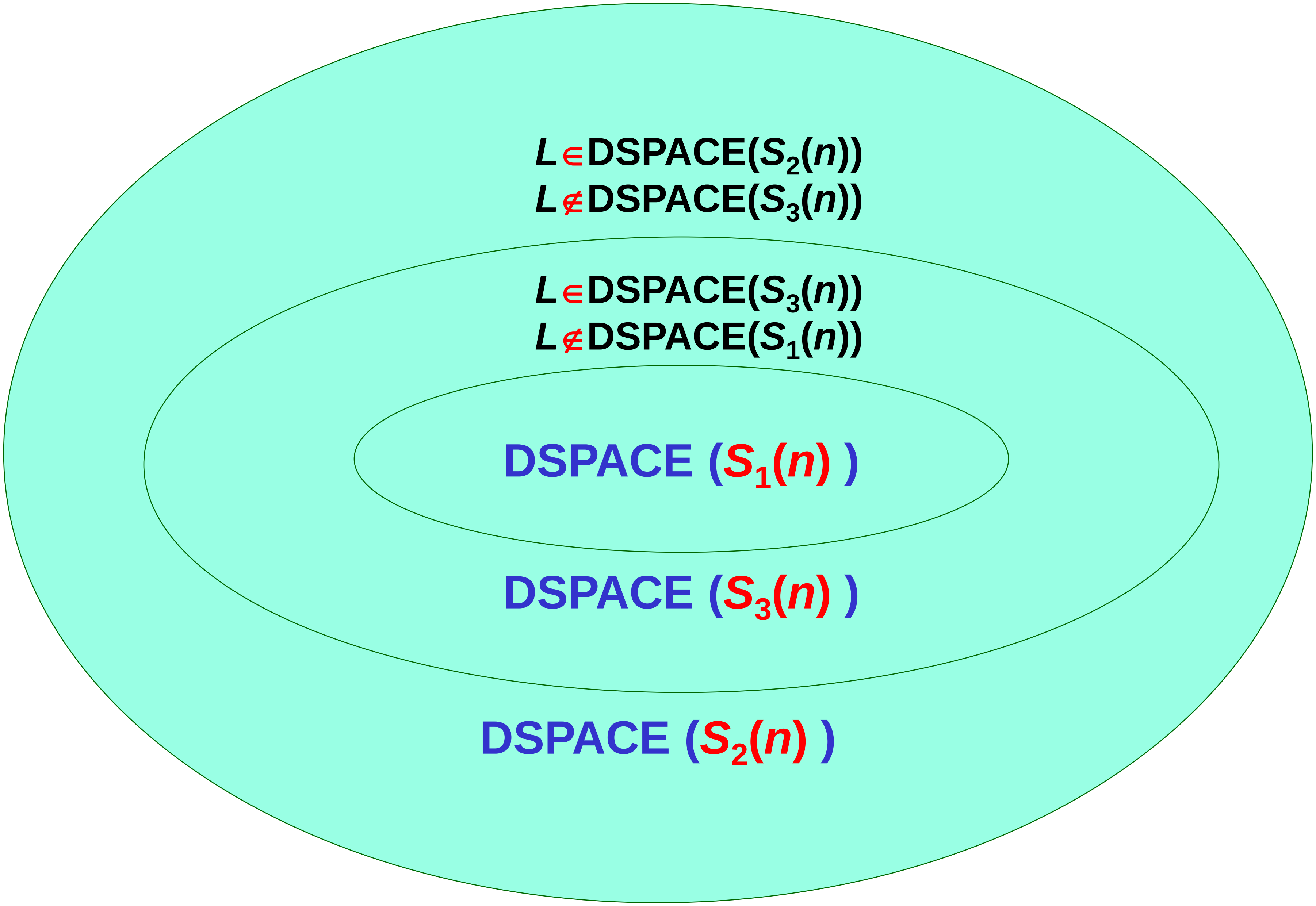
Continuity of the hierarchy for fully space-constructible functions



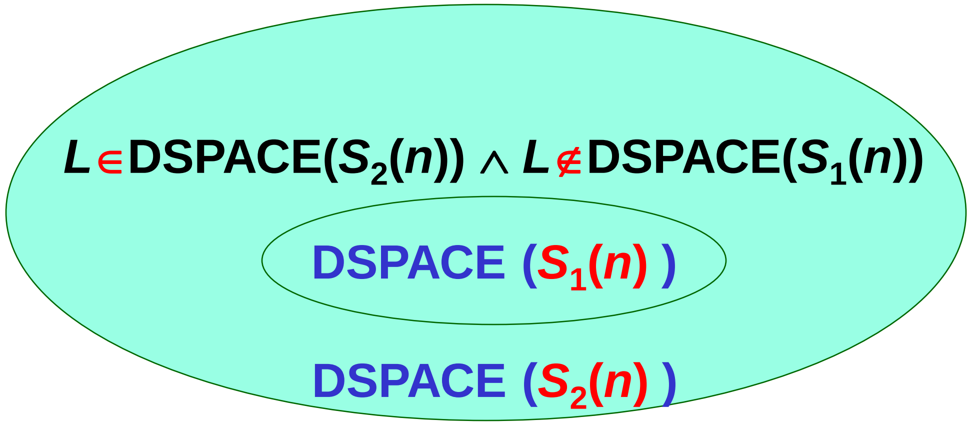
Continuity of the hierarchy for fully space-constructible functions



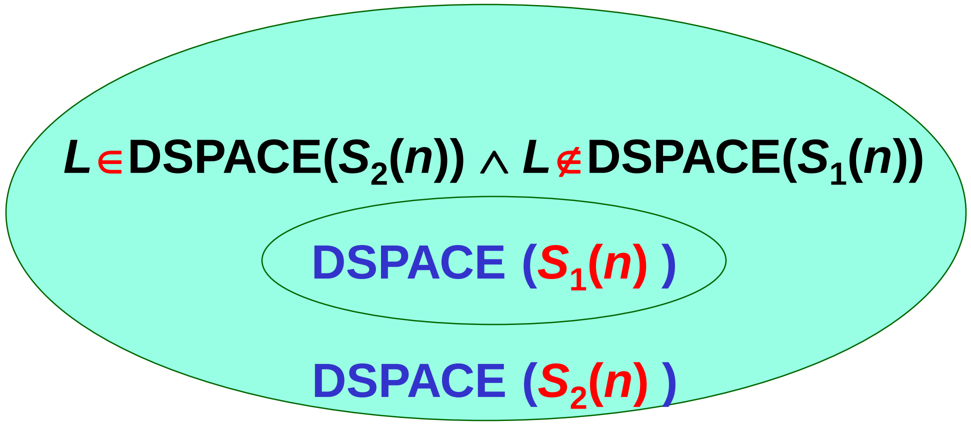
Continuity of the hierarchy for fully space-constructible functions



Continuity of the hierarchy for fully space-constructible functions

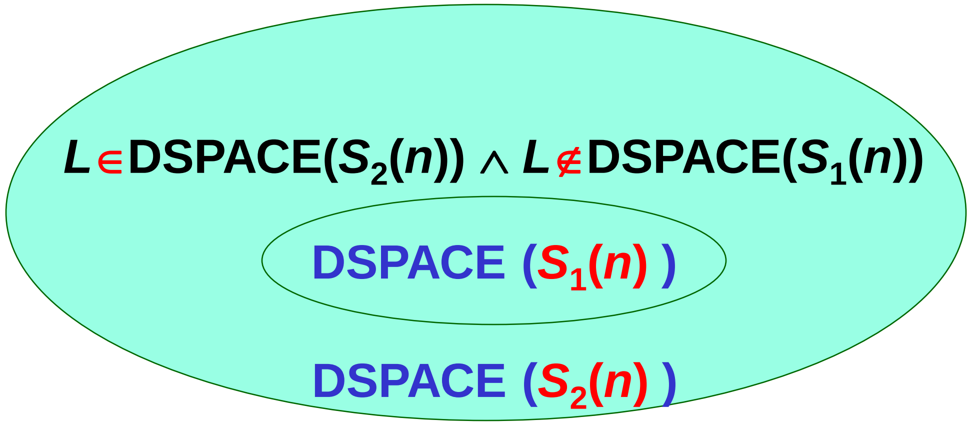


Continuity of the hierarchy for fully space-constructible functions



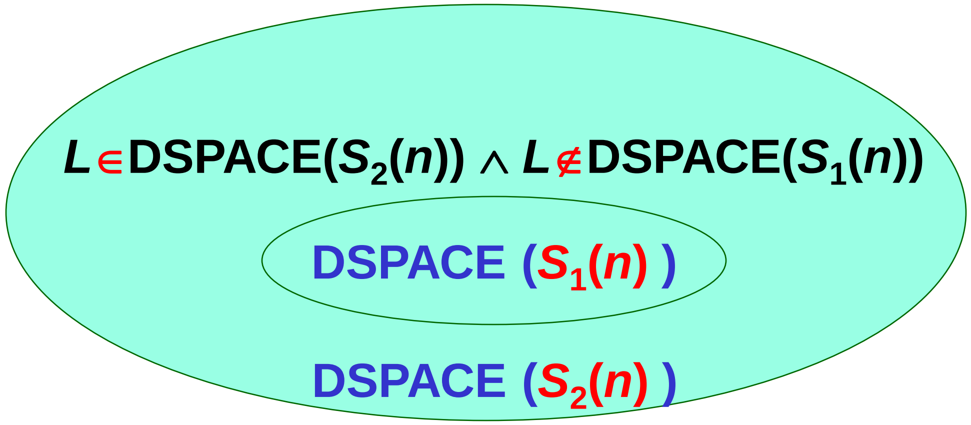
- **If**

Continuity of the hierarchy for fully space-constructible functions



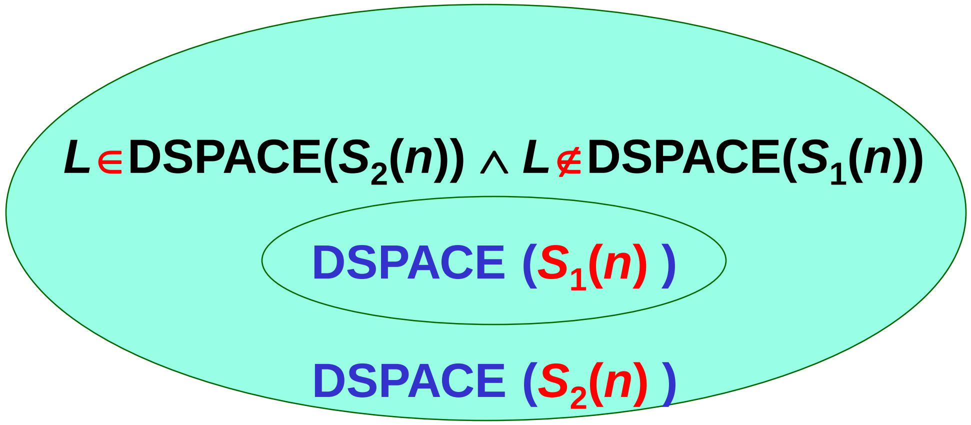
- **If**
  - **$S_2(n)$ is a fully space-constructible function**

Continuity of the hierarchy for fully space-constructible functions



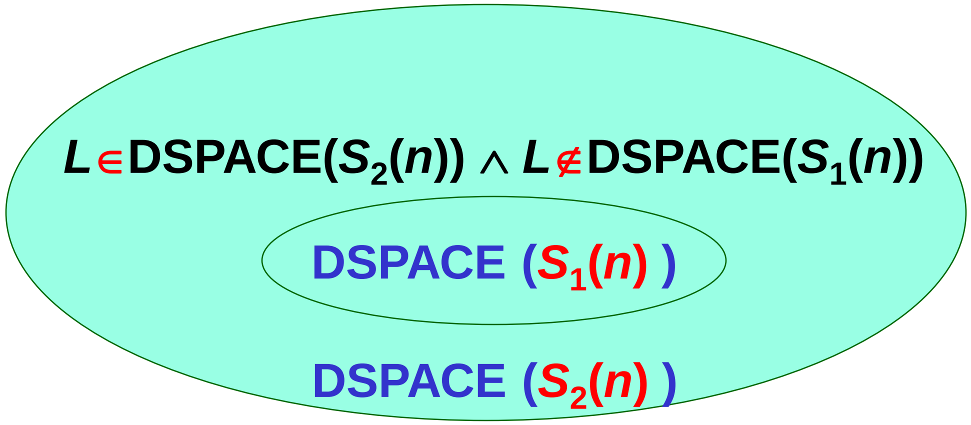
- **If**
    - **$S_2(n)$ is a fully space-constructible function**
    - **$\inf_{n\to\infty} S_1(n)/S_2(n) = 0$**

Continuity of the hierarchy for fully space-constructible functions



- **If**
  - $S_2(n)$ **is a fully space-constructible function**
  - $\inf_{n\to\infty} S_1(n)/S_2(n) = 0$
  - $S_1(n)$ **and** $S_2(n)$ **are at least** $\log_2 n$

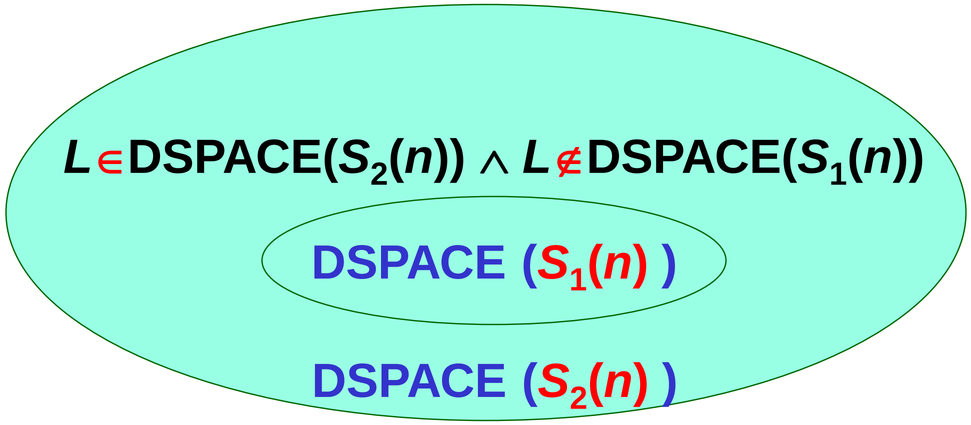
Continuity of the hierarchy for fully space-constructible functions



- **If**
  - **$S_2(n)$ is a fully space-constructible function**
  - **$\inf_{n\to\infty} S_1(n)/S_2(n) = 0$**
  - **$S_1(n)$ and $S_2(n)$ are at least $\log_2 n$**

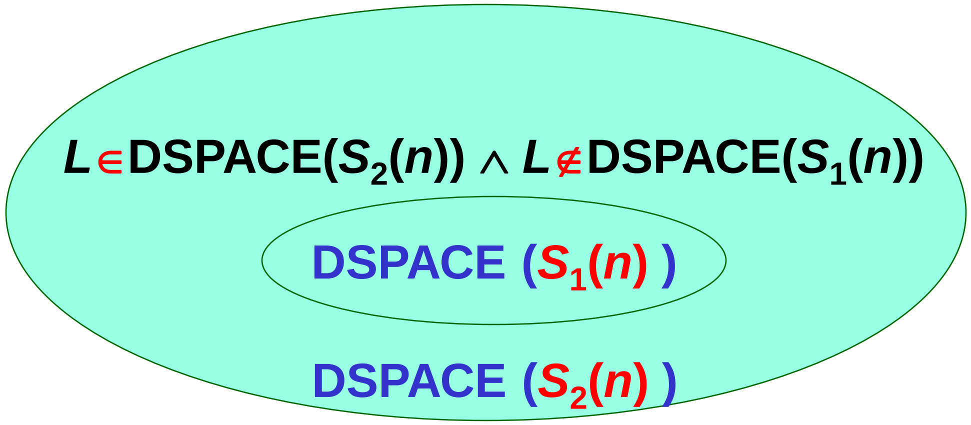
- **Then there is a language L**

Continuity of the hierarchy for fully space-constructible functions

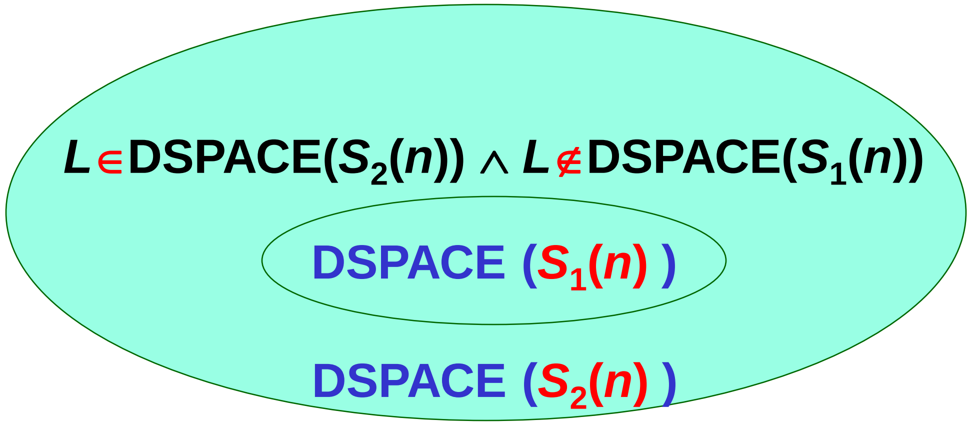


- **If**
  - **$S_2(n)$ is a fully space-constructible function**
  - **$\inf_{n\to\infty} S_1(n)/S_2(n) = 0$**
  - **$S_1(n)$ and $S_2(n)$ are at least $\log_2 n$**
- **Then there is a language L**
  - **$L \in \text{DSPACE}(S_2(n)) \wedge L \notin \text{DSPACE}(S_1(n))$**

Continuity of the hierarchy for fully space-constructible functions

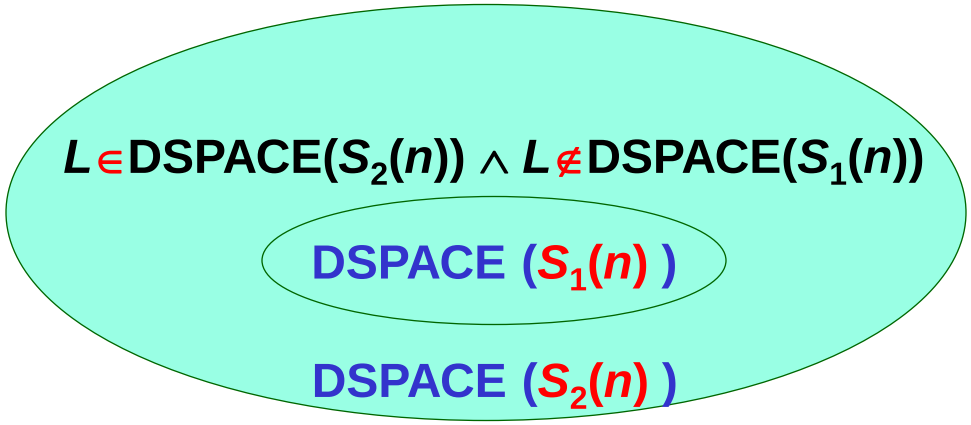


Continuity of the hierarchy for fully space-constructible functions



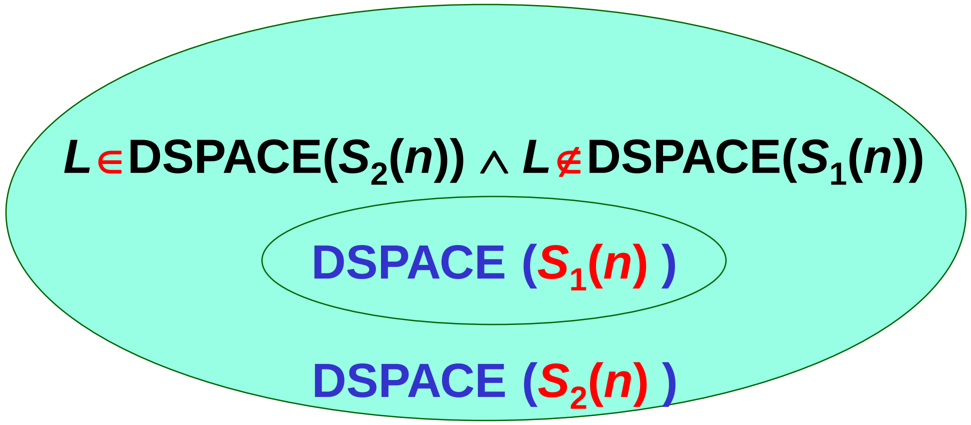
- **We construct TM *M* for which:**

Continuity of the hierarchy for fully space-constructible functions



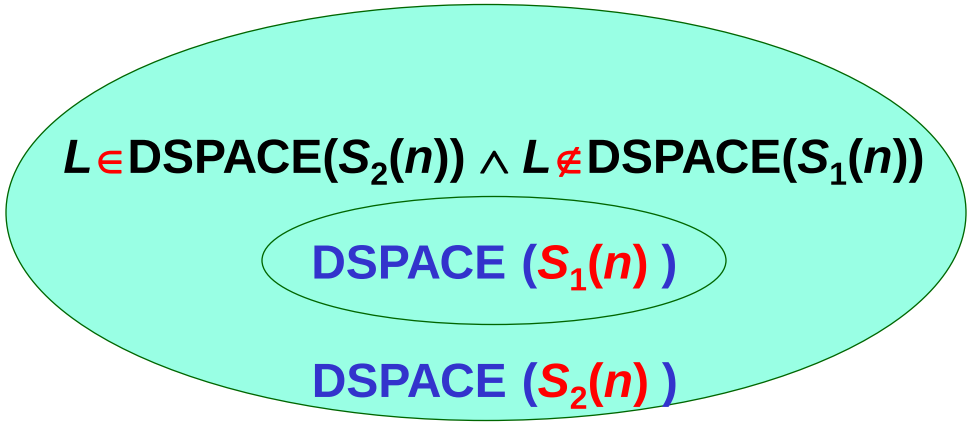
- **We construct TM *M* for which:**
  - **TM *M* has space complexity of $S_2(n)$**

Continuity of the hierarchy for fully space-constructible functions

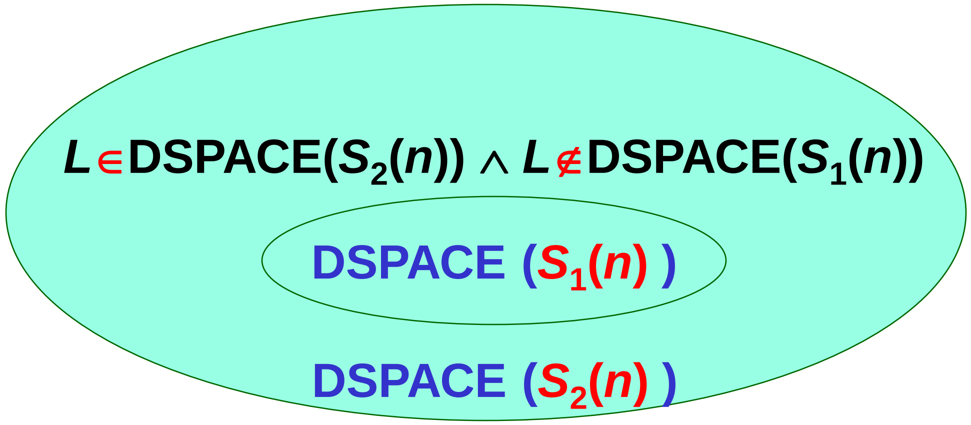


- **We construct TM *M* for which:**
  - **TM *M* has space complexity of $S_2(n)$**
  - **TM *M* gives the opposite decision than any TM with space complexity $S_1(n)$ for at least one input string**

Continuity of the hierarchy for fully space-constructible functions

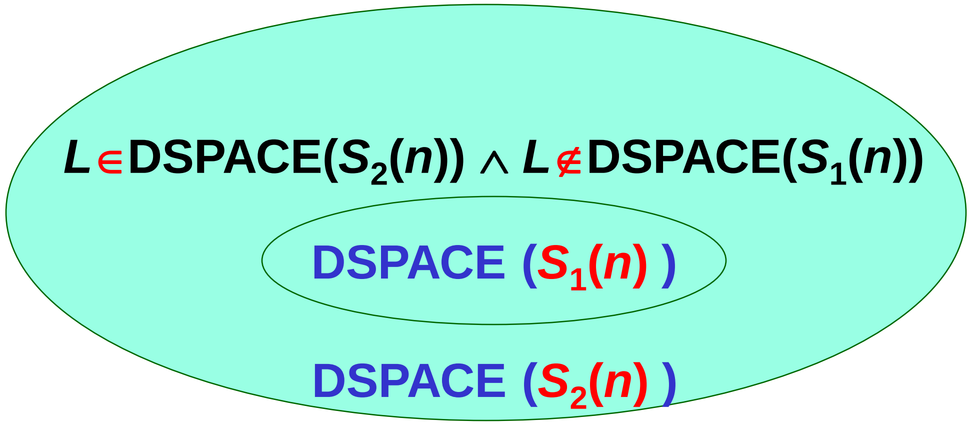


Continuity of the hierarchy for fully space-constructible functions



- **TM *M* – ensure a space complexity $S_2(n)$**

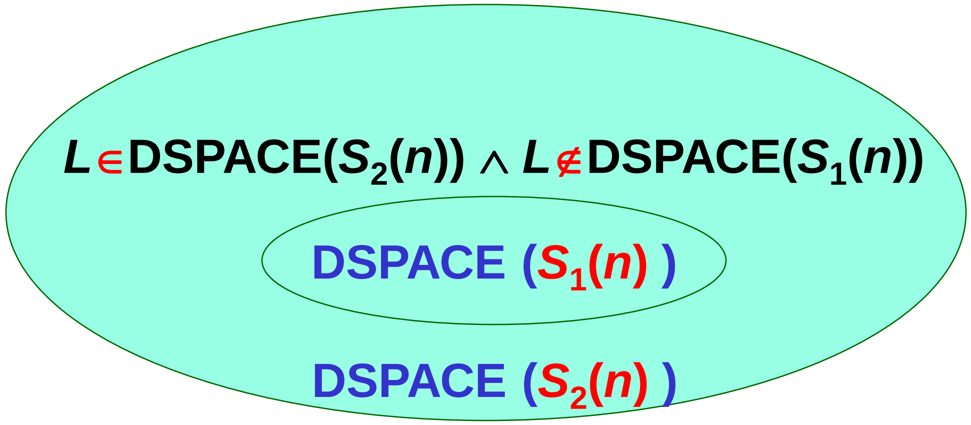
Continuity of the hierarchy for fully space-constructible functions



- **TM *M* – ensure a space complexity $S_2(n)$**

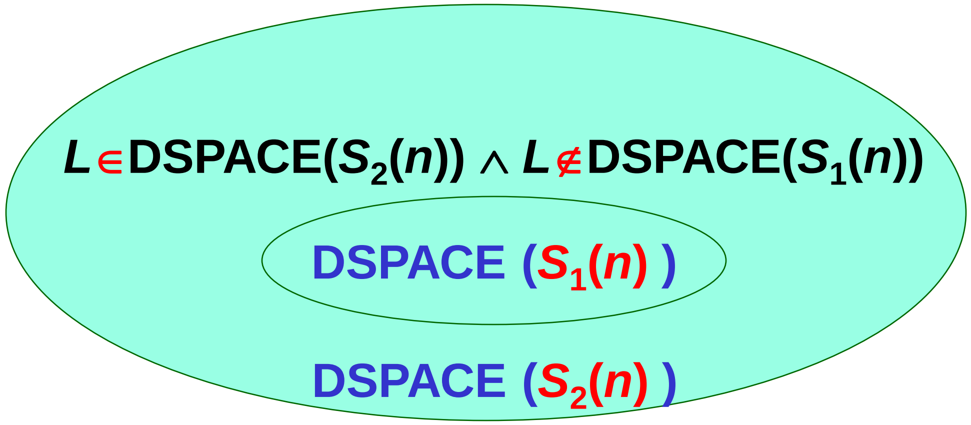
  - **We simulate any TM M_{S2} with space complexity $S_2(n)$**

Continuity of the hierarchy for fully space-constructible functions



- **TM *M* – ensure a space complexity $S_2(n)$**
  - **We simulate any TM M_{S2} with space complexity $S_2(n)$**
  - **$S_2(n)$ is fully space-constructible**

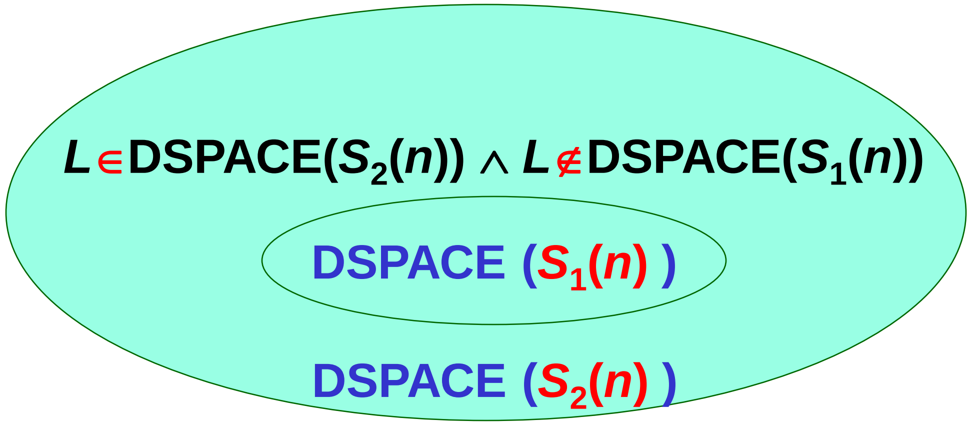
Continuity of the hierarchy for fully space-constructible functions



- **TM M – ensure a space complexity $S_2(n)$**
  - **We simulate any TM M_{S2} with space complexity $S_2(n)$**
  - **$S_2(n)$ is fully space-constructible**

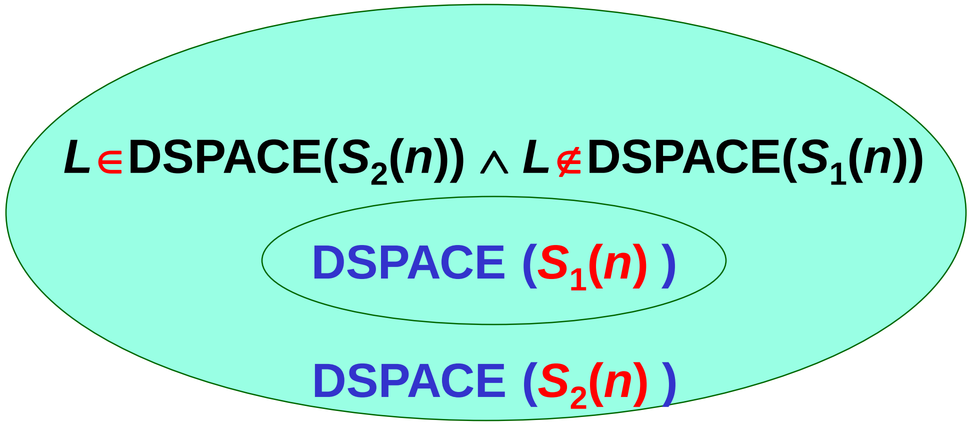
    $\Rightarrow$ **TM M_{S2} for any string of length n uses all $S_2(n)$ cells**

Continuity of the hierarchy for fully space-constructible functions



- **TM *M* - ensure a space complexity $S_2(n)$**

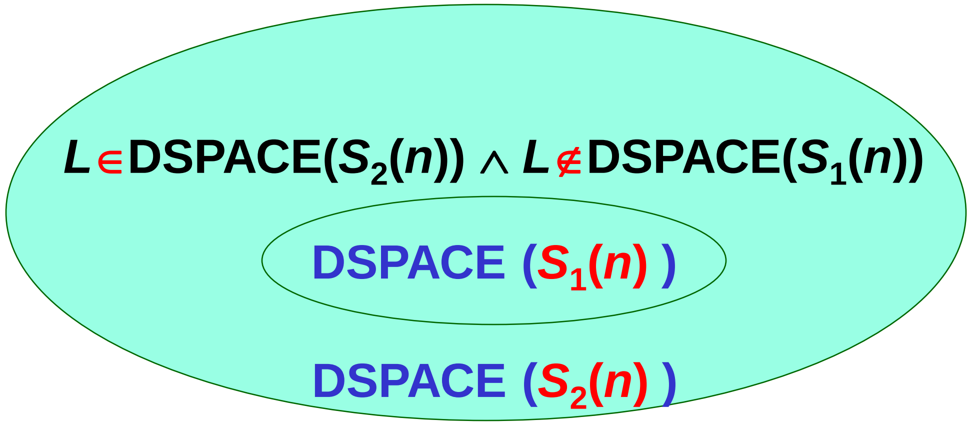
Continuity of the hierarchy for fully space-constructible functions



- **TM *M* - ensure a space complexity $S_2(n)$**

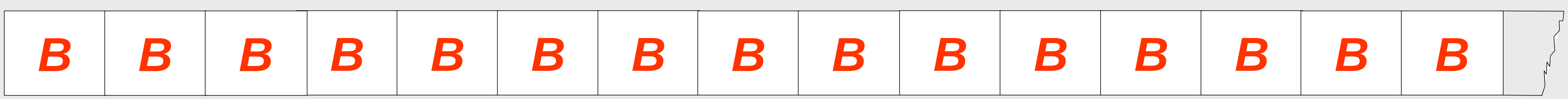
TM *M* which accepts the language *L*

Continuity of the hierarchy for fully space-constructible functions

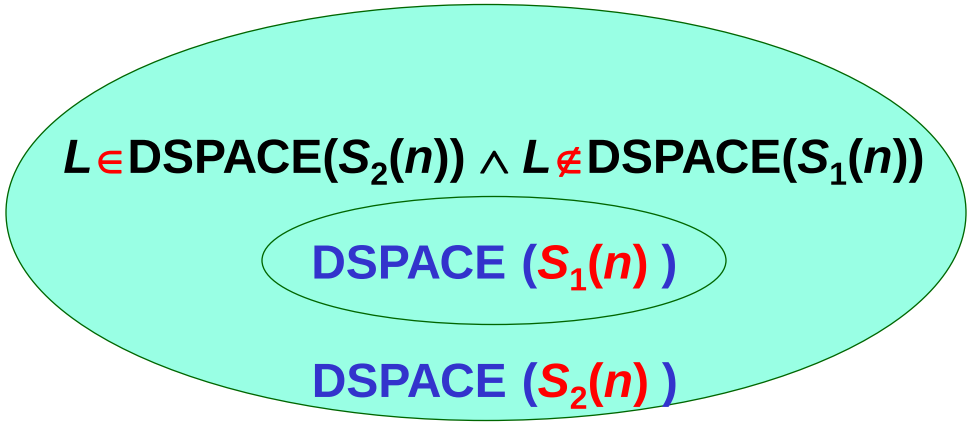


- TM M - ensure a space complexity $S_2(n)$

TM *M* which accepts the language *L*

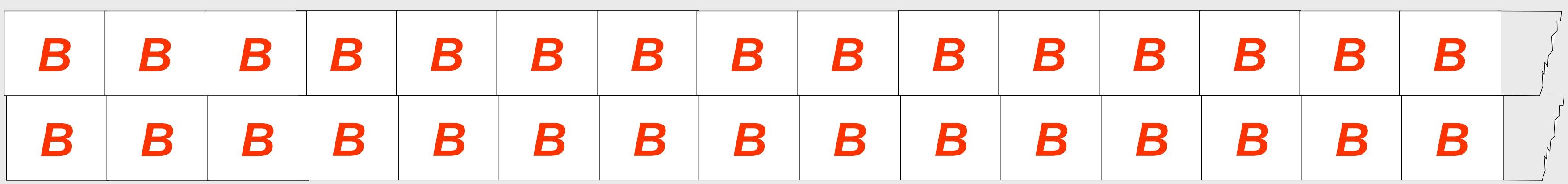


Continuity of the hierarchy for fully space-constructible functions

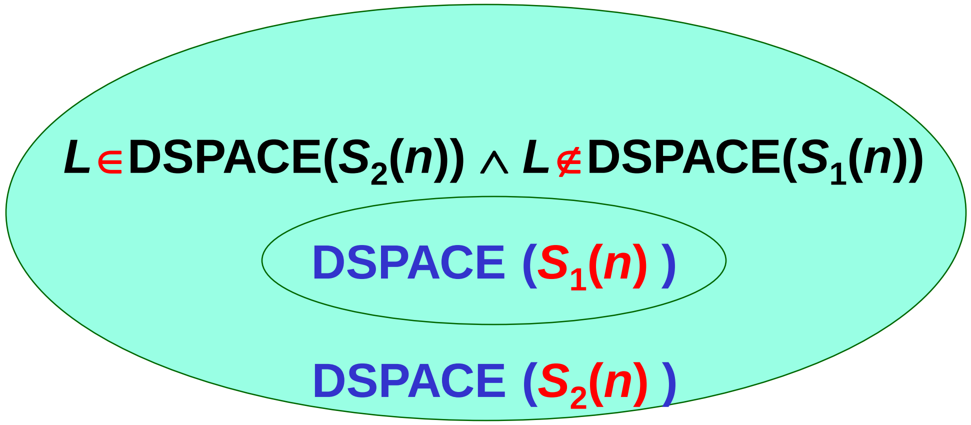


- **TM M - ensure a space complexity $S_2(n)$**

TM M which accepts the language L



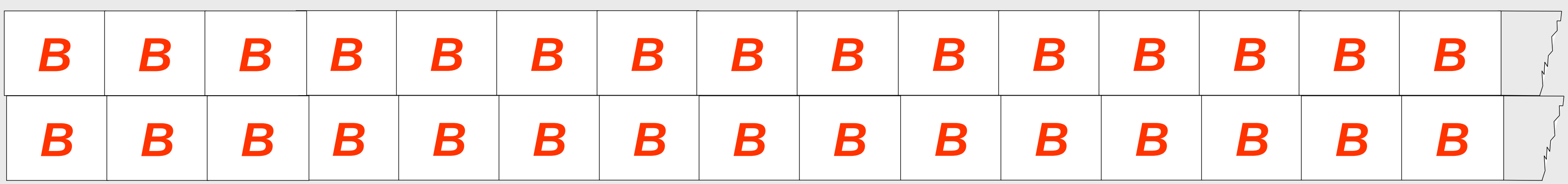
Continuity of the hierarchy for fully space-constructible functions

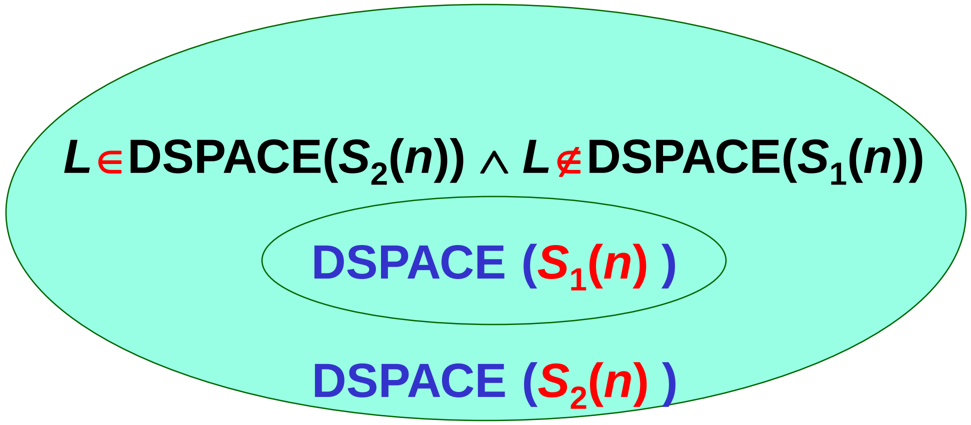


- **TM *M* - ensure a space complexity $S_2(n)$**

TM *M* which accepts the language *L*

Simulation of TM M_{S2}

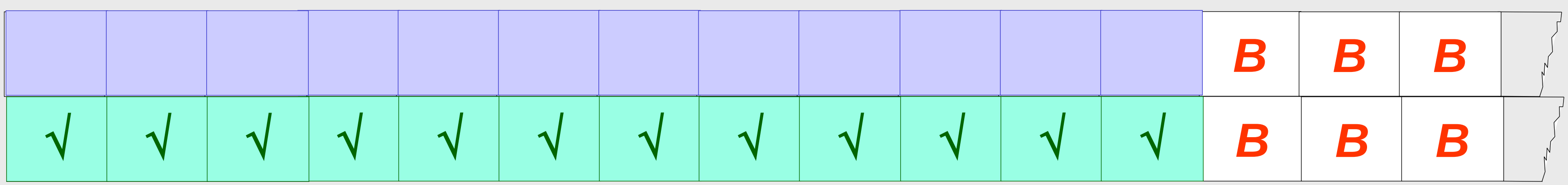


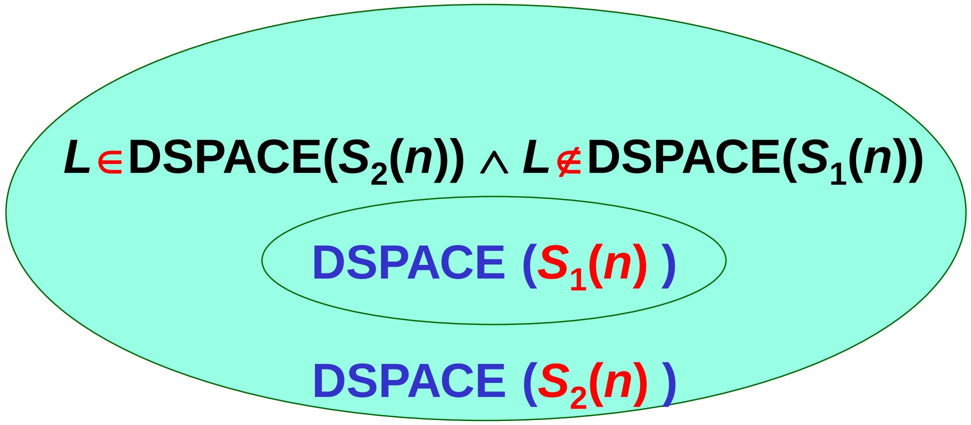


- TM M - ensure a space complexity $S_2(n)$

TM *M* which accepts the language *L*

Simulation of TM M_{S2}

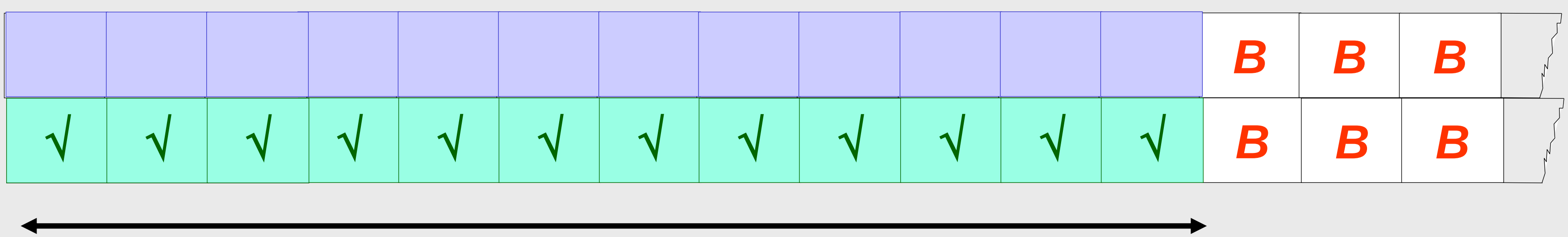




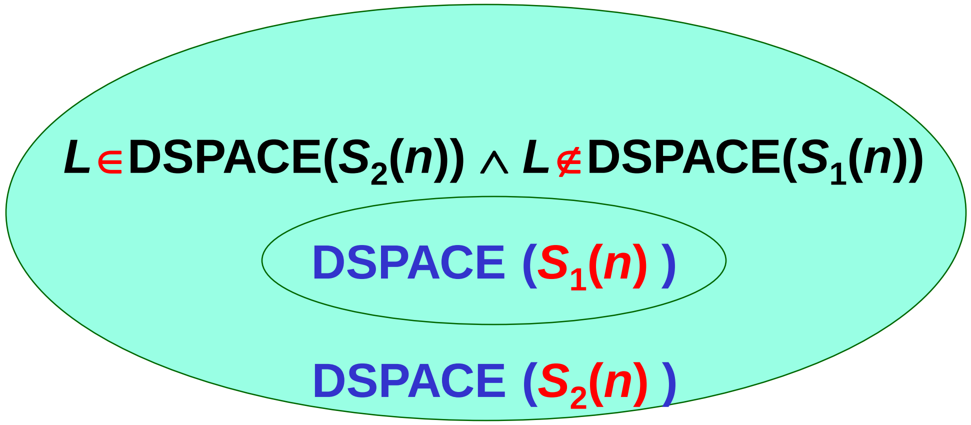
- TM *M* - ensure a space complexity $S_2(n)$

TM *M* which accepts the language *L*

Simulation of TM M_{S2}

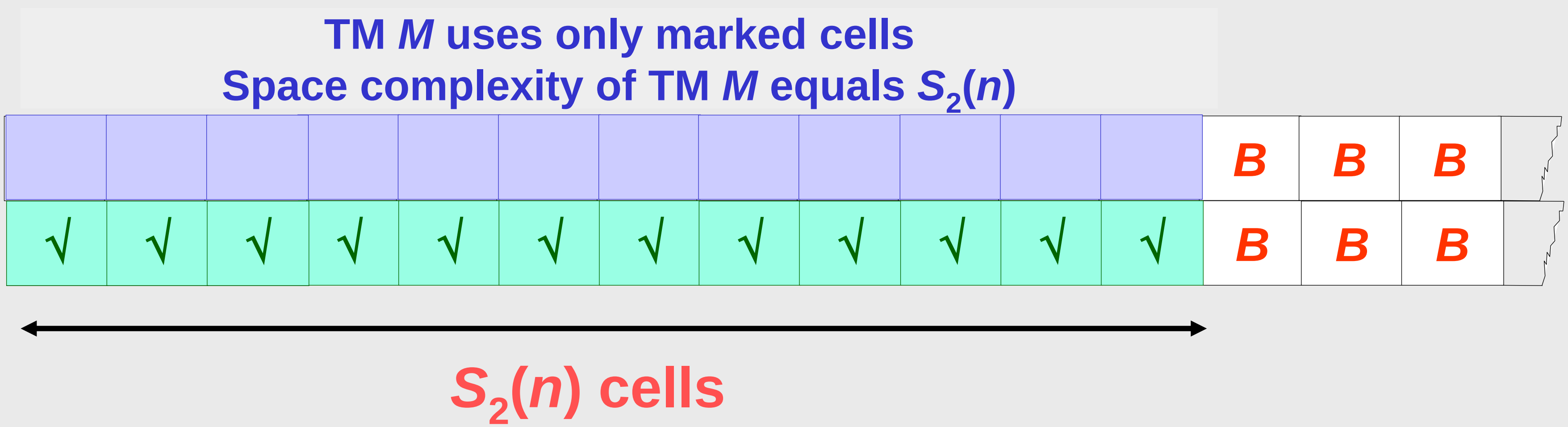


$S_2(n)$ cells

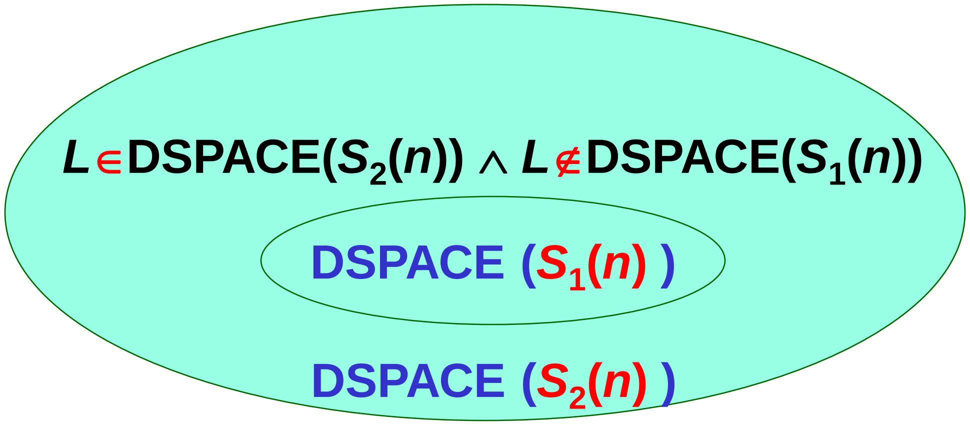


- **TM *M* - ensure a space complexity $S_2(n)$**

TM *M* which accepts the language *L*

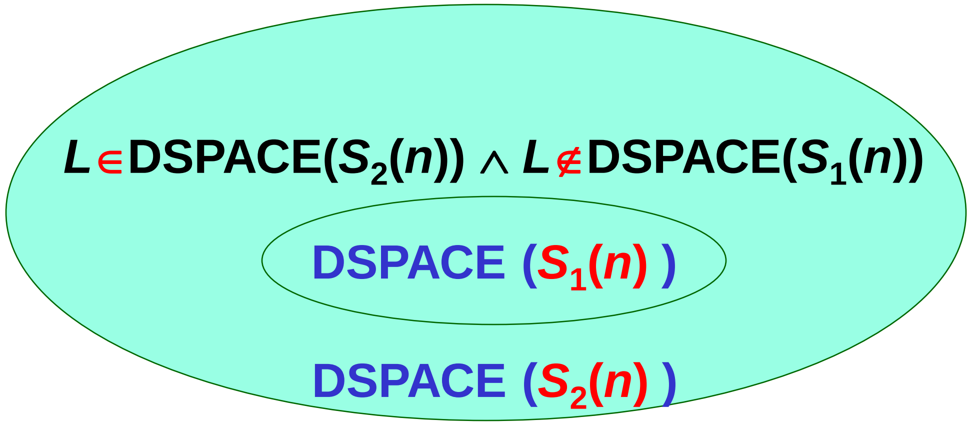


Continuity of the hierarchy for fully space-constructible functions



- **Opposite decision than any TM in class DSPACE($S_1(n)$)**

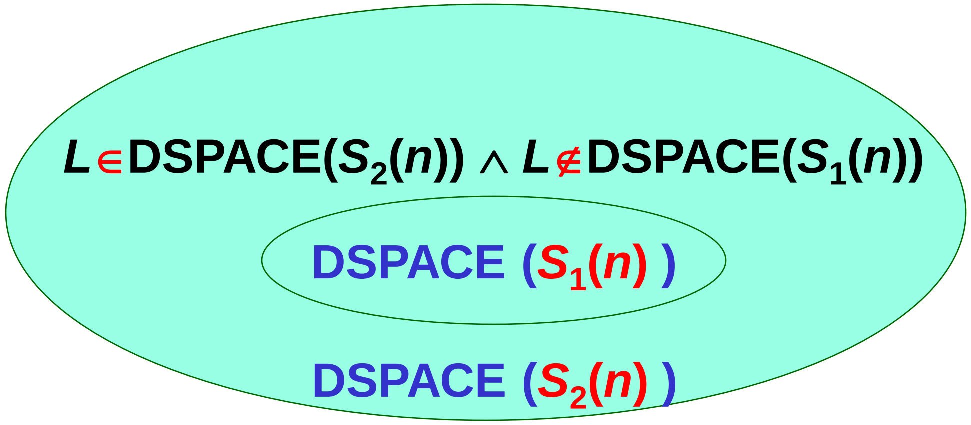
Continuity of the hierarchy for fully space-constructible functions



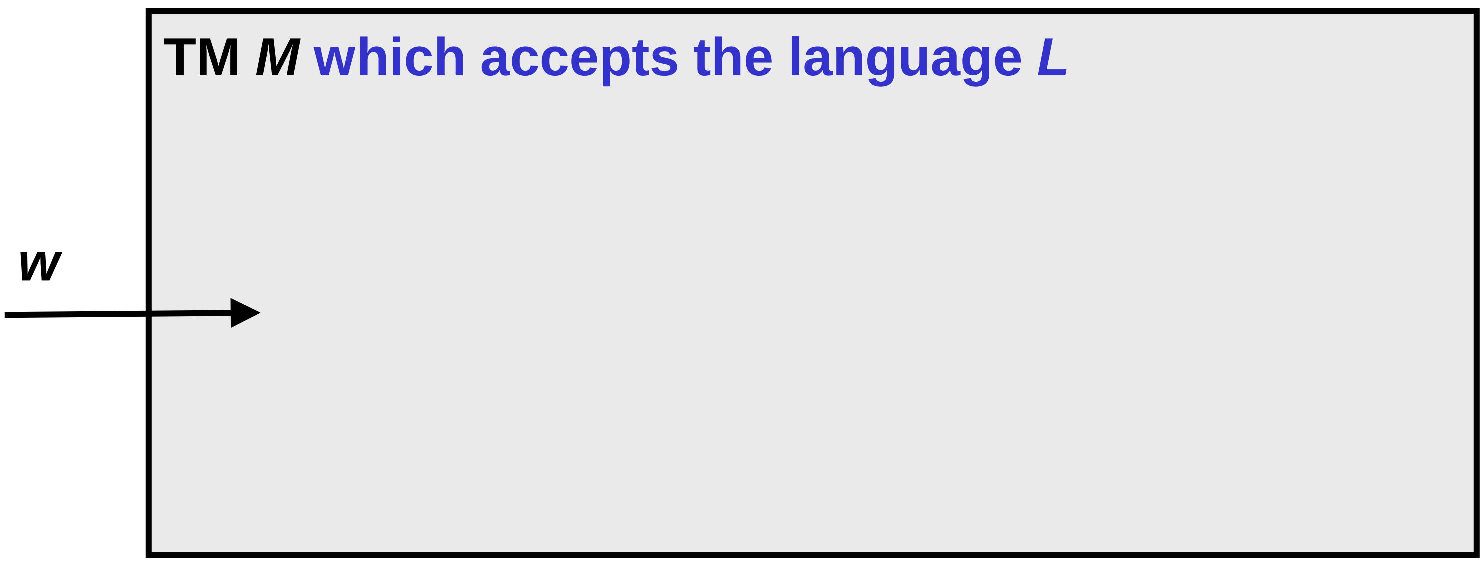
- **Opposite decision than any TM in class DSPACE($S_1(n)$)**

TM *M* which accepts the language *L*

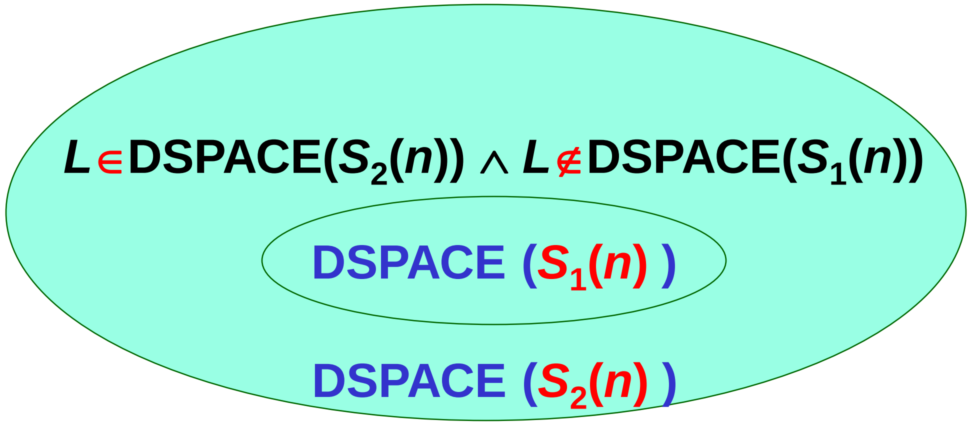
Continuity of the hierarchy for fully space-constructible functions



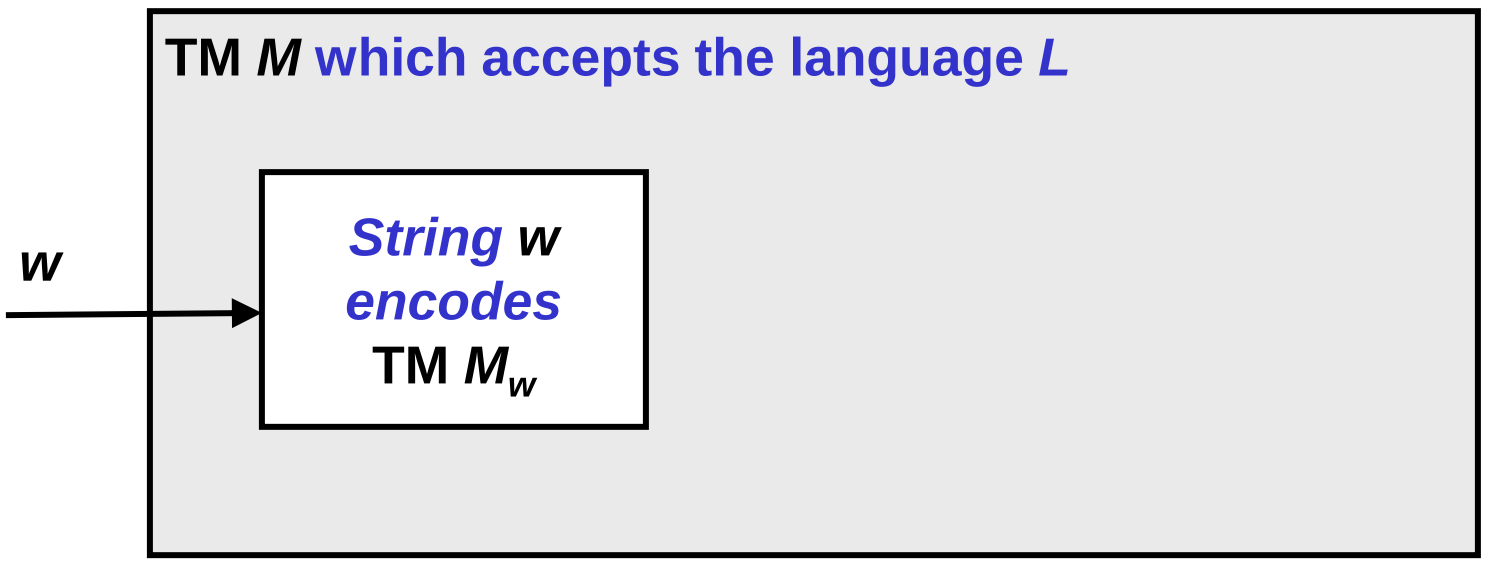
- **Opposite decision than any TM in class DSPACE($S_1(n)$)**



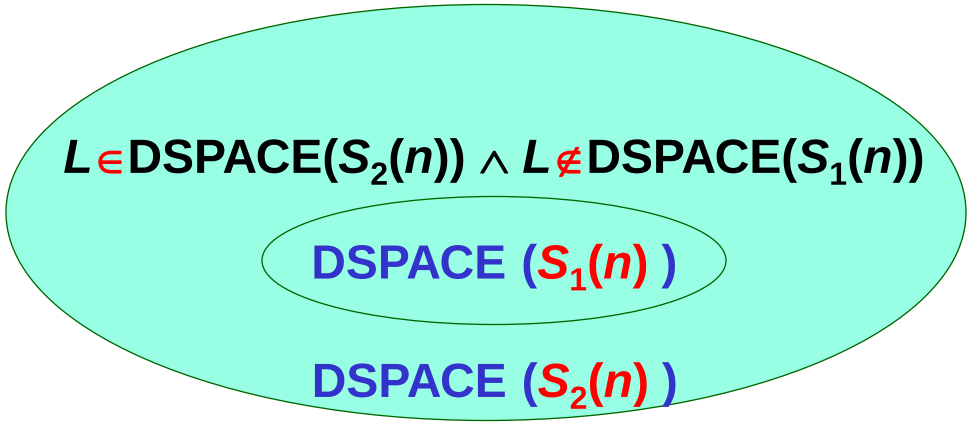
Continuity of the hierarchy for fully space-constructible functions



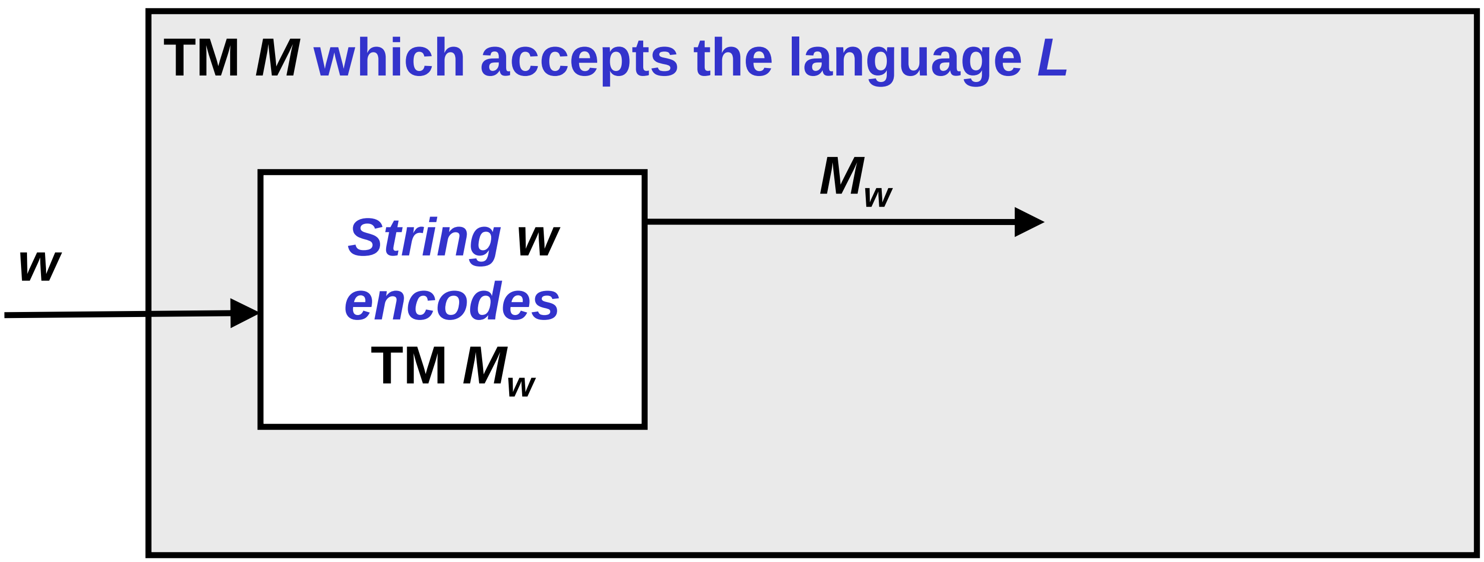
- Opposite decision than any TM in class DSPACE($S_1(n)$)



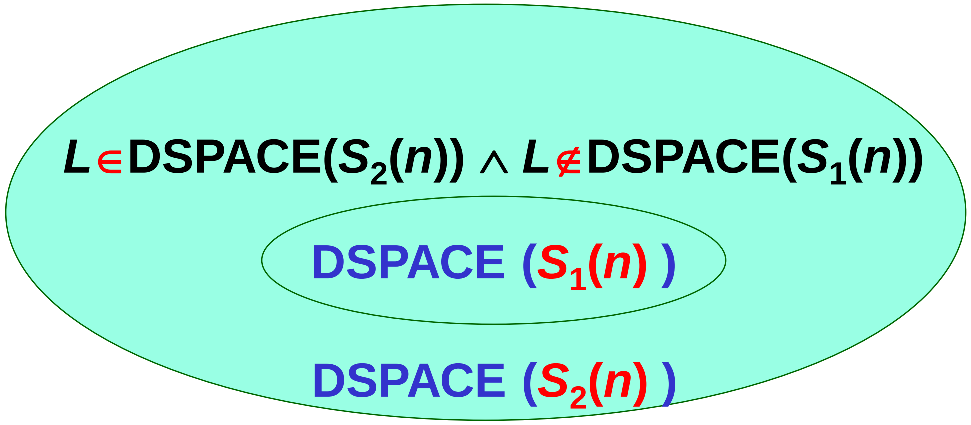
Continuity of the hierarchy for fully space-constructible functions



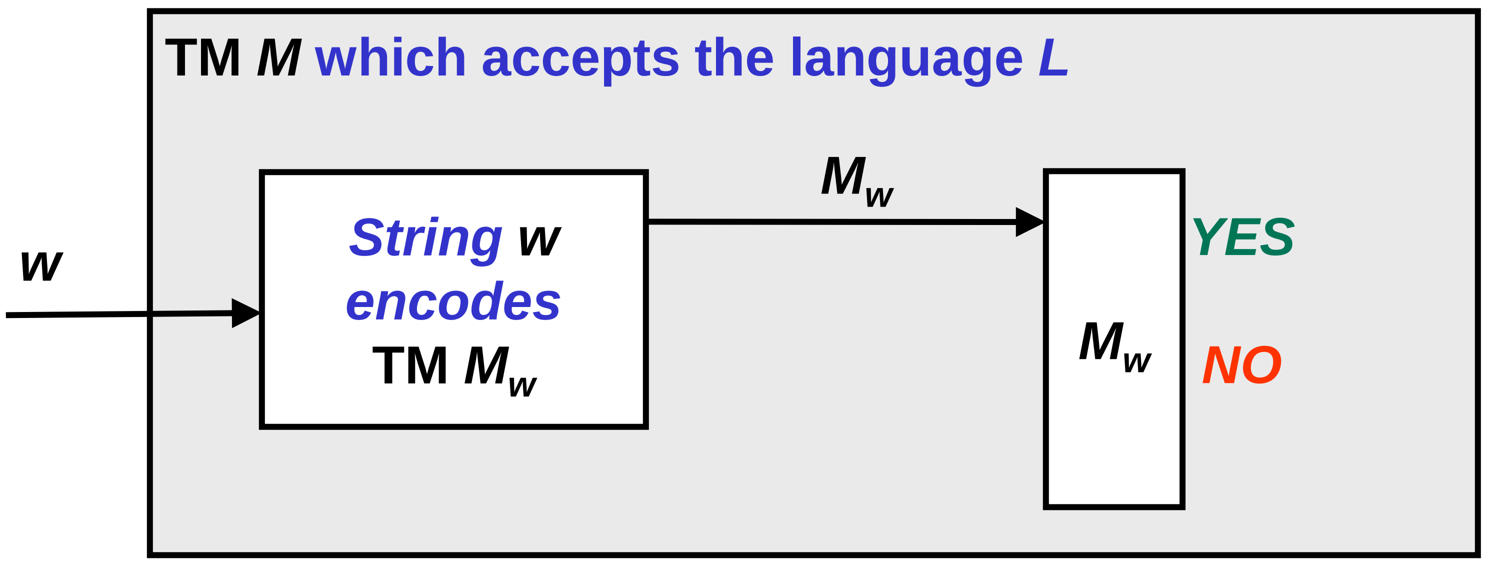
- Opposite decision than any TM in class DSPACE($S_1(n)$)



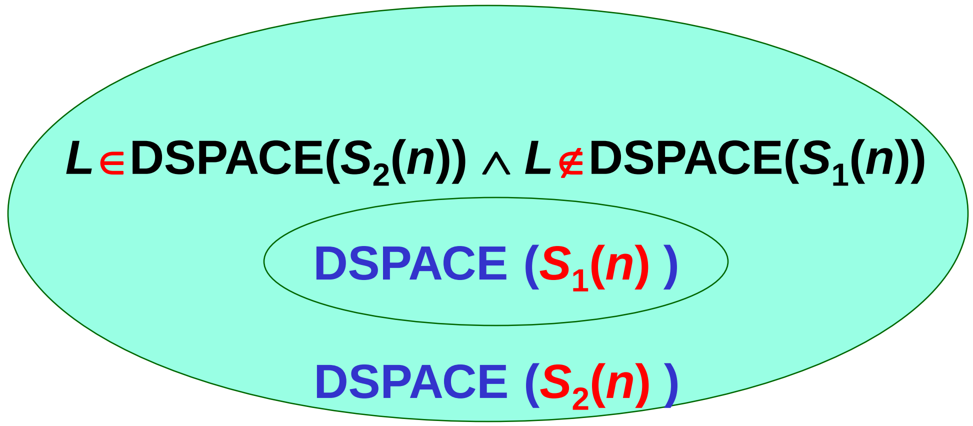
Continuity of the hierarchy for fully space-constructible functions



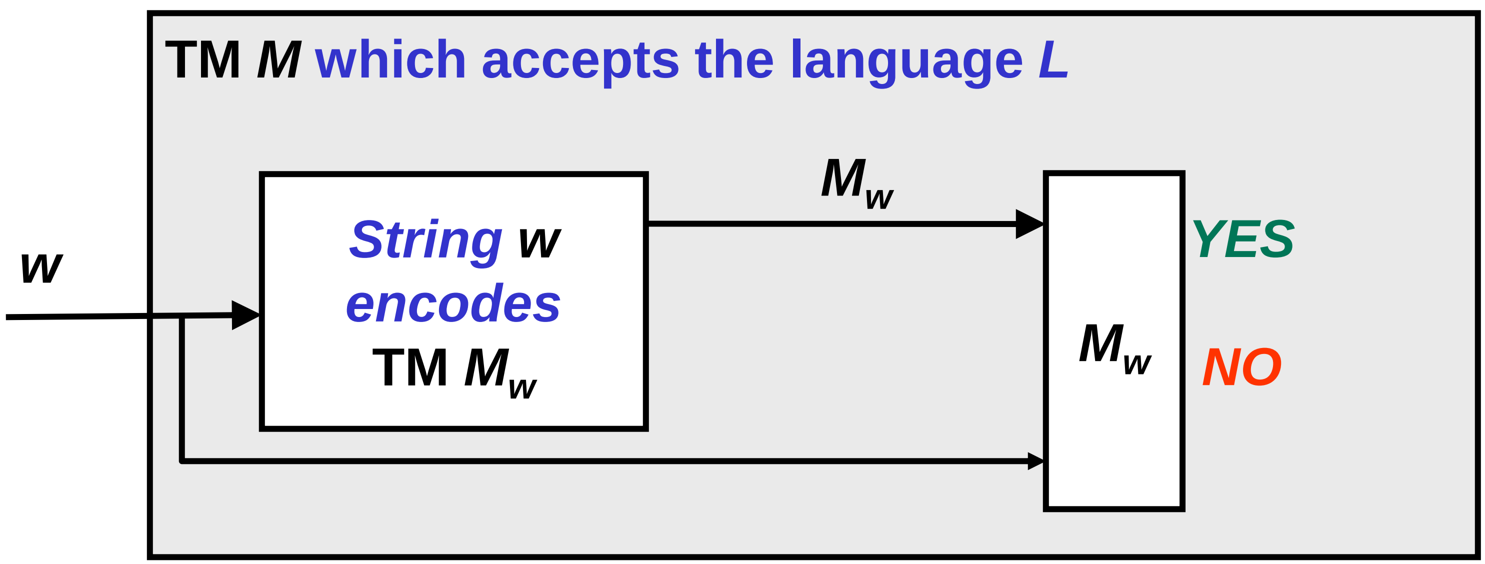
- Opposite decision than any TM in class DSPACE($S_1(n)$)



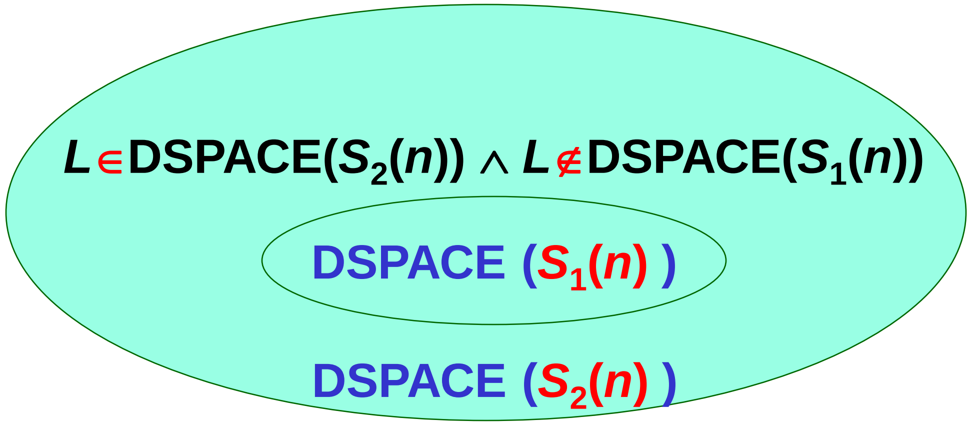
Continuity of the hierarchy for fully space-constructible functions



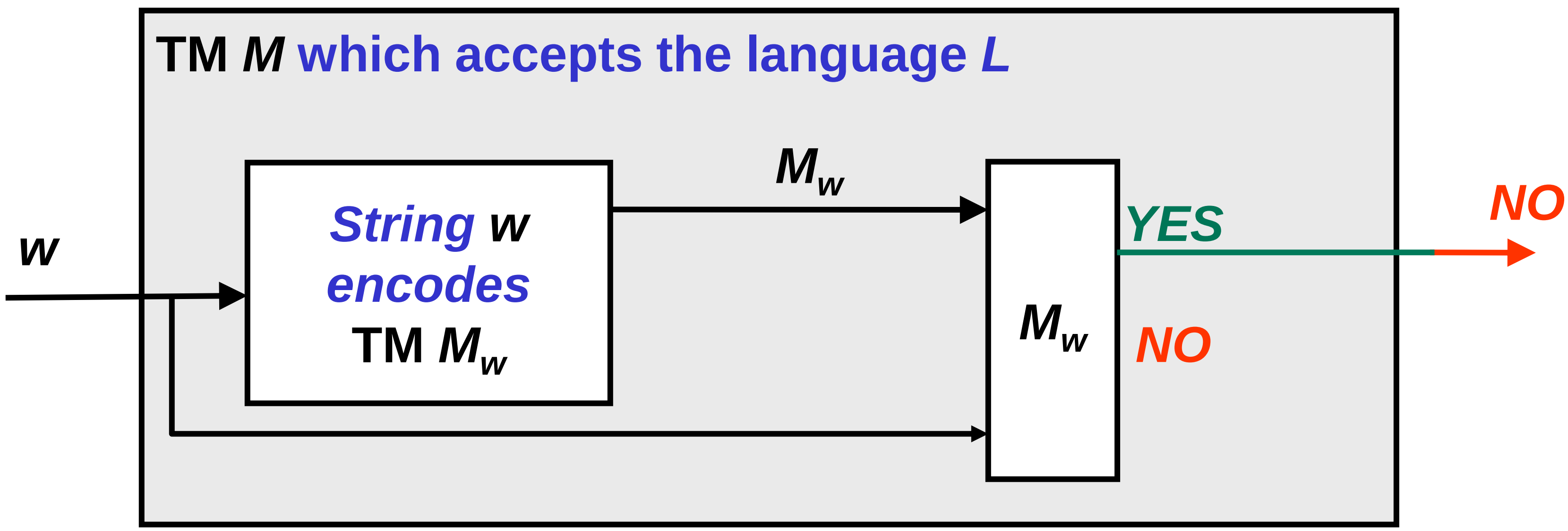
- Opposite decision than any TM in class DSPACE($S_1(n)$)



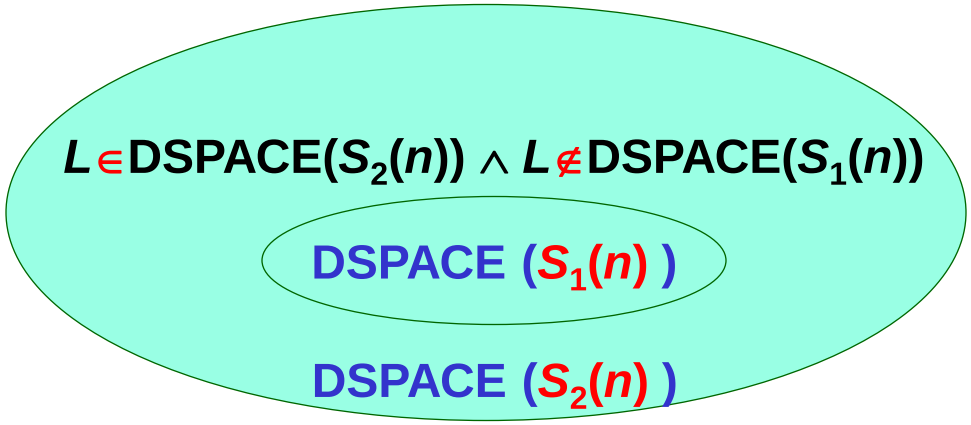
Continuity of the hierarchy for fully space-constructible functions



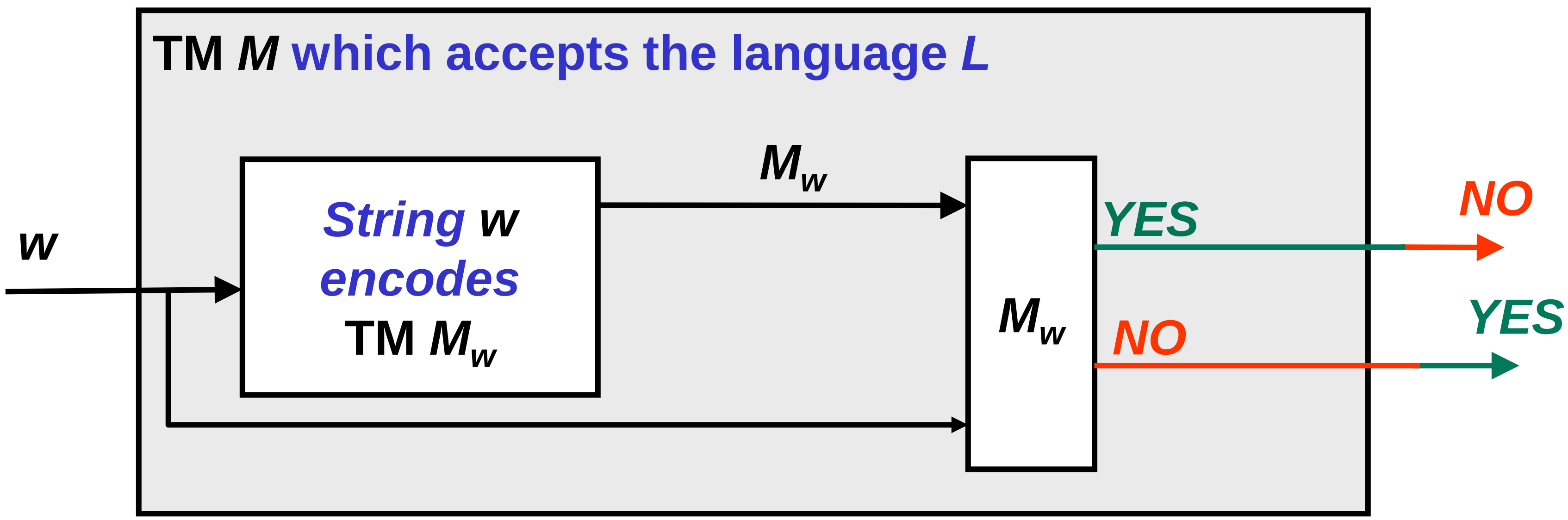
- Opposite decision than any TM in class DSPACE($S_1(n)$)



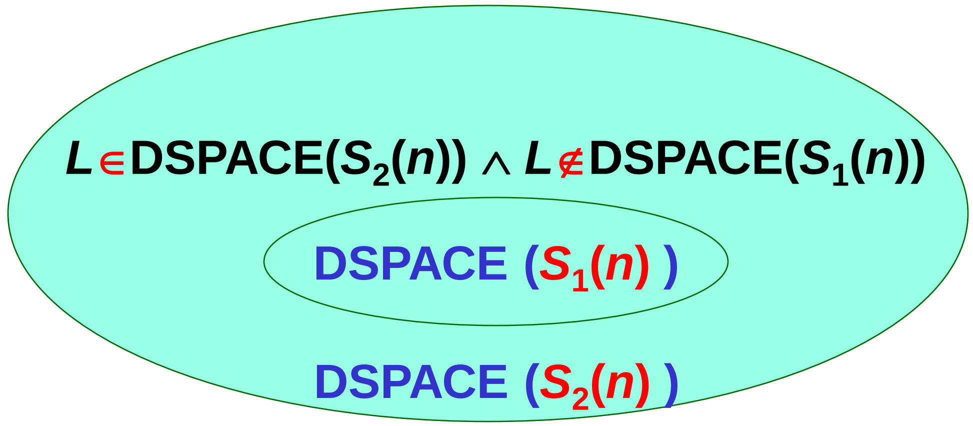
Continuity of the hierarchy for fully space-constructible functions



- Opposite decision than any TM in class DSPACE($S_1(n)$)

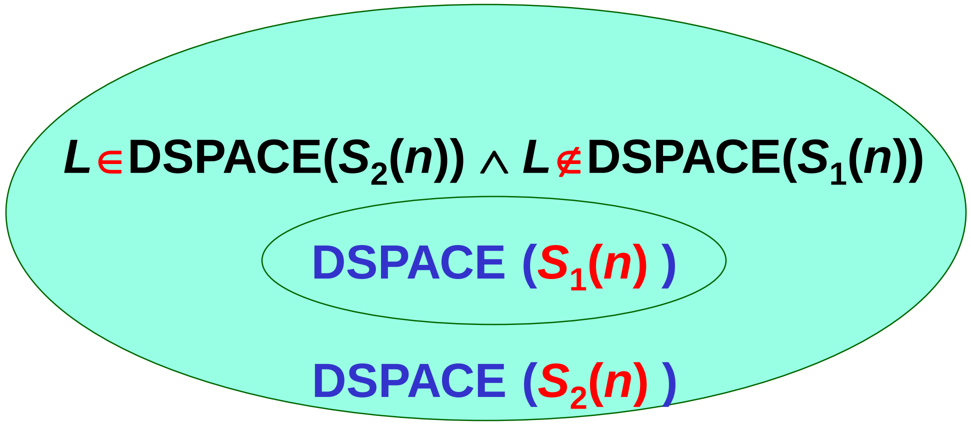


Continuity of the hierarchy for fully space-constructible functions



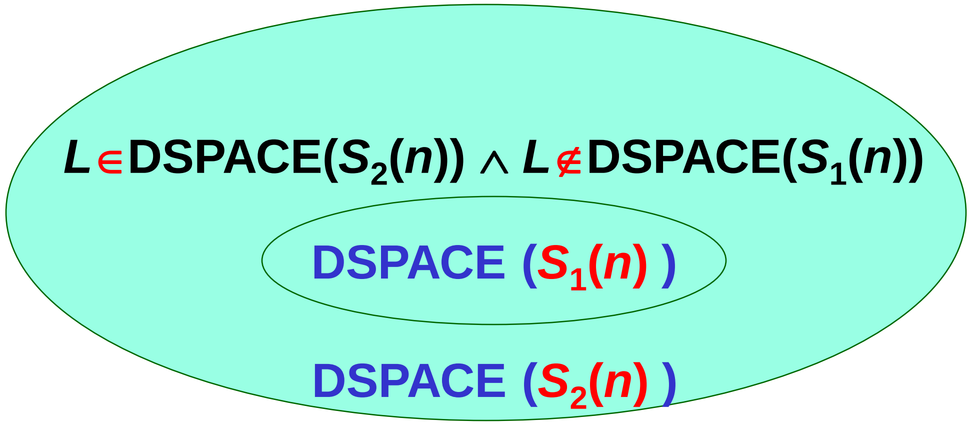
- **Simulating only those TM with space complexity $S_1(n)$**

Continuity of the hierarchy for fully space-constructible functions



- **Simulating only those TM with space complexity $S_1(n)$**
  - **$\mathrm{DSPACE}(\ S_1(n)\ ) \Rightarrow \mathrm{DTIME}(\ c^{S1(n)}\ )$**

Continuity of the hierarchy for fully space-constructible functions

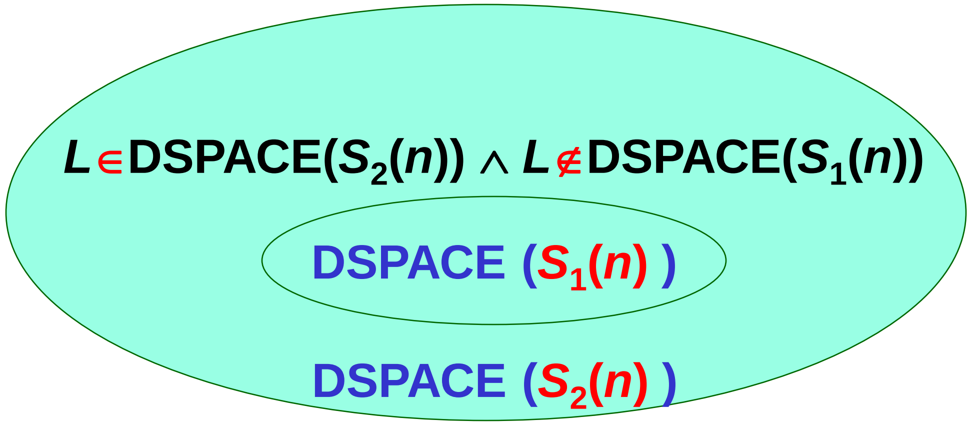


- Simulating only those TM with space complexity $S_1(n)$

- $\mathrm{DSPACE}(\ S_1(n)\ ) \Rightarrow \mathrm{DTIME}(\ c^{S1(n)}\ )$

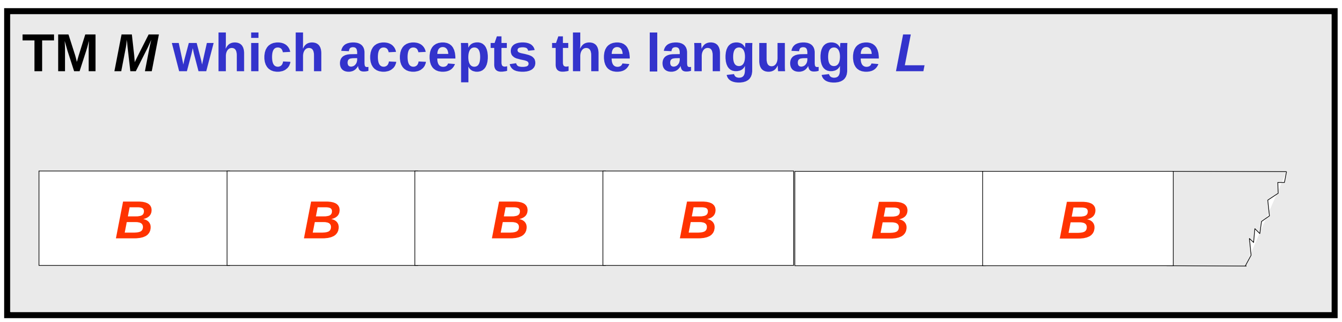
TM *M* which accepts the language *L*

Continuity of the hierarchy for fully space-constructible functions

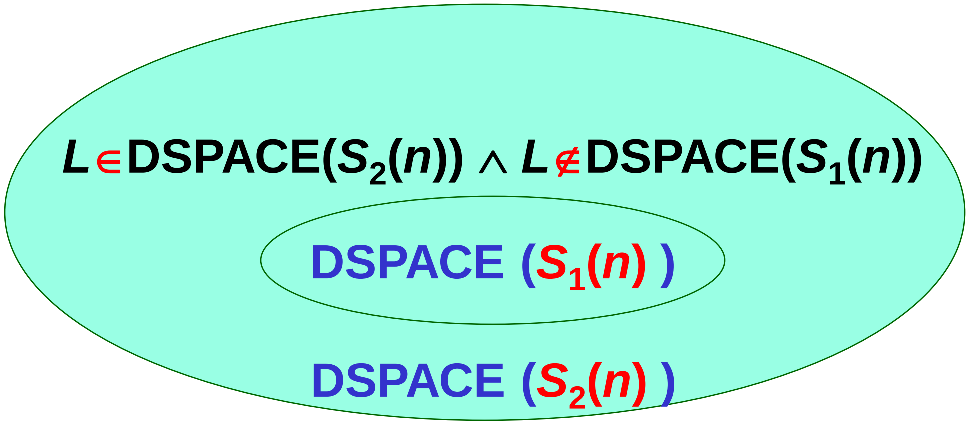


- Simulating only those TM with space complexity $S_1(n)$

- **DSPACE($S_1(n)$) $\Rightarrow$ DTIME($c^{S1(n)}$)**

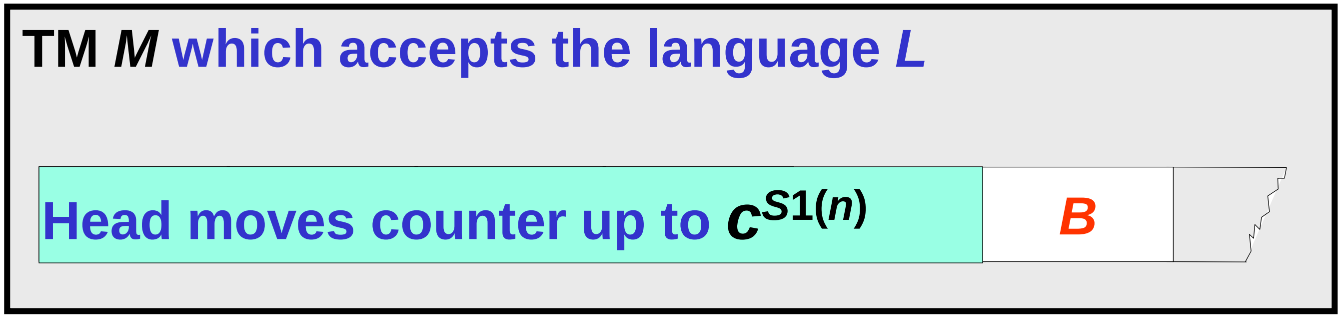


Continuity of the hierarchy for fully space-constructible functions

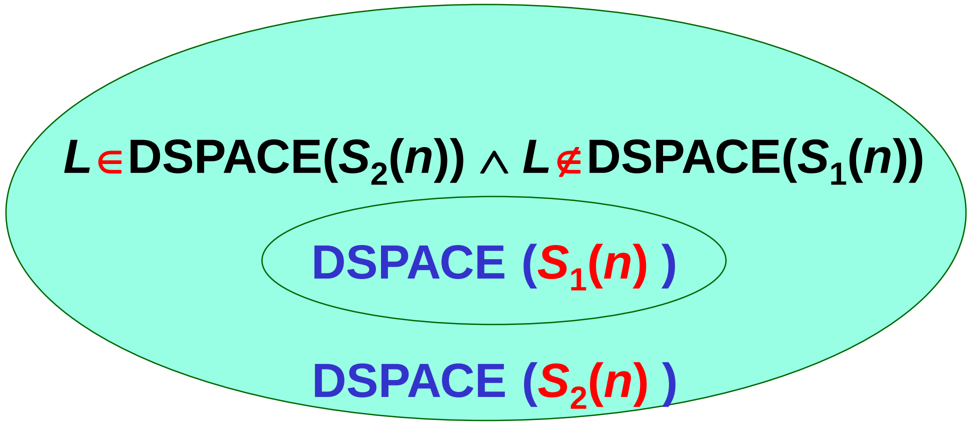


- Simulating only those TM with space complexity $S_1(n)$

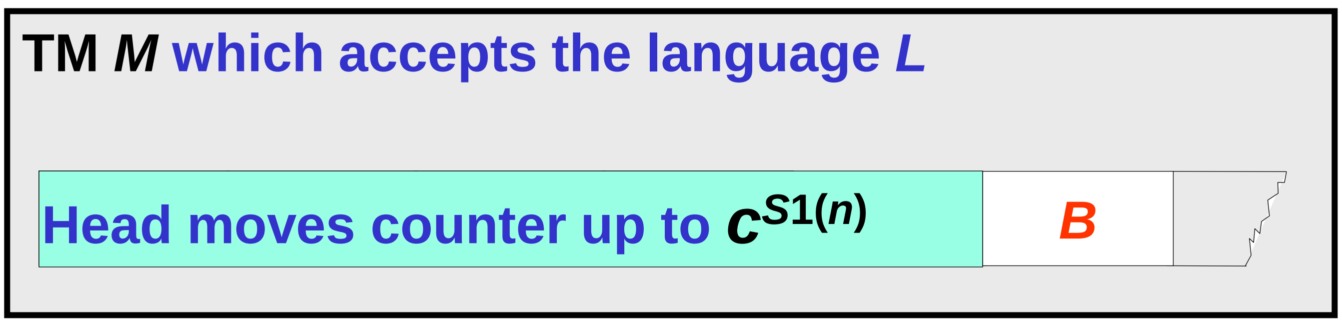
- **DSPACE($S_1(n)$) $\Rightarrow$ DTIME($c^{S1(n)}$)**



Continuity of the hierarchy for fully space-constructible functions

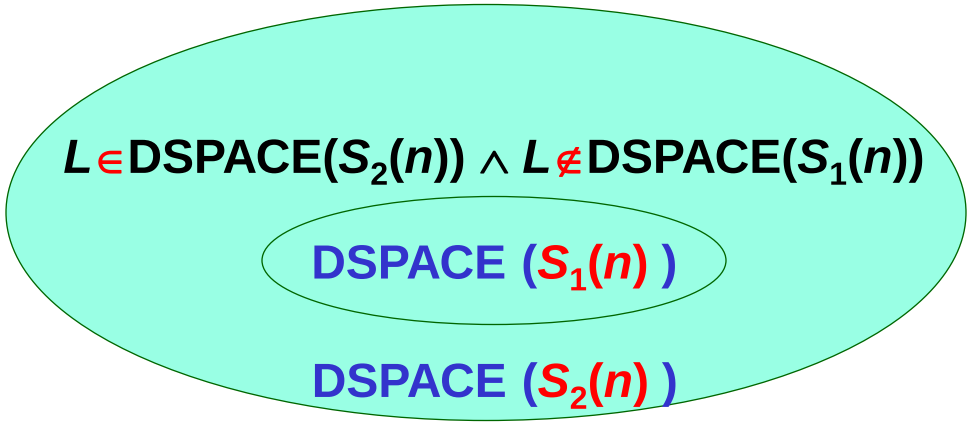


- Simulating only those TM with space complexity $S_1(n)$
- $\mathrm{DSPACE}(\ S_1(n)\ ) \Rightarrow \mathrm{DTIME}(\ c^{S1(n)}\ )$



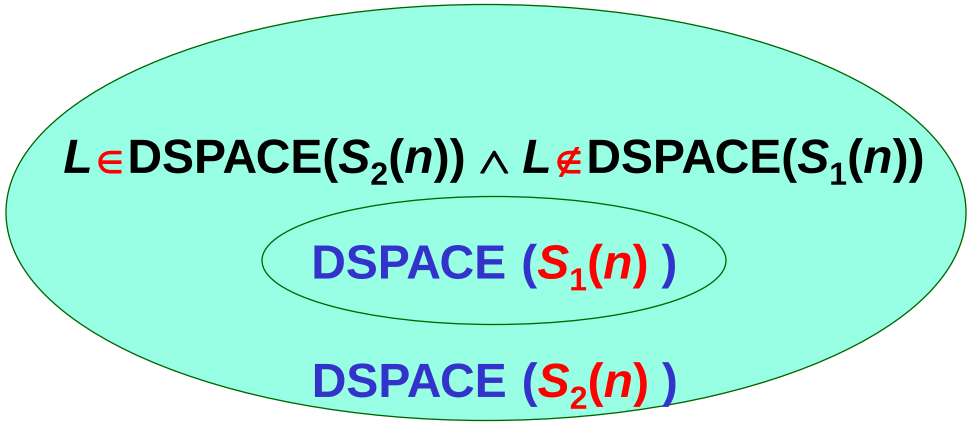
- Choosing the counter number base
  - Counter length < values of functions $S_1(n)$ and $S_2(n)$

Continuity of the hierarchy for fully space-constructible functions



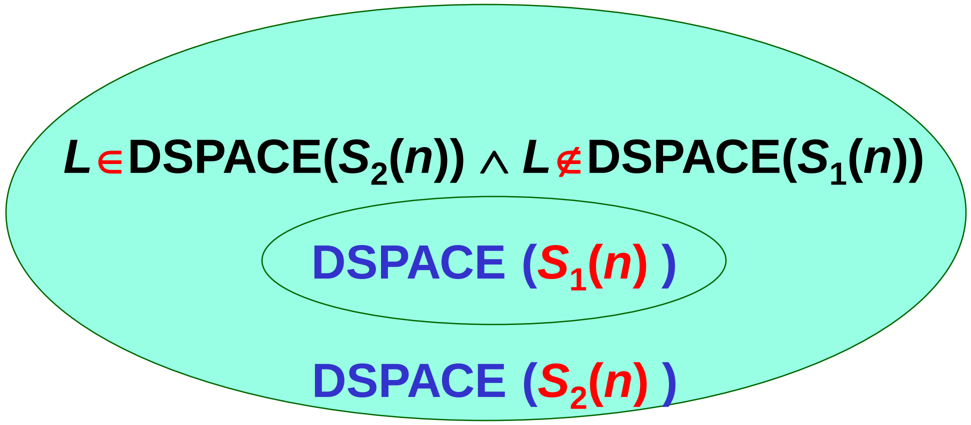
- **Ensure that *M* simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells**

Continuity of the hierarchy for fully space-constructible functions



- **Ensure that *M* simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells**
  - **Number of cells used by simulating a TM from class DSPACE($S_1(n)$) :**

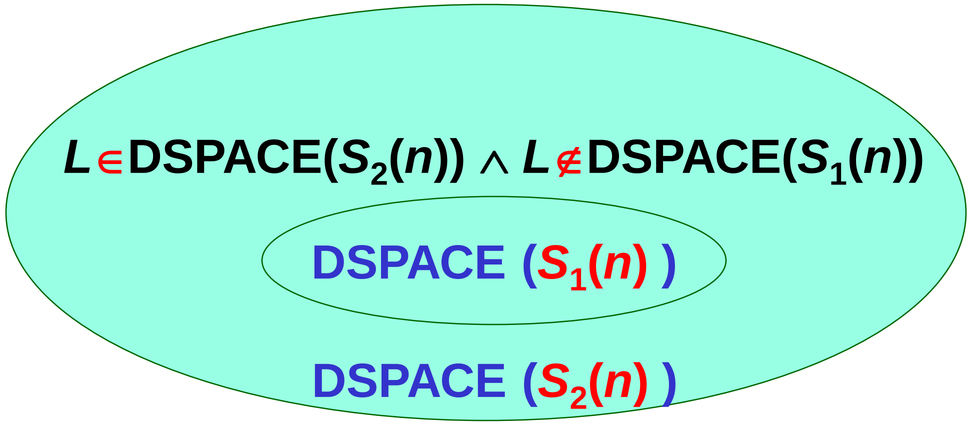
Continuity of the hierarchy for fully space-constructible functions



- **Ensure that *M* simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells**
  - **Number of cells used by simulating a TM from class DSPACE($S_1(n)$) :**

t **– number of tape symbols of a TM from class DSPACE($S_1(n)$)**

Continuity of the hierarchy for fully space-constructible functions

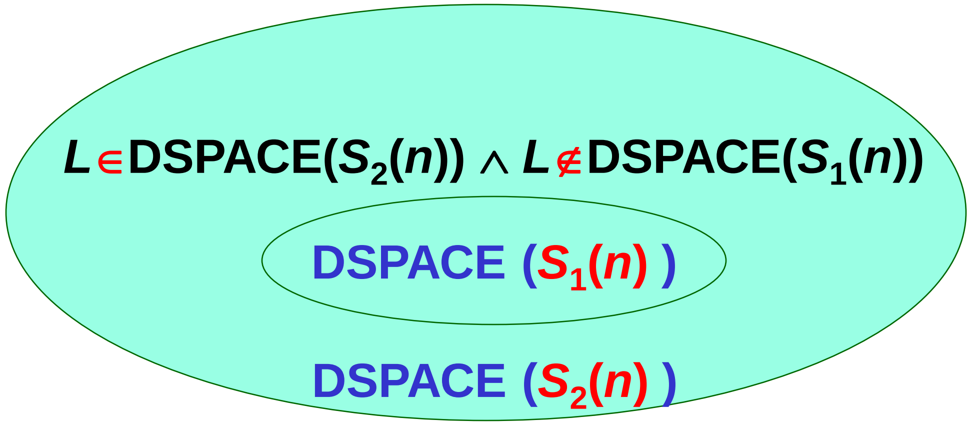


- **Ensure that M simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells**
  - **Number of cells used by simulating a TM from class DSPACE($S_1(n)$) :**

t **– number of tape symbols of a TM from class DSPACE($S_1(n)$)**

{0, 1, *B*} – tape symbols of TM *M*

Continuity of the hierarchy for fully space-constructible functions



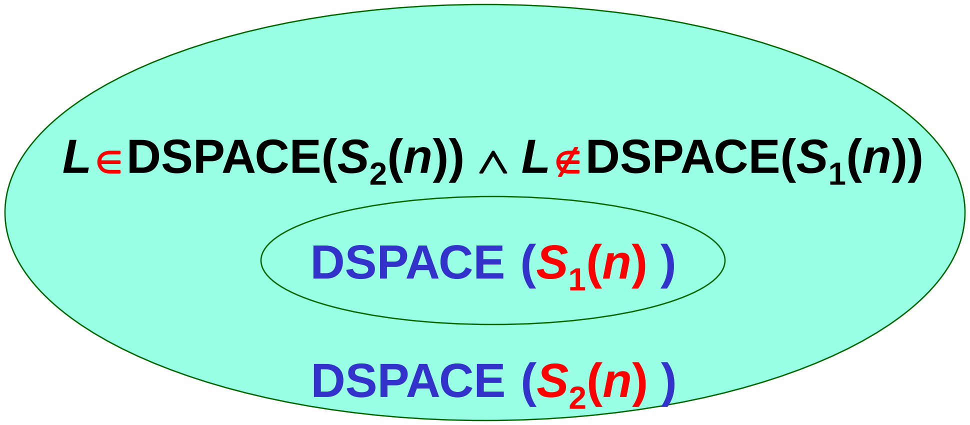
- **Ensure that *M* simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells**
  - **Number of cells used by simulating a TM from class DSPACE($S_1(n)$) :**

$$t$$

t **– number of tape symbols of a TM from class DSPACE($S_1(n)$)**

{0, 1, *B*} – tape symbols of TM *M*

Continuity of the hierarchy for fully space-constructible functions



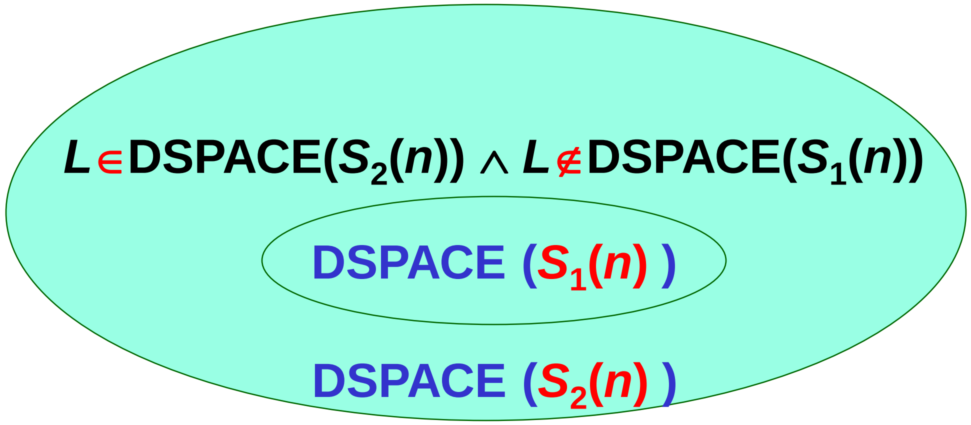
- **Ensure that M simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells**
  - **Number of cells used by simulating a TM from class DSPACE($S_1(n)$) :**

$$\log_2 t$$

t **– number of tape symbols of a TM from class DSPACE($S_1(n)$)**

{0, 1, *B*} – tape symbols of TM *M*

Continuity of the hierarchy for fully space-constructible functions



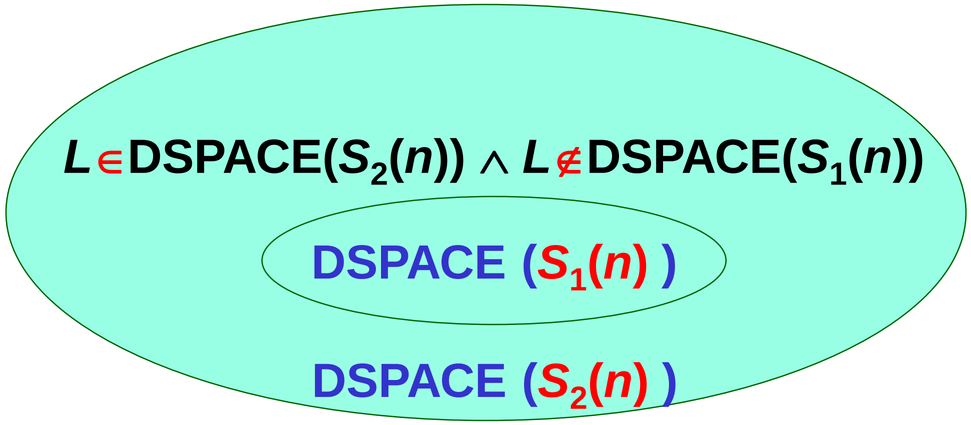
- Ensure that M simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells
  - Number of cells used by simulating a TM from class DSPACE($S_1(n)$) :

$$\lceil \log_2 t \rceil$$

t – number of tape symbols of a TM from class DSPACE($S_1(n)$)

$\{0, 1, B\}$ – tape symbols of TM M

Continuity of the hierarchy for fully space-constructible functions



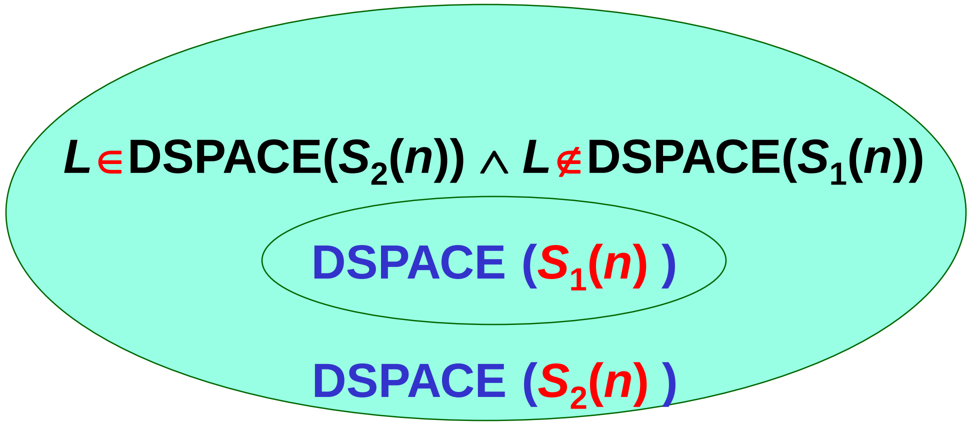
- **Ensure that *M* simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells**
  - **Number of cells used by simulating a TM from class DSPACE($S_1(n)$) :**

$$\lceil \log_2 t \rceil \, S_1(n)$$

t **– number of tape symbols of a TM from class DSPACE($S_1(n)$)**

{0, 1, *B*} – tape symbols of TM *M*

Continuity of the hierarchy for fully space-constructible functions



- **Ensure that *M* simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells**

Continuity of the hierarchy for fully space-constructible functions

$L \in \text{DSPACE}(S_2(n)) \wedge L \notin \text{DSPACE}(S_1(n))$

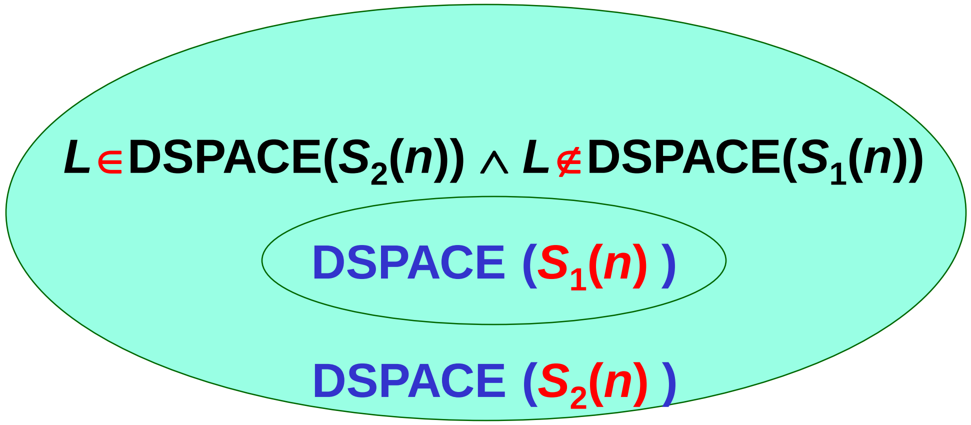
DSPACE ($S_1(n)$)

DSPACE ($S_2(n)$)

- Ensure that M simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells

- $\left(\inf_{n\to\infty} S_1(n)/S_2(n) = 0 \right) \Rightarrow \left(\lceil \log_2 t \rceil\, S_1(n) < S_2(n) \right)$

Continuity of the hierarchy for fully space-constructible functions

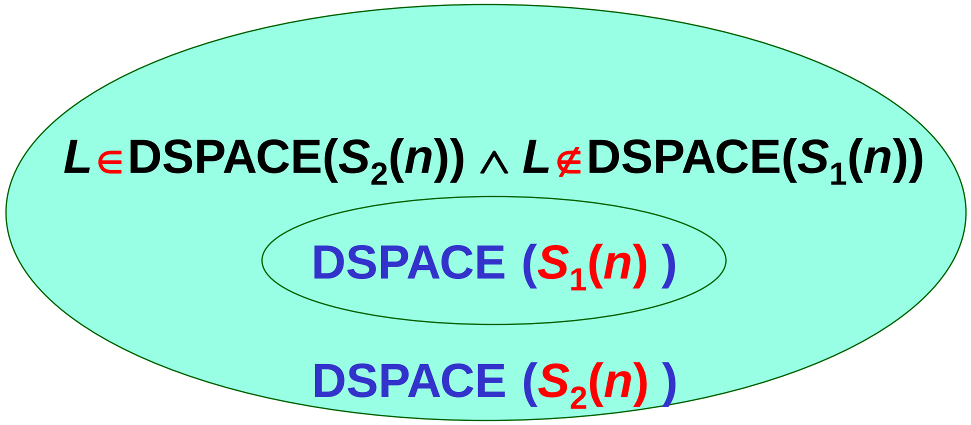


- **Ensure that *M* simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells**

- **($\inf_{n \to \infty} S_1(n)/S_2(n) = 0$) $\Rightarrow$ ($\lceil \log_2 t \rceil S_1(n) < S_2(n)$)**

- **We expand the encoding procedure of TM:**

Continuity of the hierarchy for fully space-constructible functions



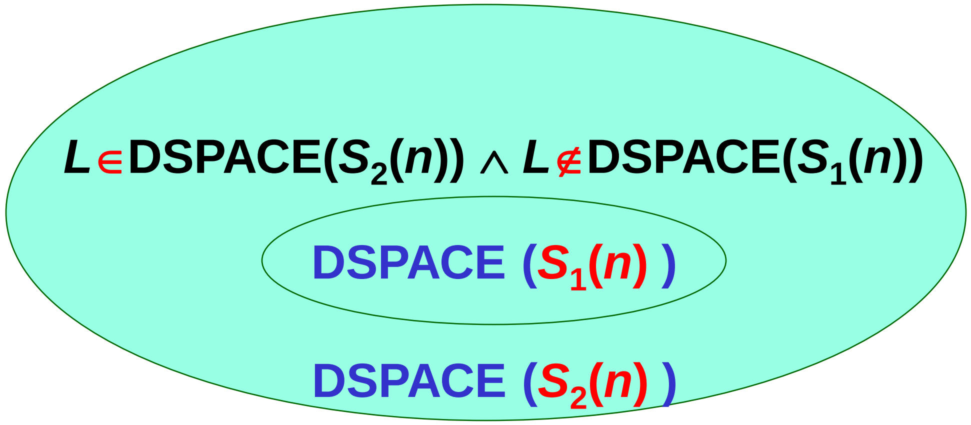
- Ensure that M simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells

  - $\left(\inf_{n\to\infty} S_1(n)/S_2(n) = 0 \right) \Rightarrow \left(\lceil \log_2 t \rceil\, S_1(n) < S_2(n) \right)$

  - We expand the encoding procedure of TM:

$$111 \;\; code_1 \;\; 11 \;\; code_2 \;\; 11 \; \text{- - -} \; 11 \;\; code_r \;\; 111$$

Continuity of the hierarchy for fully space-constructible functions



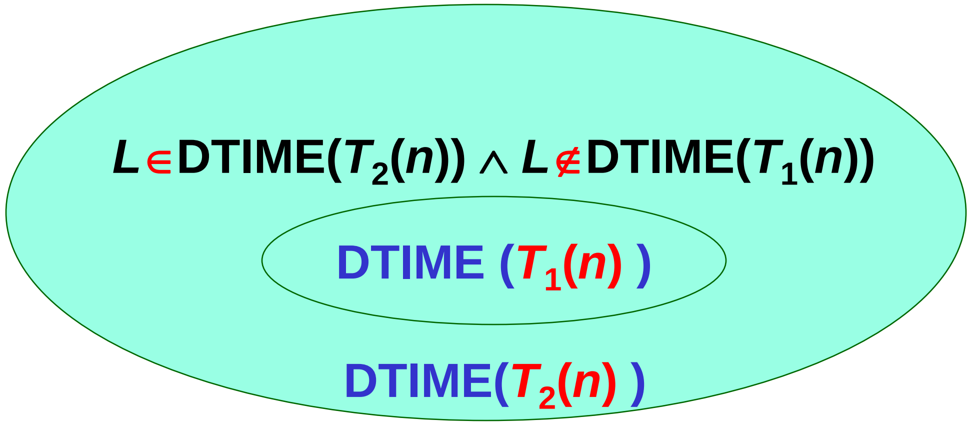
- Ensure that M simulates M_w from DSPACE($S_1(n)$) in $S_2(n)$ cells

  - $(\ \inf_{n\to\infty} S_1(n)/S_2(n) = 0\ ) \Rightarrow (\ \lceil \log_2 t \rceil\, S_1(n) < S_2(n)\ )$

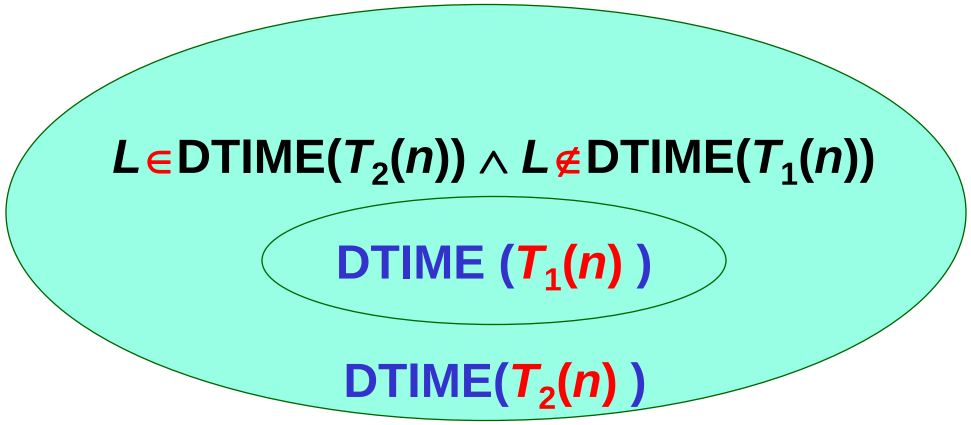
  - **We expand the encoding procedure of TM:**

111111111111111111111 *code*$_1$ **11** *code*$_2$ **11 - - - 11** *code*$_r$ **111**

Continuity of the hierarchy for fully time-constructible functions

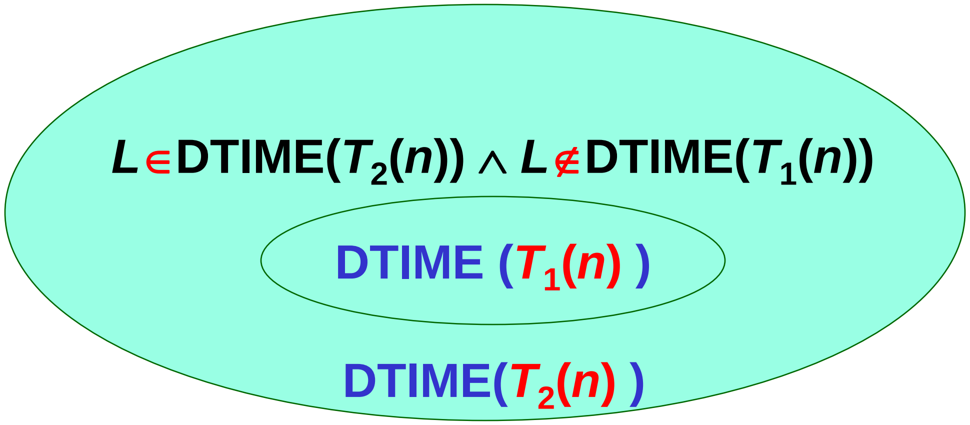


Continuity of the hierarchy for fully time-constructible functions



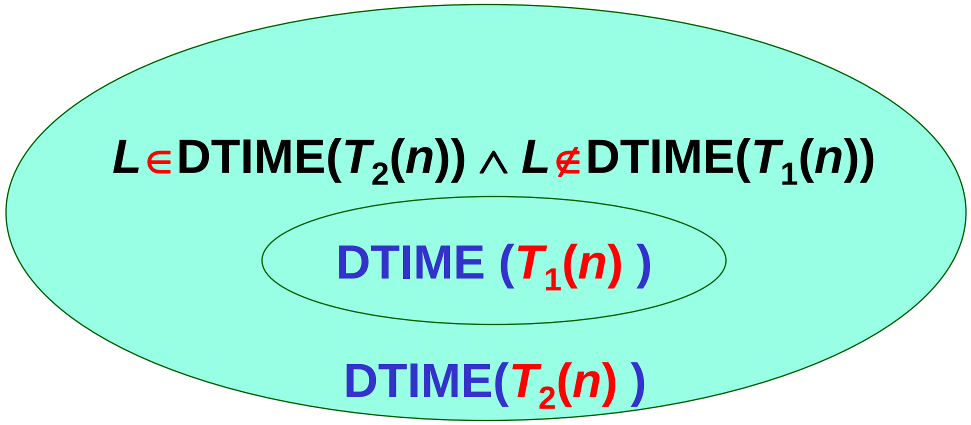
- If

Continuity of the hierarchy for fully time-constructible functions



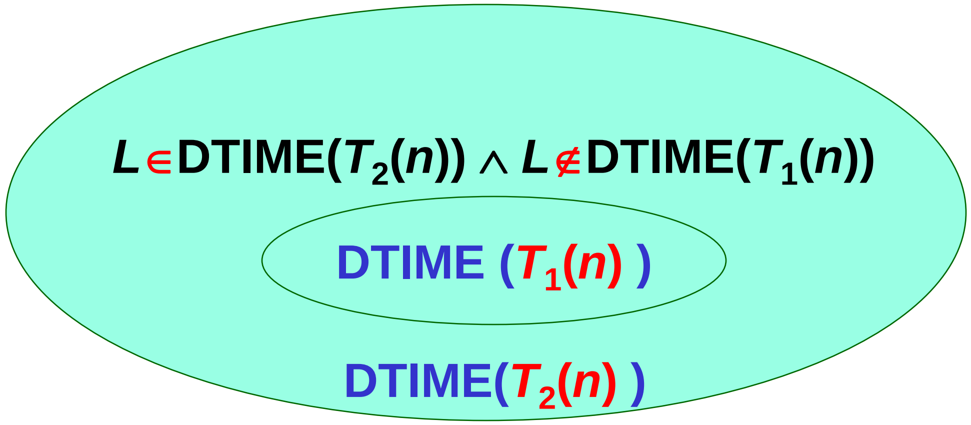
- **If**
  - **$T_2(n)$ is a fully time-constructible function**

Continuity of the hierarchy for fully time-constructible functions



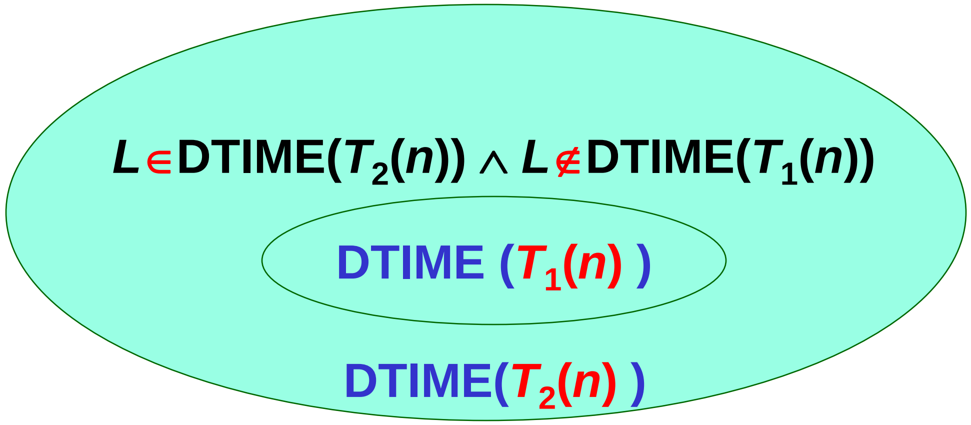
- If
  - $T_2(n)$ is a fully time-constructible function
  - $\inf_{n\to\infty} T_1(n) \log T_1(n) / T_2(n) = 0$

Continuity of the hierarchy for fully time-constructible functions



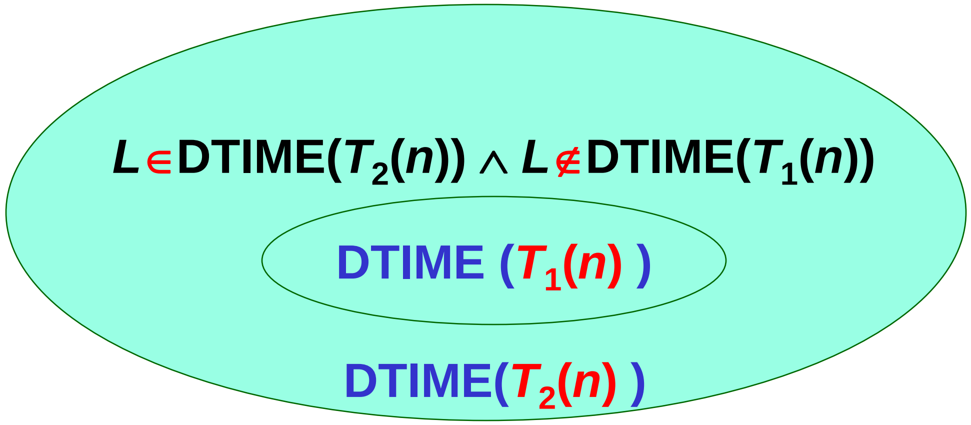
- **If**
  - **$T_2(n)$ is a fully time-constructible function**
  - **$\inf_{n\to\infty} T_1(n) \log T_1(n)/T_2(n) = 0$**
- **Then there is a language *L***

Continuity of the hierarchy for fully time-constructible functions

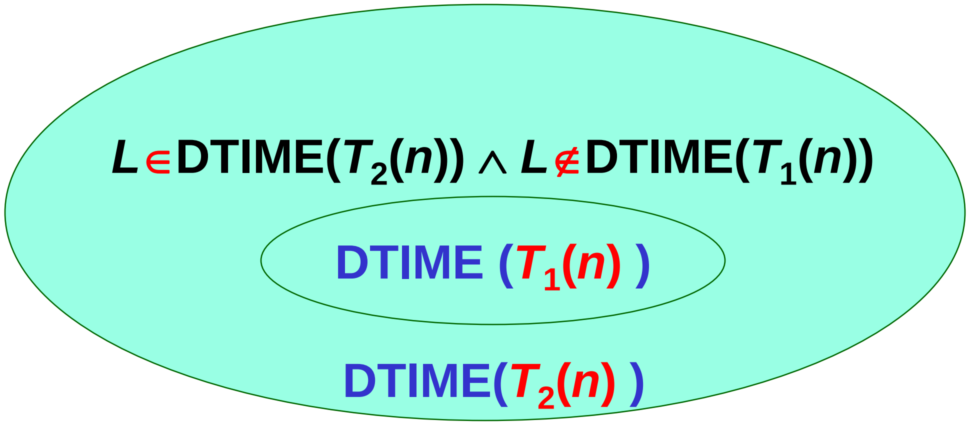


- **If**
  - $T_2(n)$ **is a fully time-constructible function**
  - $\inf_{n\to\infty} T_1(n) \log T_1(n) / T_2(n) = 0$
- **Then there is a language** L
  - $L \in \mathrm{DTIME}(T_2(n)) \wedge L \notin \mathrm{DTIME}(T_1(n))$

Continuity of the hierarchy for fully time-constructible functions

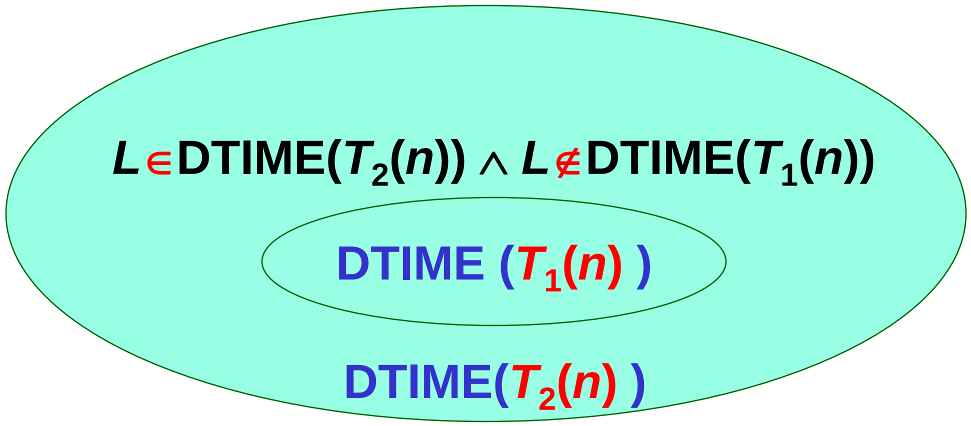


Continuity of the hierarchy for fully time-constructible functions



- **We construct a TM *M* for which:**

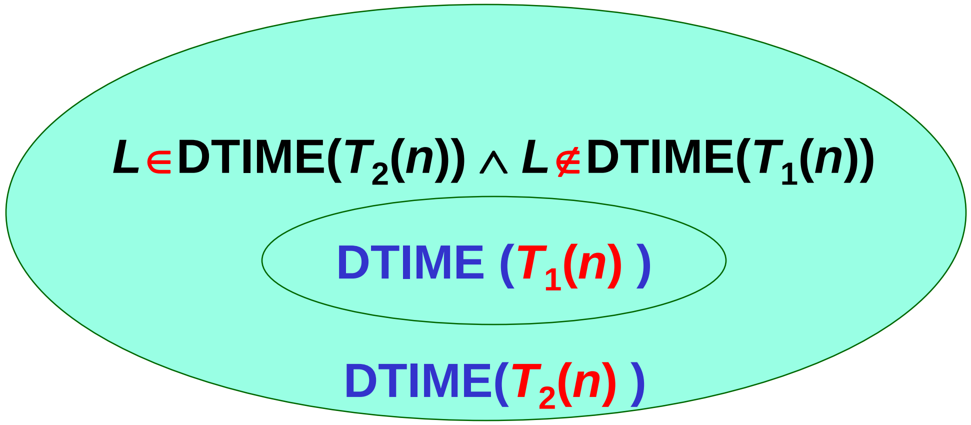
Continuity of the hierarchy for fully time-constructible functions



- **We construct a TM *M* for which:**

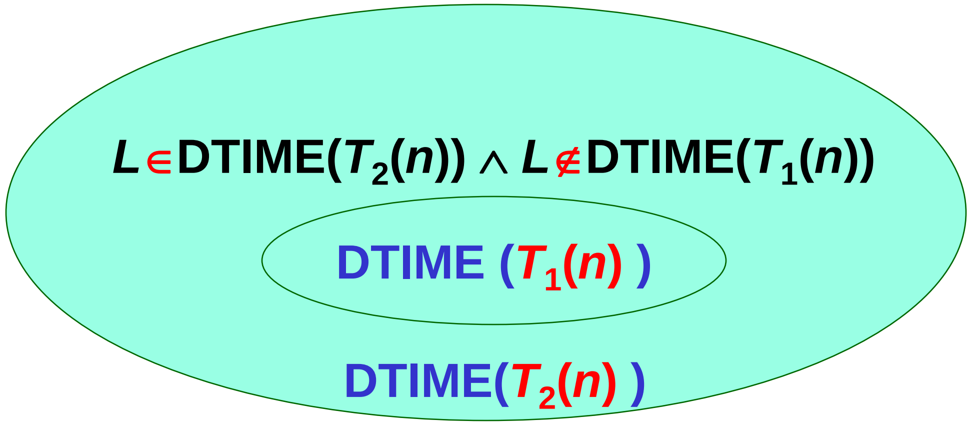
  - **TM *M* has time complexity $T_2(n)$**

Continuity of the hierarchy for fully time-constructible functions

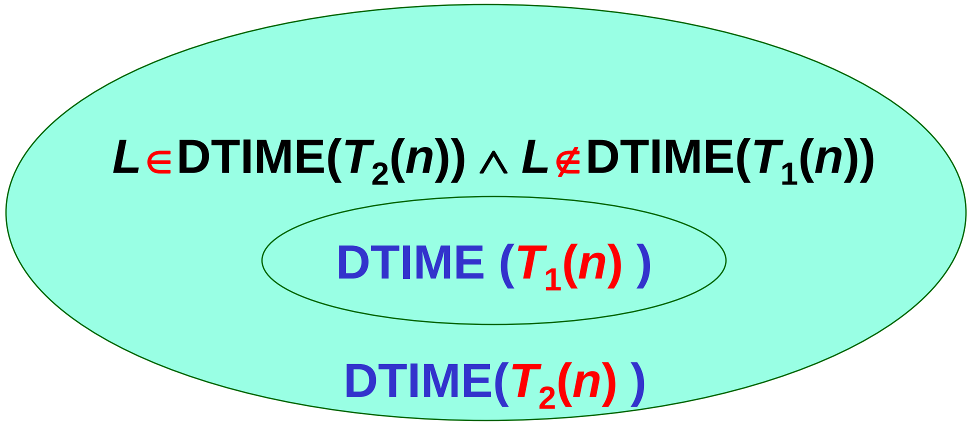


- **We construct a TM *M* for which:**
  - **TM *M* has time complexity $T_2(n)$**
  - **TM *M* gives the opposite decision than any TM with time complexity $T_1(n)$ for at least one input string**

Continuity of the hierarchy for fully time-constructible functions

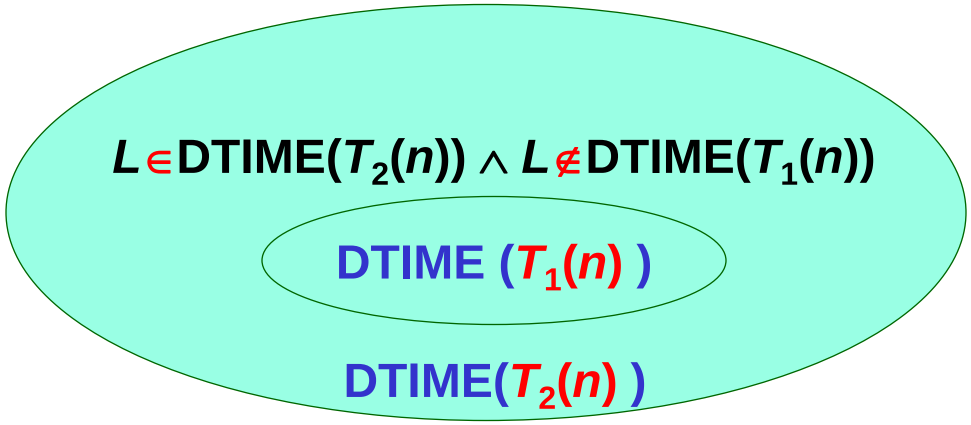


Continuity of the hierarchy for fully time-constructible functions



- **TM *M* – ensure a time complexity of $T_2(n)$**

Continuity of the hierarchy for fully time-constructible functions



- **TM *M* – ensure a time complexity of $T_2(n)$**
  - **Parallel simulation**

Continuity of the hierarchy for fully time-constructible functions

$L \in \mathrm{DTIME}(T_2(n)) \wedge L \notin \mathrm{DTIME}(T_1(n))$

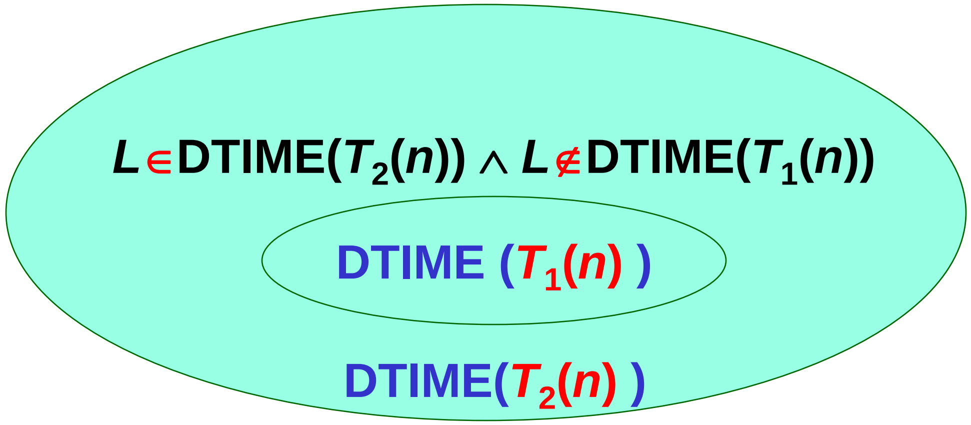
DTIME $(T_1(n))$

DTIME$(T_2(n))$

- TM M – ensure a time complexity of $T_2(n)$

  - **Parallel simulation**
    - — work of TM M_w for an input string w

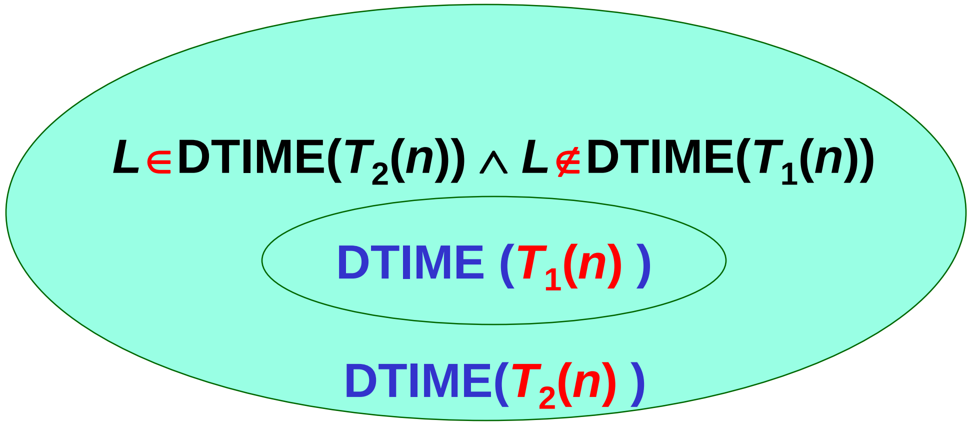
Continuity of the hierarchy for fully time-constructible functions



- TM M – ensure a time complexity of $T_2(n)$

  - **Parallel simulation**
    - — work of TM M_w for an input string w
    - — work of any TM M_{T2} with time complexity $T_2(n)$

Continuity of the hierarchy for fully time-constructible functions

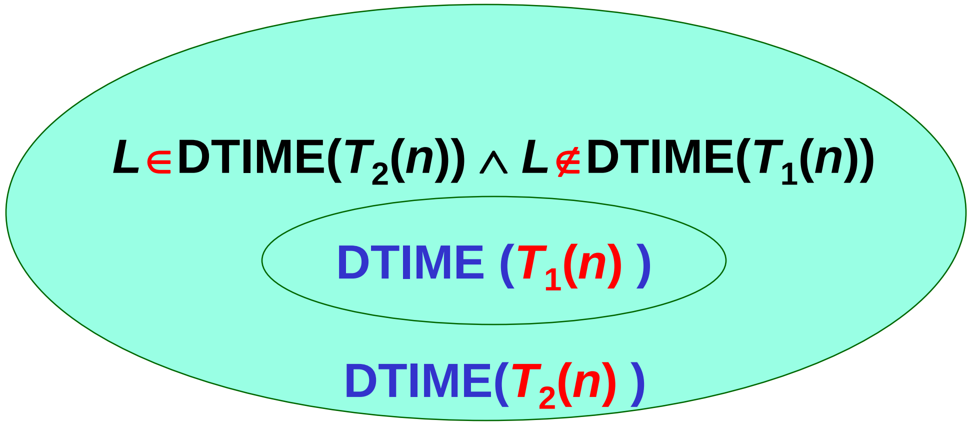


- TM M – ensure a time complexity of $T_2(n)$

  - **Parallel simulation**
    - work of TM M_w for an input string w
    - work of any TM M_{T2} with time complexity $T_2(n)$

  - **$T_2(n)$ is fully time-constructible**

Continuity of the hierarchy for fully time-constructible functions



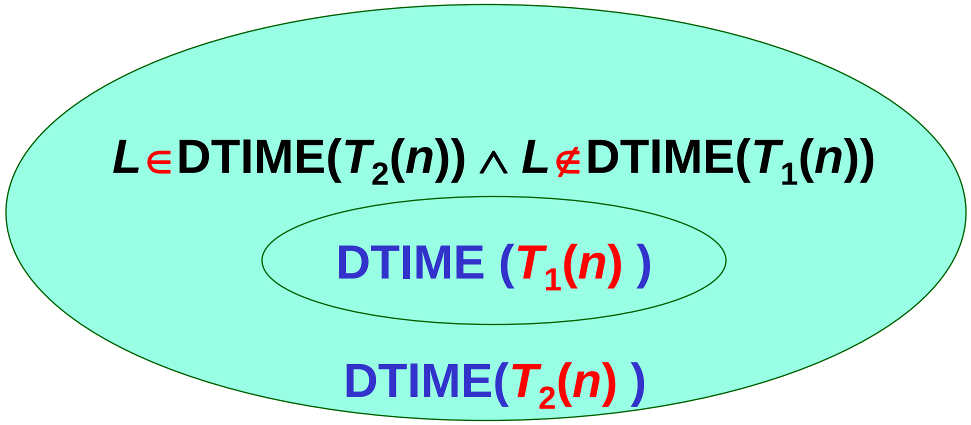
- **TM M – ensure a time complexity of $T_2(n)$**

  - **Parallel simulation**
    - — **work of TM M_w for an input string w**
    - — **work of any TM M_{T2} with time complexity $T_2(n)$**

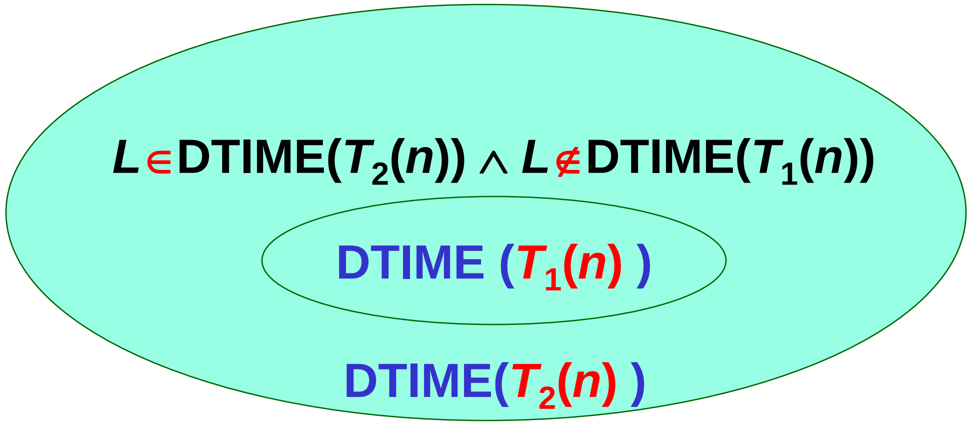
  - **$T_2(n)$ is fully time-constructible**

    $\Rightarrow$ **TM M_{T2} for any string of length n does $T_2(n)$ moves**

Continuity of the hierarchy for fully time-constructible functions

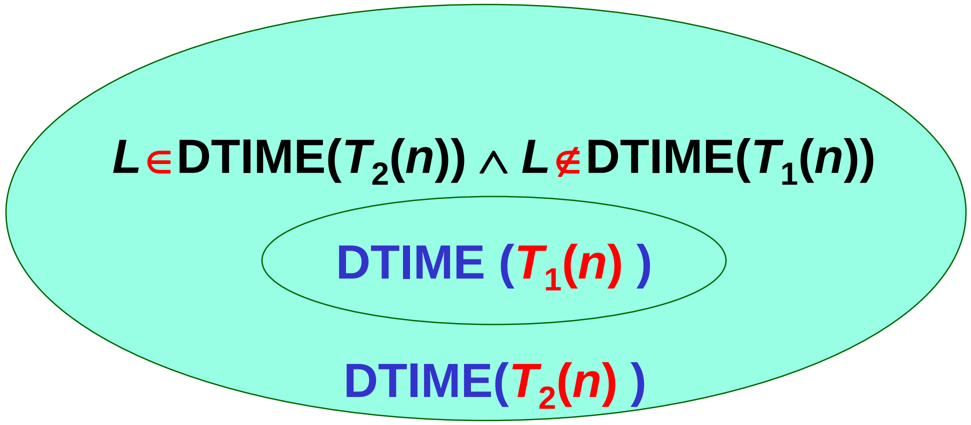


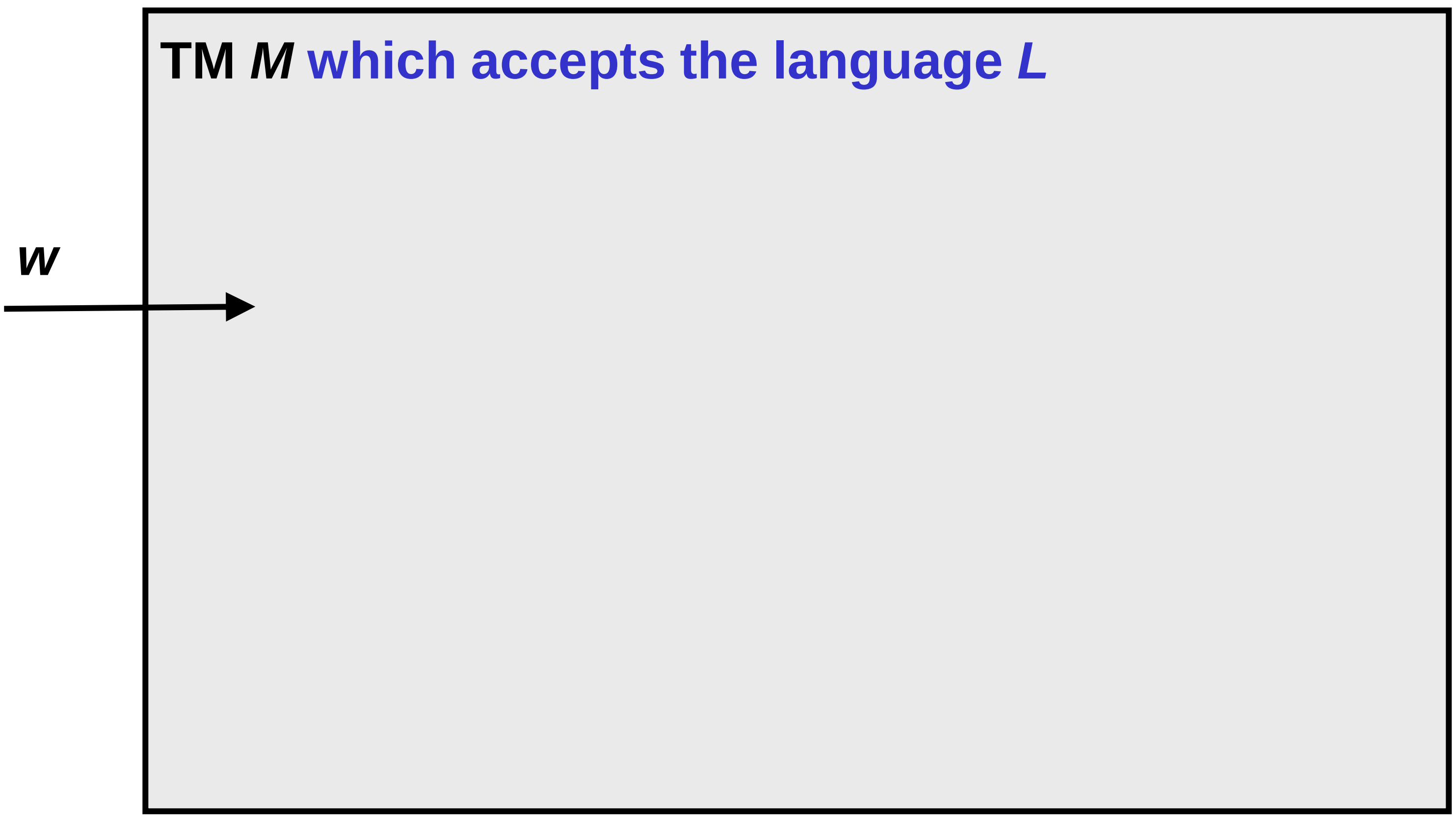
Continuity of the hierarchy for fully time-constructible functions



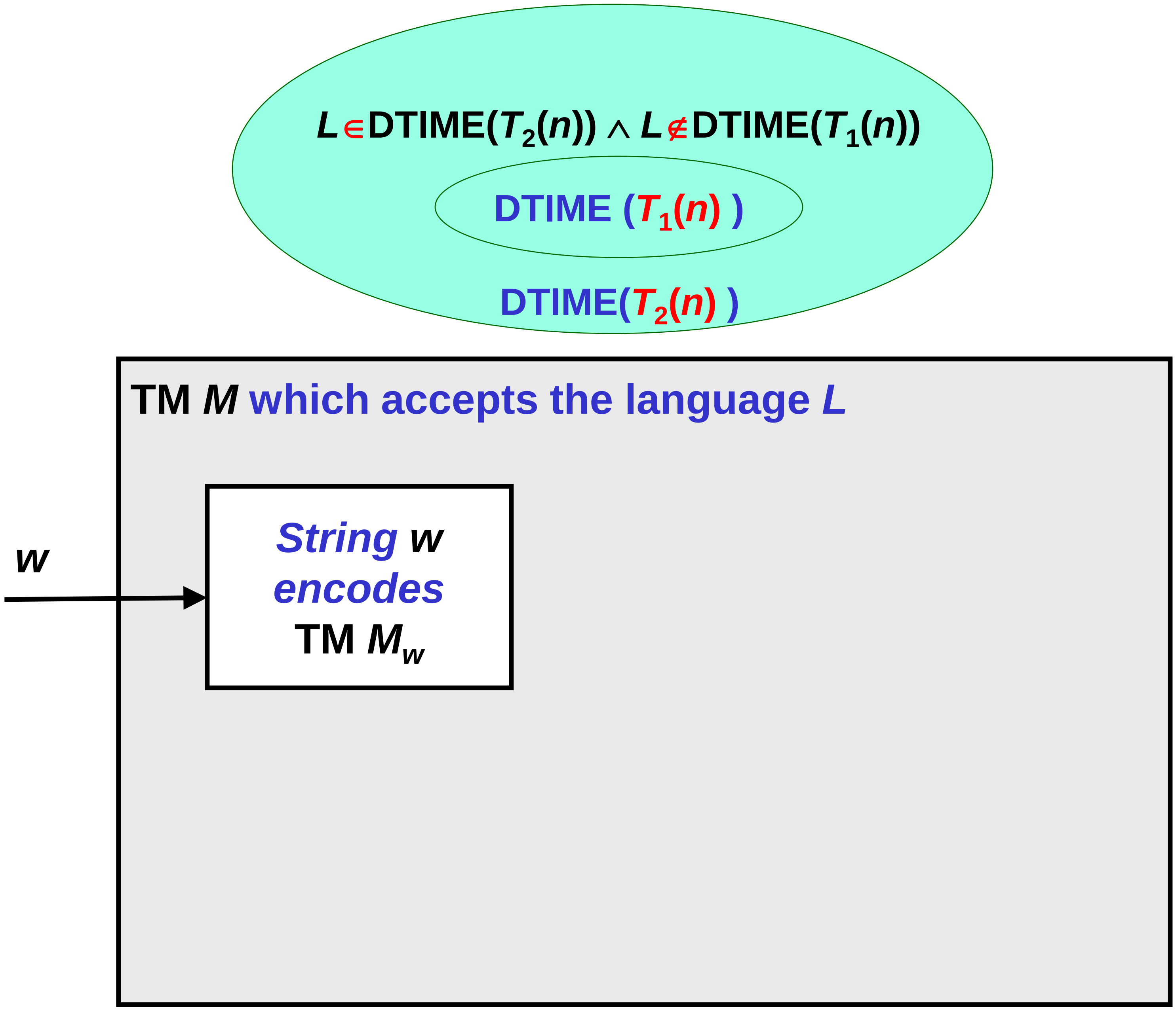
TM *M* which accepts the language *L*

Continuity of the hierarchy for fully time-constructible functions

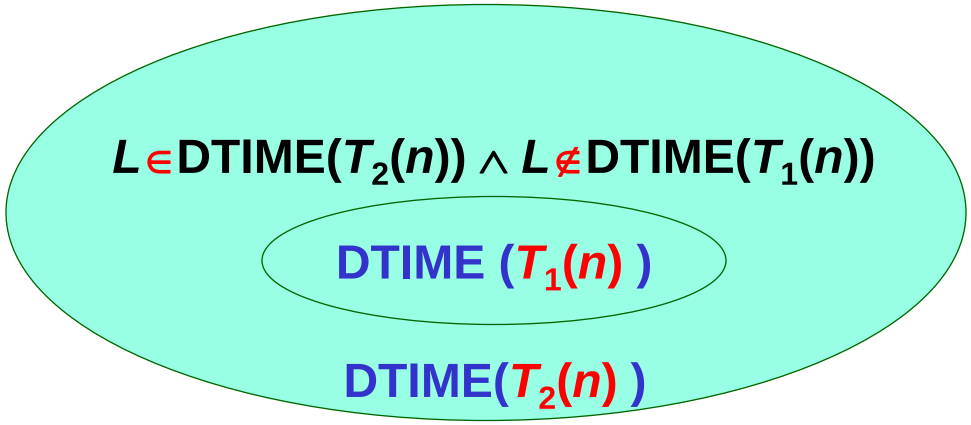


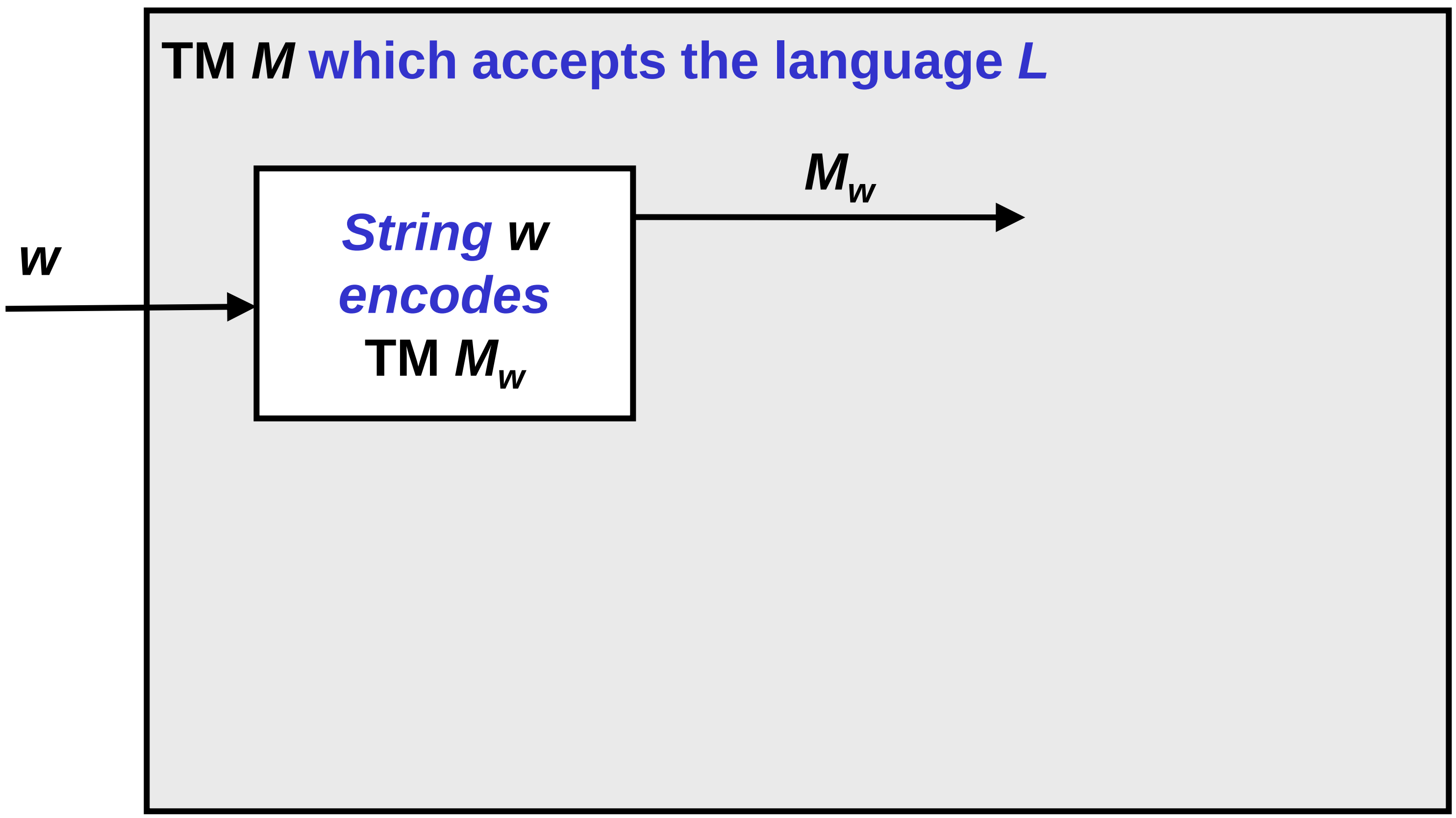


Continuity of the hierarchy for fully time-constructible functions

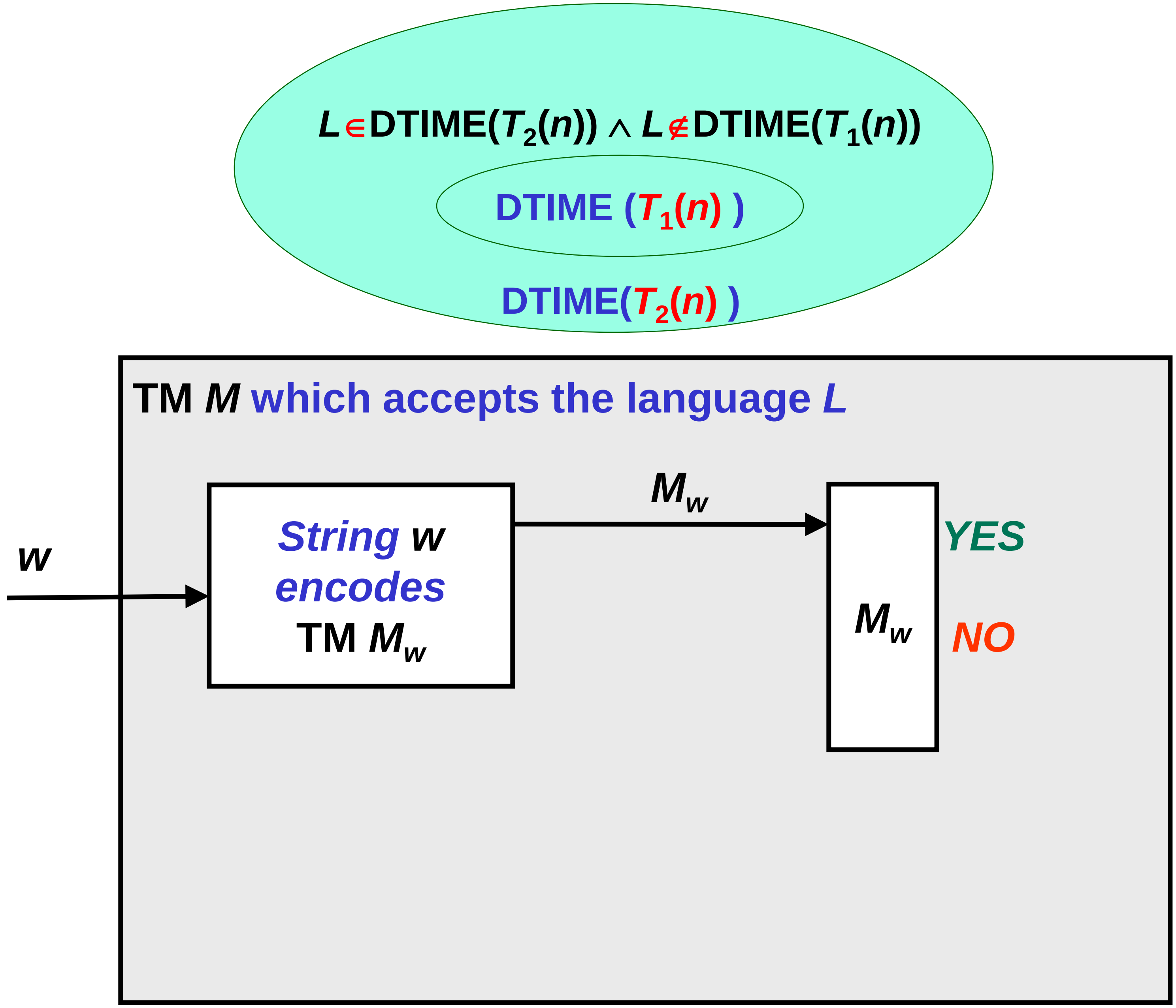


Continuity of the hierarchy for fully time-constructible functions

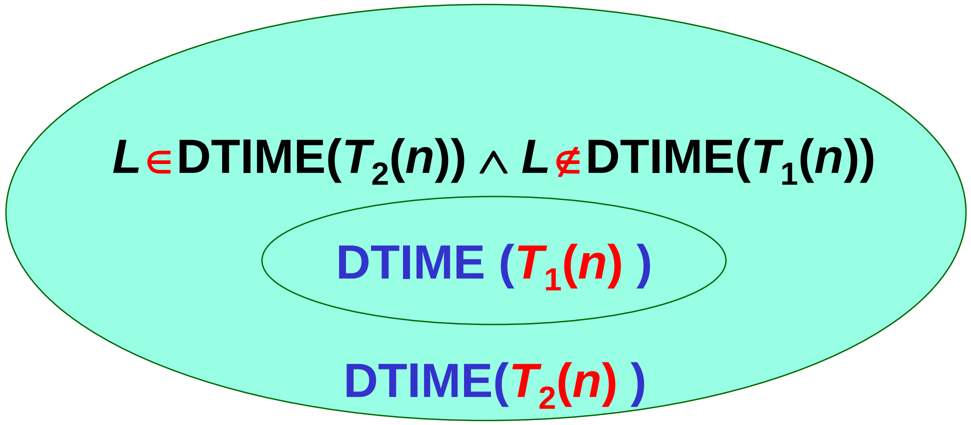


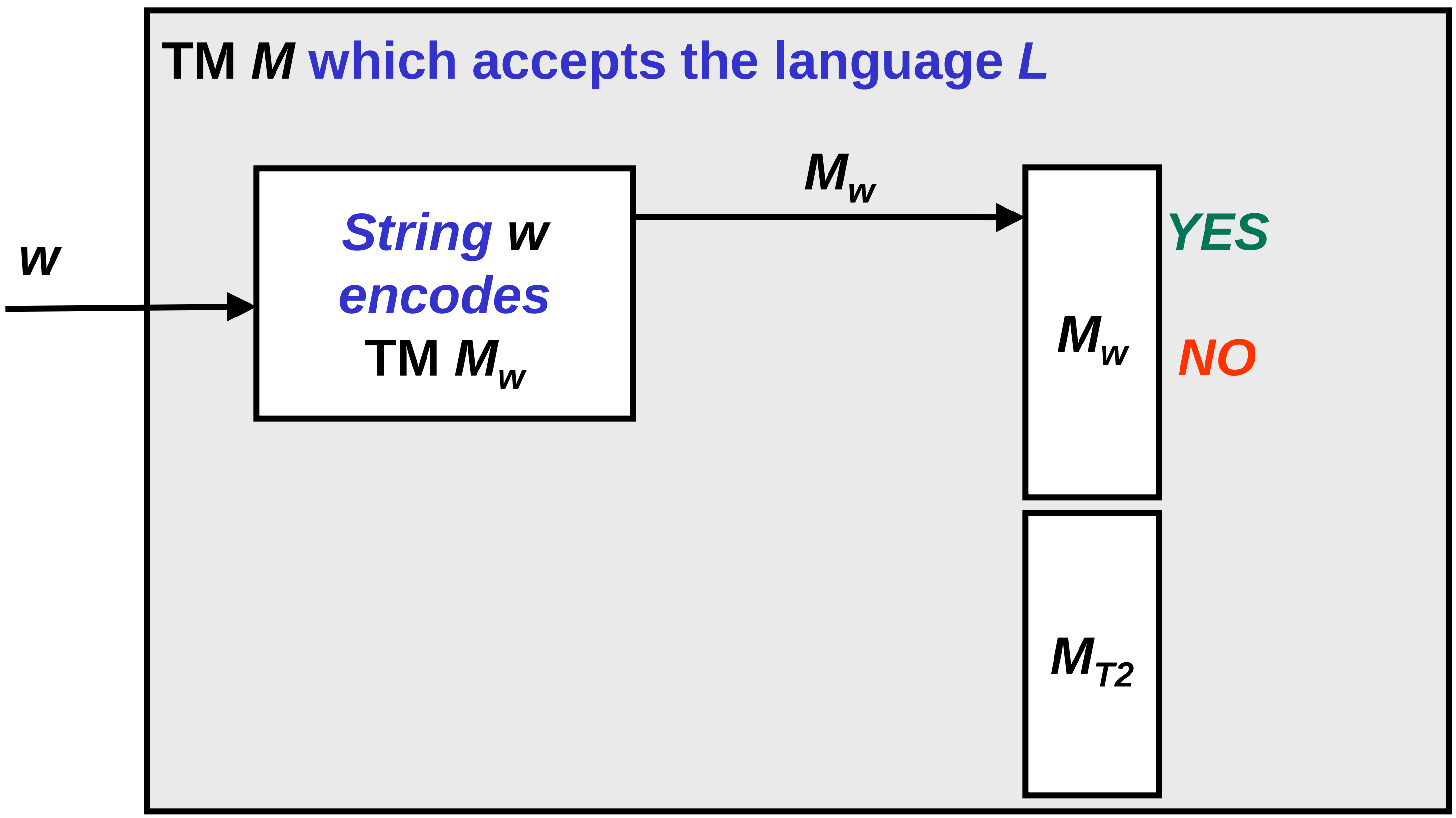


Continuity of the hierarchy for fully time-constructible functions

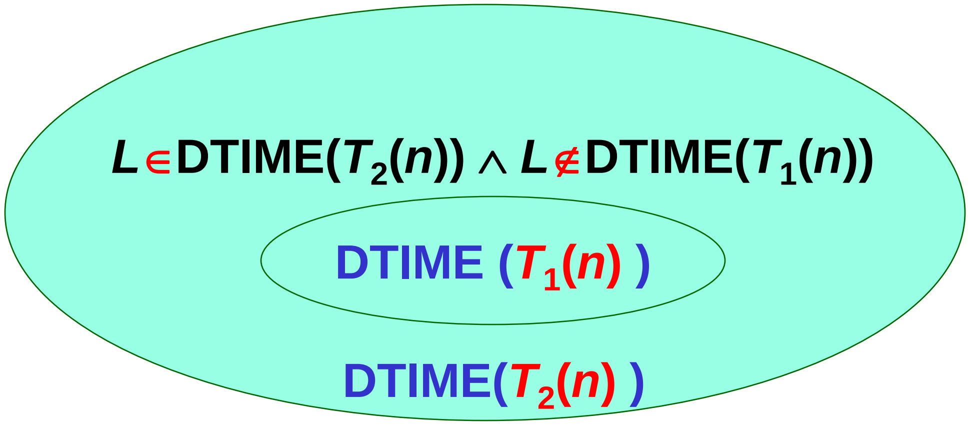


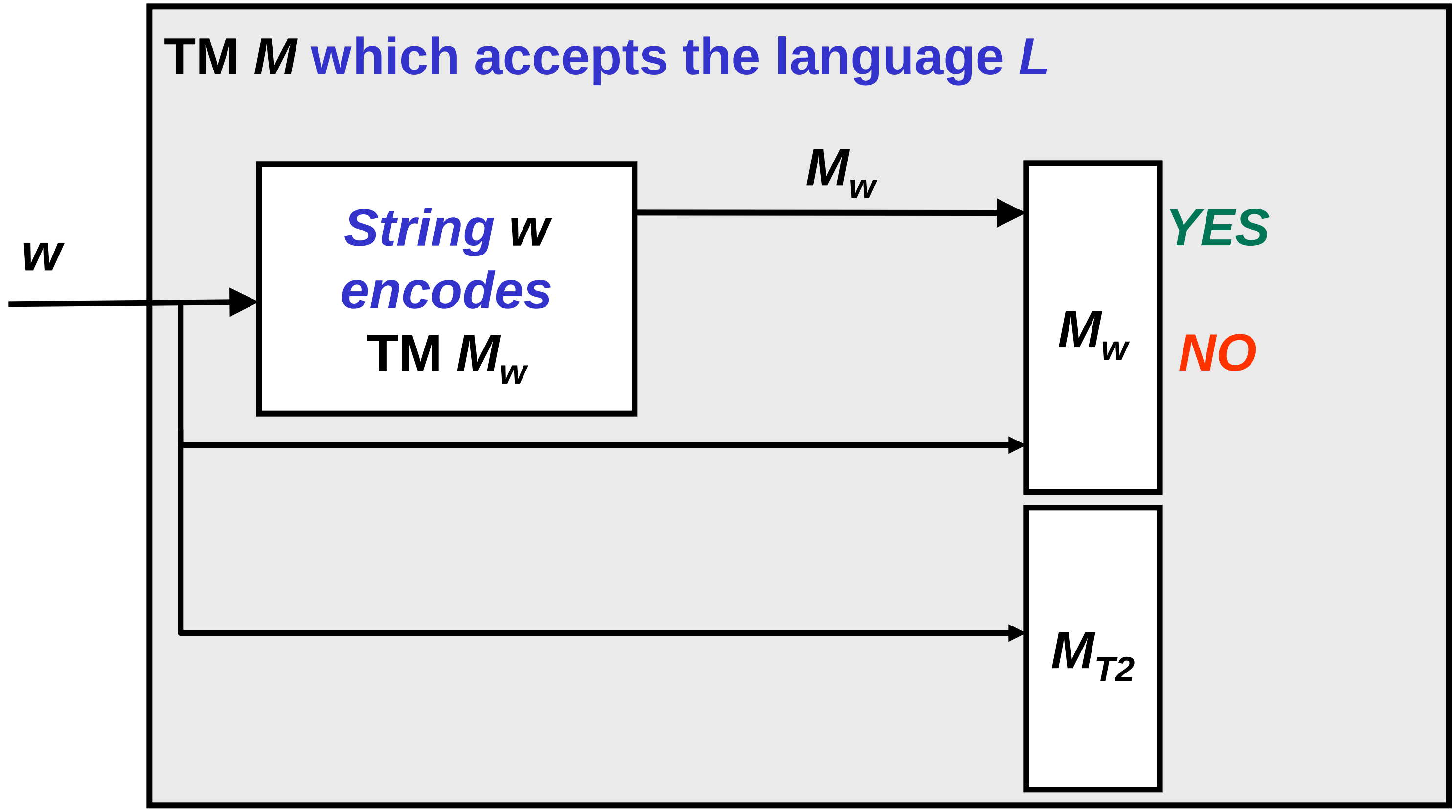
Continuity of the hierarchy for fully time-constructible functions



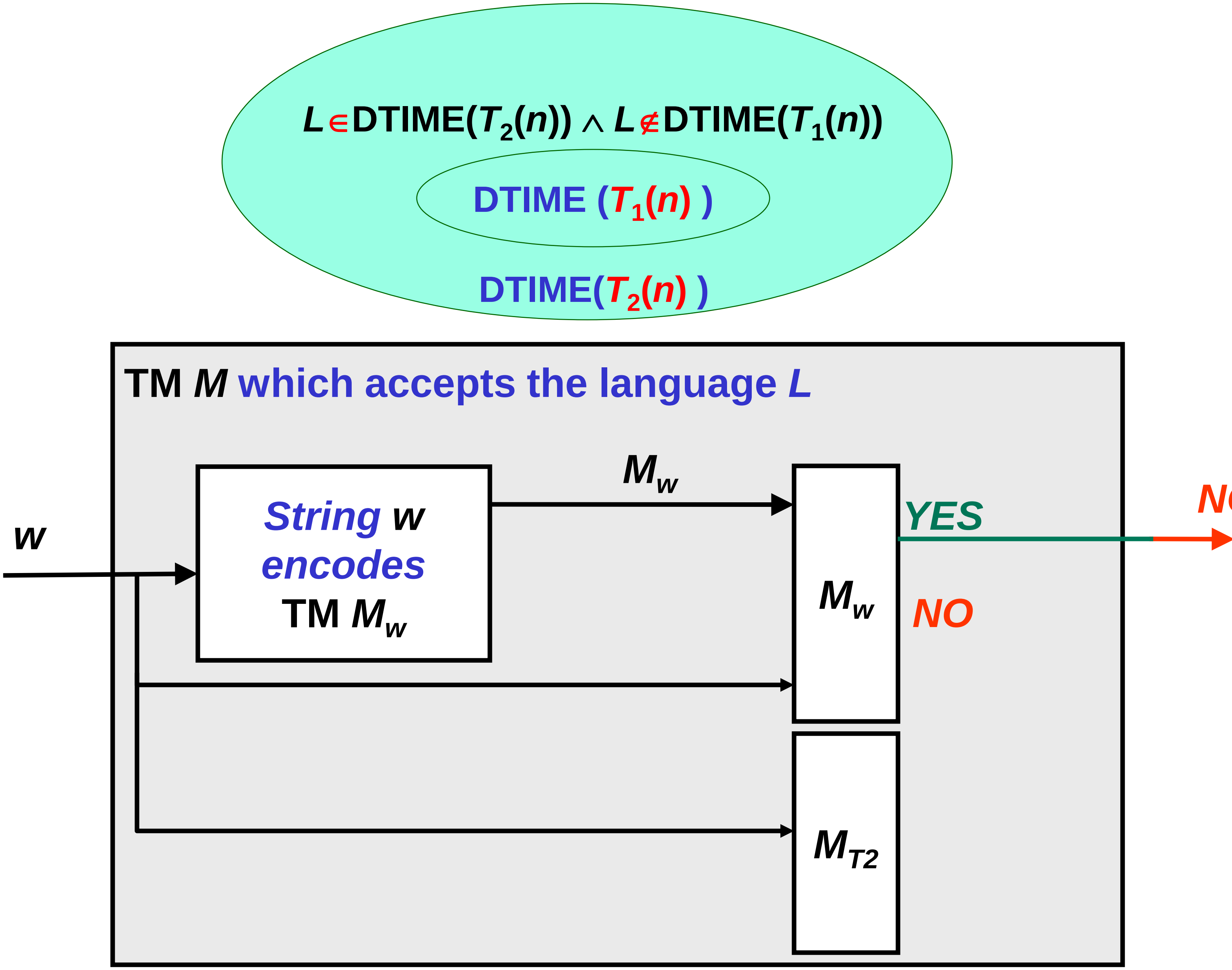


Continuity of the hierarchy for fully time-constructible functions

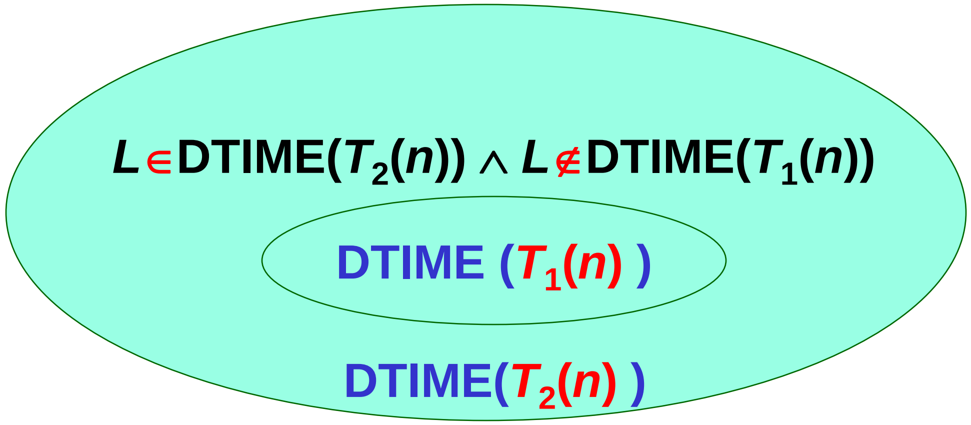


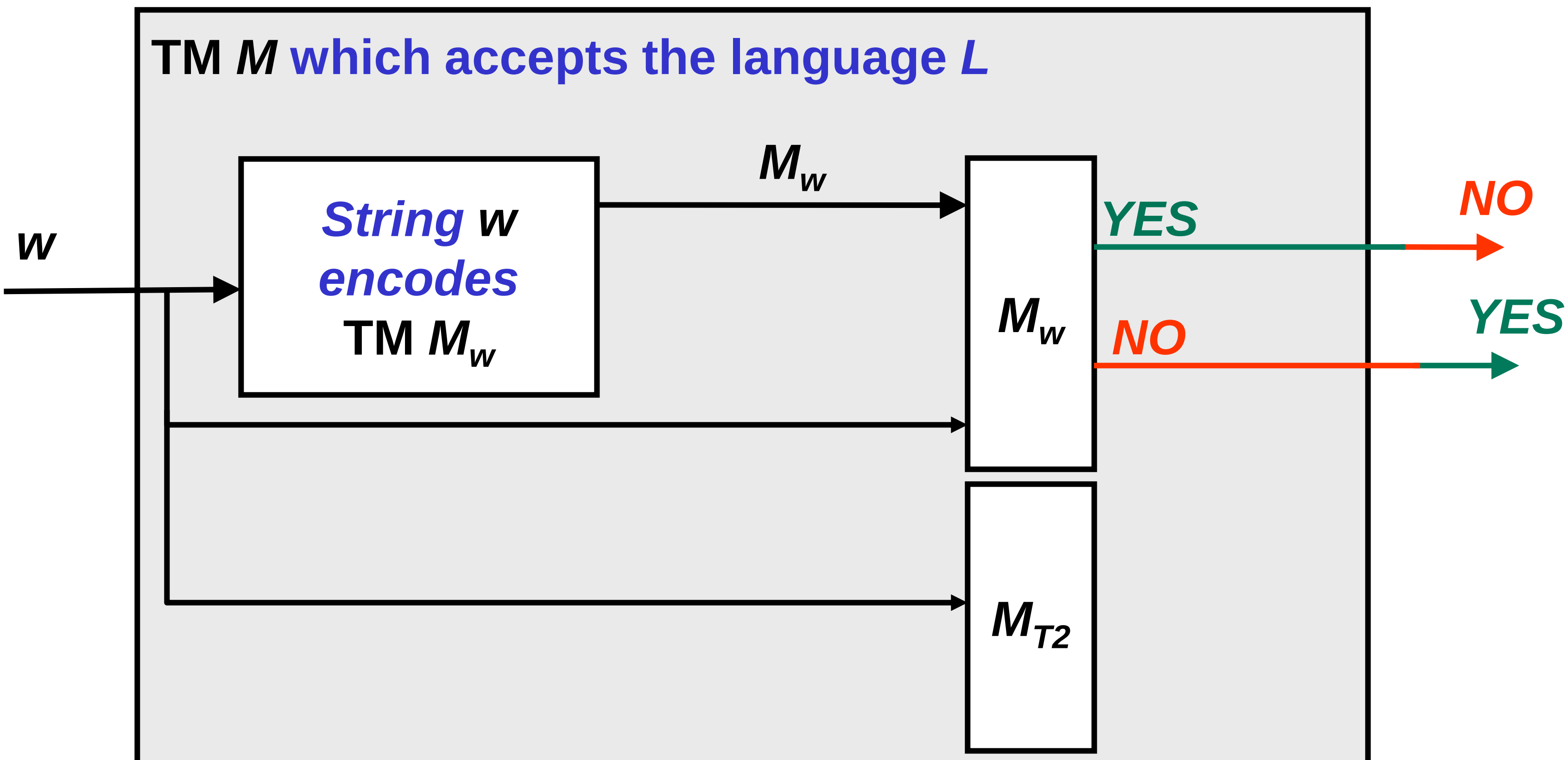


Continuity of the hierarchy for fully time-constructible functions

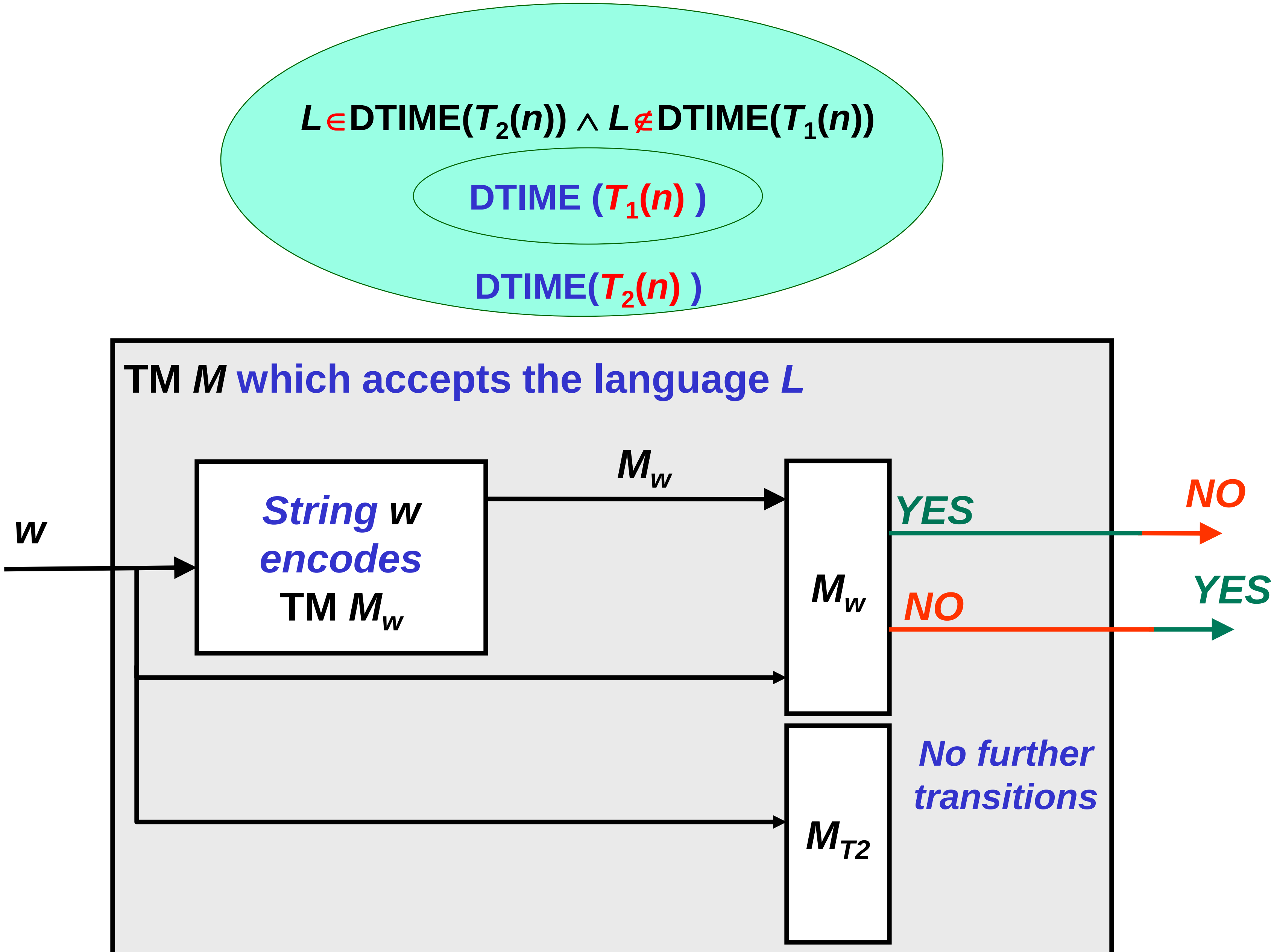


Continuity of the hierarchy for fully time-constructible functions

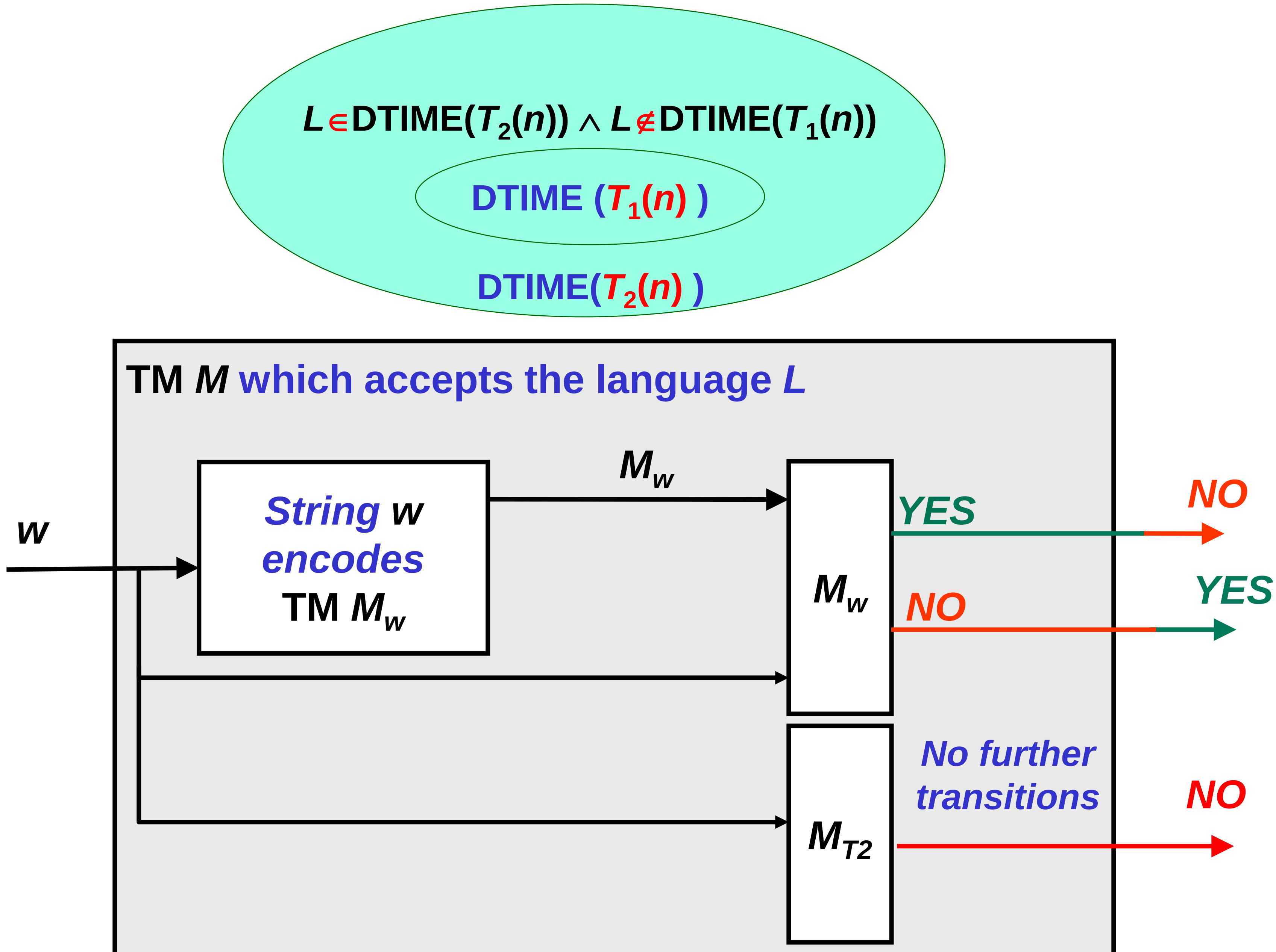




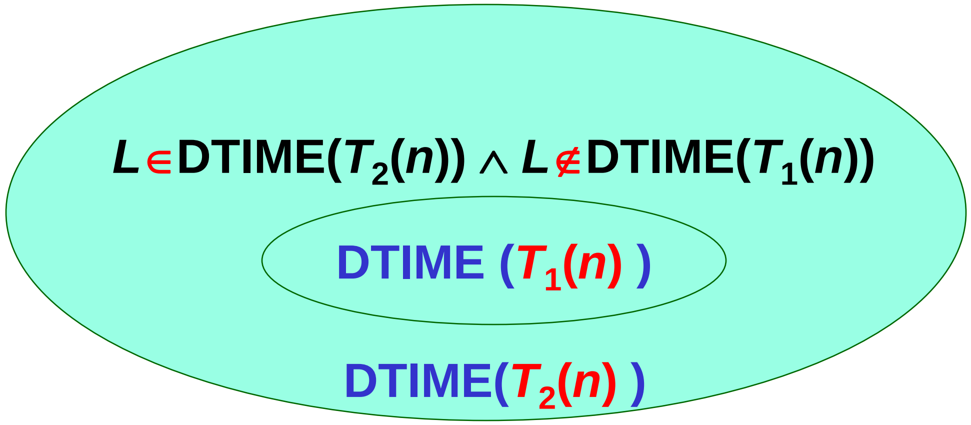
Continuity of the hierarchy for fully time-constructible functions



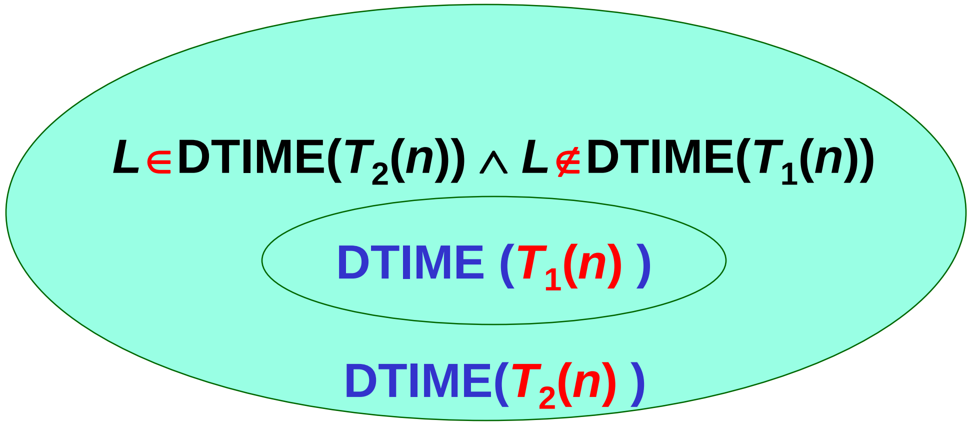
Continuity of the hierarchy for fully time-constructible functions



Continuity of the hierarchy for fully time-constructible functions

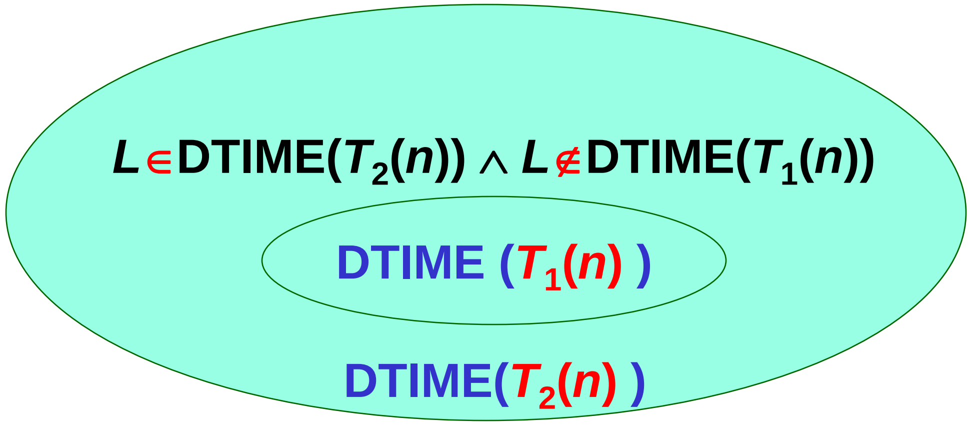


Continuity of the hierarchy for fully time-constructible functions



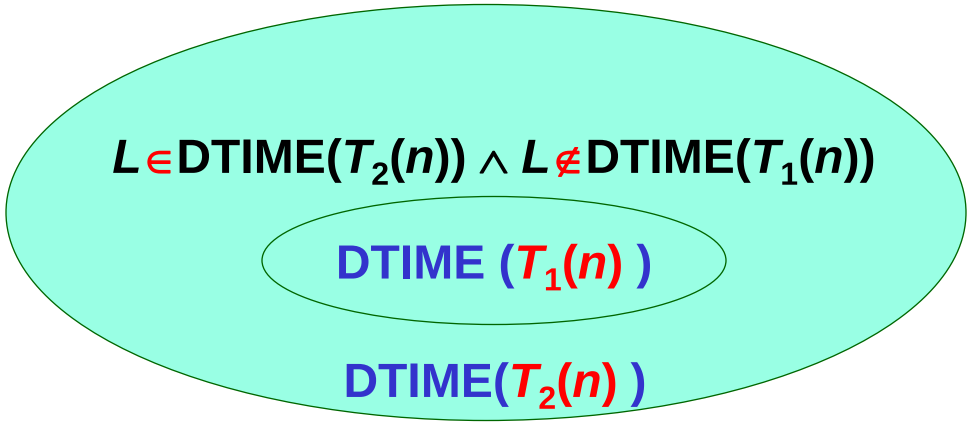
- **Ensure that M simulates M_w from class DTIME$(T_1(n))$ in $T_2(n)$ moves**

Continuity of the hierarchy for fully time-constructible functions



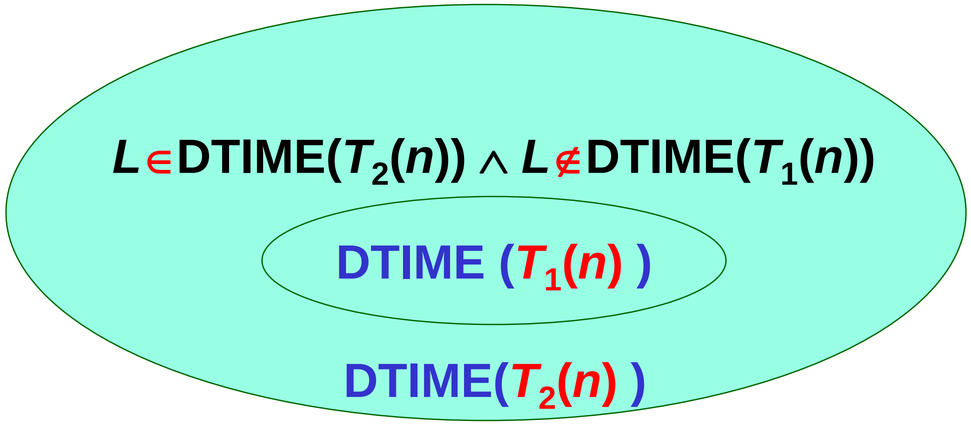
- **Ensure that *M* simulates M_w from class DTIME($T_1(n)$) in $T_2(n)$ moves**
  - **Multiple tapes of TM M_w are reduced to two tapes**
    - **To enable simulating TM M_w with an arbitrary number of tapes by the TM *M* with a limited number of tapes**
    - $T_1(n) \Rightarrow T_1(n)\log T_1(n)$

Continuity of the hierarchy for fully time-constructible functions



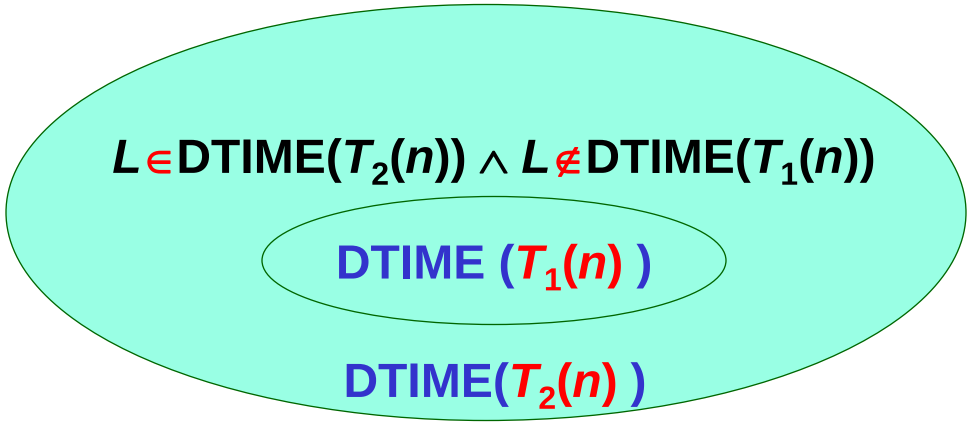
- Ensure that M simulates M_w from class DTIME($T_1(n)$) in $T_2(n)$ moves
  - Multiple tapes of TM M_w are reduced to two tapes
    - To enable simulating TM M_w with an arbitrary number of tapes by the TM M with a limited number of tapes
    - $T_1(n) \Rightarrow T_1(n)\log T_1(n)$
  - $\left(\inf_{n\to\infty} T_1(n)\log T_1(n)/T_2(n) = 0 \right) \Rightarrow \left(T_1(n)\log T_1(n) < T_2(n) \right)$

Continuity of the hierarchy for fully time-constructible functions



- **Ensure that M simulates M_w from class DTIME$(T_1(n))$ in $T_2(n)$ moves**
  - **Multiple tapes of TM M_w are reduced to two tapes**
    - **To enable simulating TM M_w with an arbitrary number of tapes by the TM M with a limited number of tapes**
    - $T_1(n) \Rightarrow T_1(n)\log T_1(n)$
  - $\left(\inf_{n\to\infty} T_1(n)\log T_1(n)/T_2(n) = 0 \right) \Rightarrow \left(T_1(n)\log T_1(n) < T_2(n) \right)$
  - **Encoding procedure of TM:**

Continuity of the hierarchy for fully time-constructible functions



- **Ensure that M simulates M_w from class DTIME($T_1(n)$) in $T_2(n)$ moves**
  - **Multiple tapes of TM M_w are reduced to two tapes**
    - **To enable simulating TM M_w with an arbitrary number of tapes by the TM M with a limited number of tapes**
    - $T_1(n) \Rightarrow T_1(n)\log T_1(n)$
  - $(\ \inf_{n\to\infty} T_1(n)\log T_1(n)/T_2(n) = 0\ ) \Rightarrow (\ T_1(n)\log T_1(n) < T_2(n)\ )$
  - **Encoding procedure of TM:**

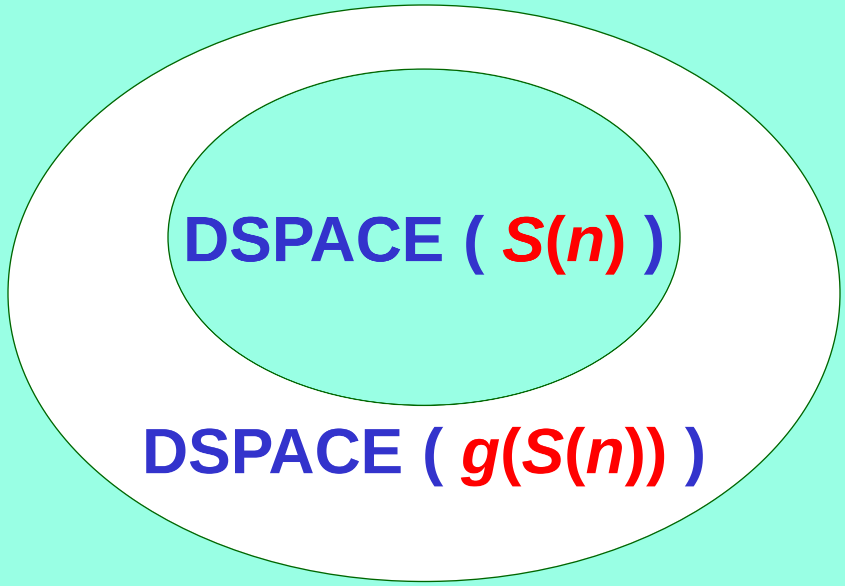
1111111111111111111111 $code_1$ **11** $code_2$ **11 - - - 11** $code_r$ **111**

Gaps in the hierarchy

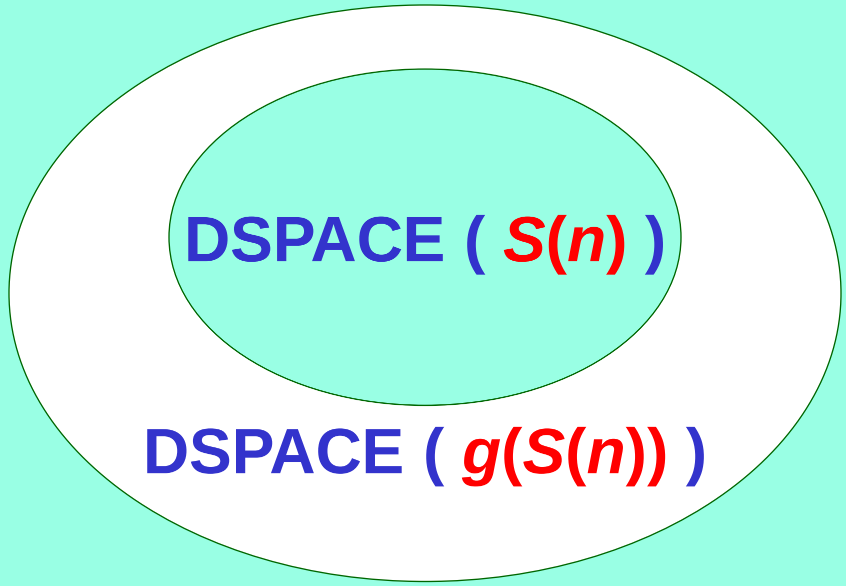
Gaps in the hierarchy

DSPACE (*S*(*n*))

Gaps in the hierarchy

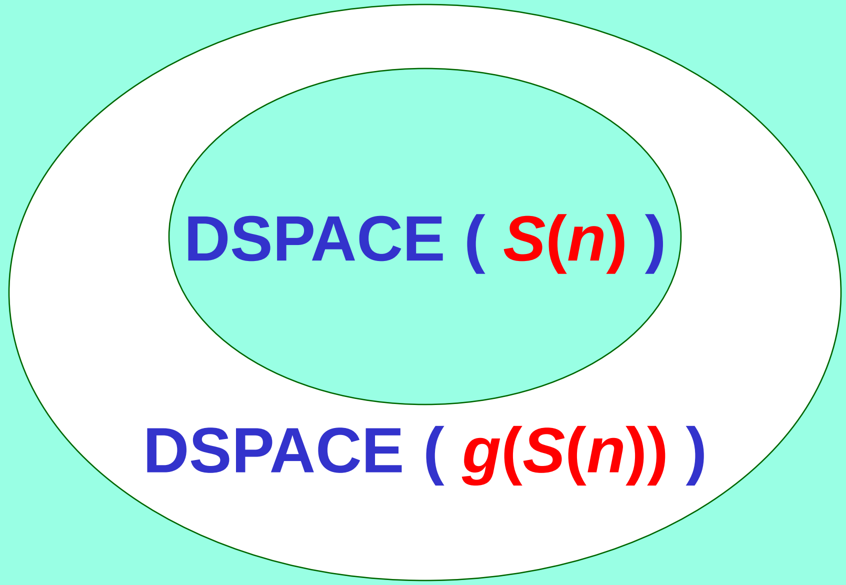


Gaps in the hierarchy



- $g(n)$ is an arbitrary total recursive function, $g(n) \geq n$

Gaps in the hierarchy



- ***g*(*n*) is an arbitrary total recursive function, $g(n) \geq n$**
  - ***g(n)*** is not fully time or space constructible

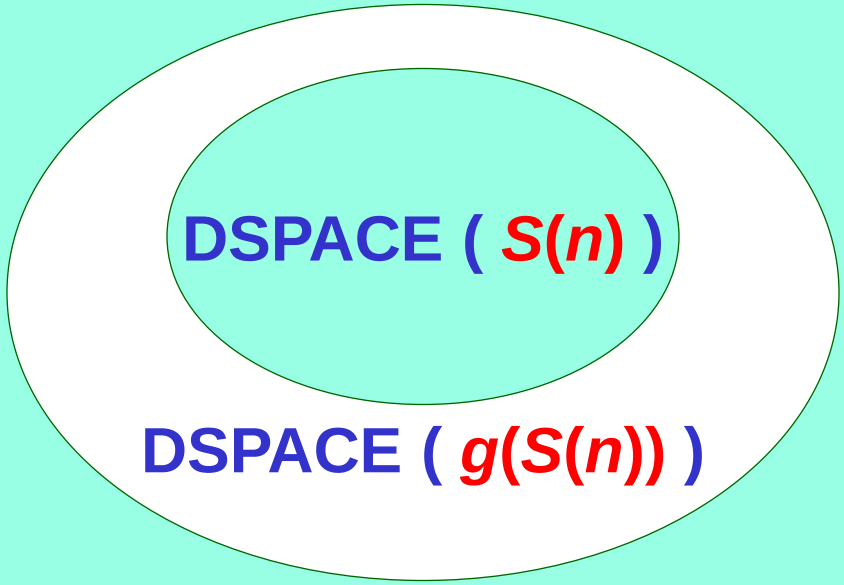
Gaps in the hierarchy

$$\mathrm{DSPACE}\,(\,S(n)\,) \subset \mathrm{DSPACE}\,(\,g(S(n))\,)$$

- $g(n)$ **is an arbitrary total recursive function,** $g(n) \geq n$
  - $g(n)$ is not fully time or space constructible

- ***It is possible to construct*** **a total recursive function** $S(n)$

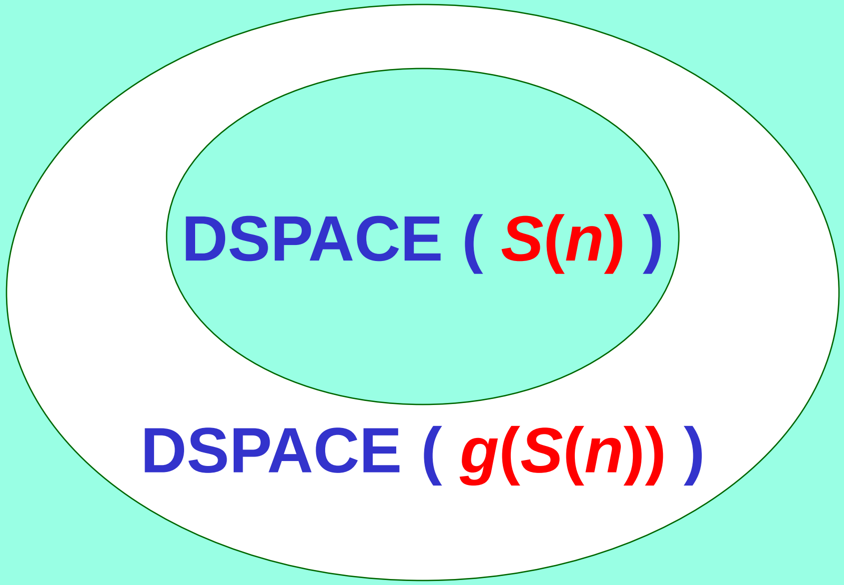
Gaps in the hierarchy



- ***g*(*n*)** **is an arbitrary total recursive function,** ***g*(*n*)≥*n***
    - ***g(n)*** is not fully time or space constructible

- ***It is possible to construct*** **a total recursive function** ***S*(*n*)**
    - DSPACE($S(n)$) = DSPACE($g(S(n))$)

Gaps in the hierarchy



- ***g*(*n*)** **is an arbitrary total recursive function, *g*(*n*)≥*n***
  - ***g(n)*** is not fully time or space constructible

- ***It is possible to construct* a total recursive function *S*(*n*)**
  - DSPACE($S(n)$) = DSPACE($g(S(n))$)
  - DTIME($f(n)$) = NTIME($f(n)$) = DSPACE($f(n)$) = NSPACE($f(n)$)

Optimal TM

Optimal TM

- $r(n)$ – any total recursive function

Optimal TM

- ***r*(*n*) – any total recursive function**
- ***It is possible to construct* a language *L***

Optimal TM

- ***r*(*n*) – any total recursive function**
- ***It is possible to construct* a language *L***
    - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$

Optimal TM

- ***r*(*n*) – any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$

Optimal TM

- ***r*(*n*) – any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Optimal TM

- ***r*(*n*) – any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1

Optimal TM

- **$r(n)$ – any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1	**Code TM M_2**

Optimal TM

- $r(n)$ – **any total recursive function**
- ***It is possible to construct* a language L**
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1	**Code TM M_2**	**Code TM M_3**

Optimal TM

- ***r*(*n*) – any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1	**Code TM M_2**	**Code TM M_3**	**Code TM M_4**

Optimal TM

- **$r(n)$ – any total recursive function**
- ***It is possible to construct* a language L**
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1	**Code TM M_2**	**Code TM M_3**	**Code TM M_4**	**Code TM M_5**

Optimal TM

- ***r*(*n*) – any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1	**Code TM M_2**	**Code TM M_3**	**Code TM M_4**	**Code TM M_5**	**Code TM M_6**

- ***r*(*n*) – any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1	**Code TM M_2**	**Code TM M_3**	**Code TM M_4**	**Code TM M_5**	**Code TM M_6**

$L(M_2) = L$

Optimal TM

- ***r*(*n*) – any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1	**Code TM M_2**	**Code TM M_3**	**Code TM M_4**	**Code TM M_5**	**Code TM M_6**

$L(M_2) = L$ $L(M_4) = L$

Optimal TM

- ***r*(*n*)** **– any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1	**Code TM M_2**	**Code TM M_3**	**Code TM M_4**	**Code TM M_5**	**Code TM M_6**

$L(M_2) = L$ $\qquad$ $L(M_4) = L$ $\qquad$ $L(M_6) = L$

Optimal TM

- ***r*(*n*)** **– any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1	**Code TM M_2**	**Code TM M_3**	**Code TM M_4**	**Code TM M_5**	**Code TM M_6**

$L(M_2) = L$ $\quad$ $L(M_4) = L$ $\quad$ $L(M_6) = L$

$$S_2(n) \geq r(S_4(n))$$

Optimal TM

- ***r*(*n*) – any total recursive function**
- ***It is possible to construct* a language *L***
  - for any TM M_i with complexity $S_i(n)$ that accepts the language $L(M_i)=L$
  - there is a TM M_j with complexity $S_j(n)$ that accepts the language $L(M_j)=L(M_i)=L$
  - $S_i(n) \geq r(S_j(n))$ for almost all values of n

Code TM M_1	**Code TM M_2**	**Code TM M_3**	**Code TM M_4**	**Code TM M_5**	**Code TM M_6**

$L(M_2) = L$ $\qquad$ $L(M_4) = L$ $\qquad$ $L(M_6) = L$

$S_2(n) \geq r(S_4(n))$ $\qquad$ $S_4(n) \geq r(S_6(n))$

Union of language classes

Union of language classes

- $\{f_i(n) \mid i=1, 2, \ldots\}$ – a set of recursive functions

Union of language classes

- $\{f_i(n) \mid i=1, 2, \ldots\}$ – a set of recursive functions
- If we can construct a TM *M* that outputs the list

Union of language classes

- **$\{f_i(n) \mid i=1, 2, ...\}$ – a set of recursive functions**
- **If we can construct a TM *M* that outputs the list**
    - TM M_1, M_2, ... compute $f_1(n)$, $f_2(n)$, ...

Union of language classes

- **$\{f_i(n) \mid i=1, 2, ...\}$ – a set of recursive functions**
- **If we can construct a TM M that outputs the list**
  - TM M_1, M_2, ... compute $f_1(n)$, $f_2(n)$, ...
- **and if for all n and i it holds that**

Union of language classes

- $\{f_i(n) \mid i=1, 2, \ldots\}$ **– a set of recursive functions**
- **If we can construct a TM *M* that outputs the list**
  - TM M_1, M_2, ... compute $f_1(n)$, $f_2(n)$, ...
- **and if for all *n* and *i* it holds that**
  - $f_i(n) < f_{i+1}(n)$

Union of language classes

- **$\{f_i(n) \mid i=1, 2, ...\}$ – a set of recursive functions**
- **If we can construct a TM *M* that outputs the list**
  - TM M_1, M_2, ... compute $f_1(n)$, $f_2(n)$, ...
- **and if for all *n* and *i* it holds that**
  - $f_i(n) < f_{i+1}(n)$
- **then there is a recursive function *S*(*n*) for which**

Union of language classes

- **$\{f_i(n) \mid i=1, 2, \ldots\}$ – a set of recursive functions**
- **If we can construct a TM *M* that outputs the list**
  - TM M_1, M_2, ... compute $f_1(n)$, $f_2(n)$, ...
- **and if for all *n* and *i* it holds that**
  - $f_i(n) < f_{i+1}(n)$
- **then there is a recursive function $S(n)$ for which**

$$\mathrm{DSPACE}(\, S(n) \,) = \bigcup_{i \geq 1} \mathrm{DSPACE}(\, f_i(n) \,)$$

Union of language classes

- $\{f_i(n) \mid i=1, 2, \ldots\}$ – a set of recursive functions
- If we can construct a TM *M* that outputs the list
  - TM $M_1, M_2, \ldots$ compute $f_1(n), f_2(n), \ldots$.
- and if for all n and i it holds that
  - $f_i(n) < f_{i+1}(n)$
- then there is a recursive function $S(n)$ for which

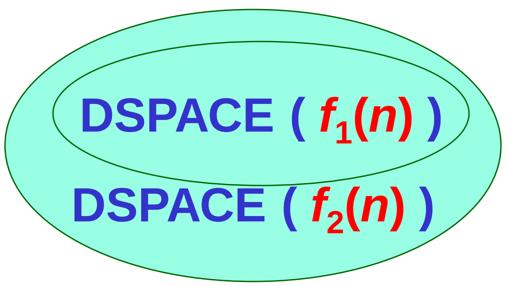
$$\mathrm{DSPACE}(\ S(n)\ ) = \bigcup_{i \geq 1} \mathrm{DSPACE}(\ f_i(n)\ )$$

DSPACE ($f_1(n)$)

Union of language classes

- $\{f_i(n) \mid i=1, 2, \ldots\}$ – a set of recursive functions
- If we can construct a TM *M* that outputs the list
  - TM M_1, M_2, ... compute $f_1(n)$, $f_2(n)$, ...
- and if for all *n* and *i* it holds that
  - $f_i(n) < f_{i+1}(n)$
- then there is a recursive function $S(n)$ for which

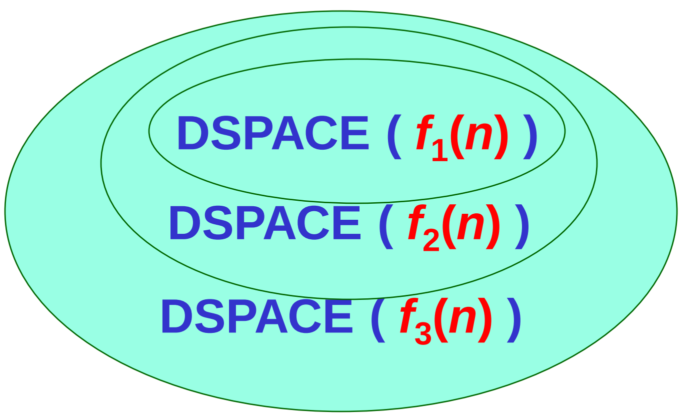
$$\mathrm{DSPACE}(\,S(n)\,) = \bigcup_{i \geq 1} \mathrm{DSPACE}(\,f_i(n)\,)$$



Union of language classes

- **$\{f_i(n) \mid i=1, 2, \ldots\}$ – a set of recursive functions**
- **If we can construct a TM *M* that outputs the list**
  - TM M_1, M_2, ... compute $f_1(n)$, $f_2(n)$, ...
- **and if for all *n* and *i* it holds that**
  - $f_i(n) < f_{i+1}(n)$
- **then there is a recursive function $S(n)$ for which**

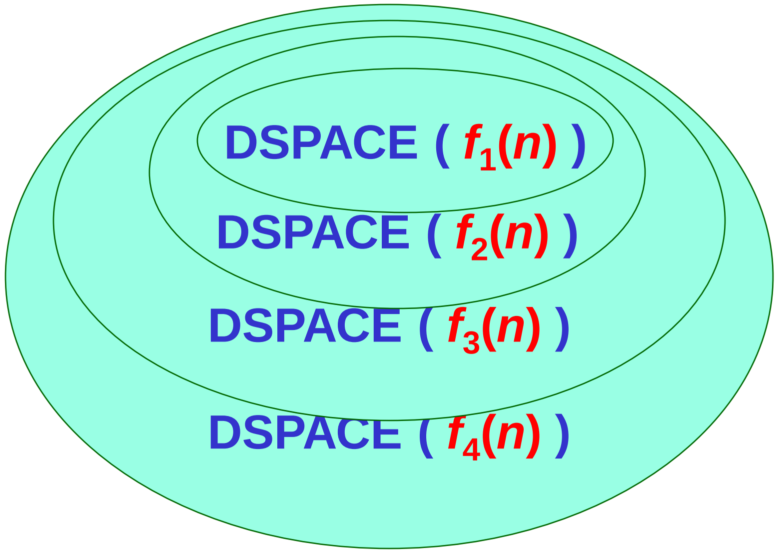
$$\mathrm{DSPACE}(\, S(n)\,) = \bigcup_{i \geq 1} \mathrm{DSPACE}(\, f_i(n)\,)$$



Union of language classes

- $\{f_i(n) \mid i=1, 2, \ldots\}$ – a set of recursive functions
- If we can construct a TM M that outputs the list
  - TM M_1, M_2, ... compute $f_1(n)$, $f_2(n)$, ...
- and if for all n and i it holds that
  - $f_i(n) < f_{i+1}(n)$
- then there is a recursive function $S(n)$ for which

$$\text{DSPACE}(\ S(n)\ ) = \bigcup_{i \geq 1} \text{DSPACE}(\ f_i(n)\ )$$

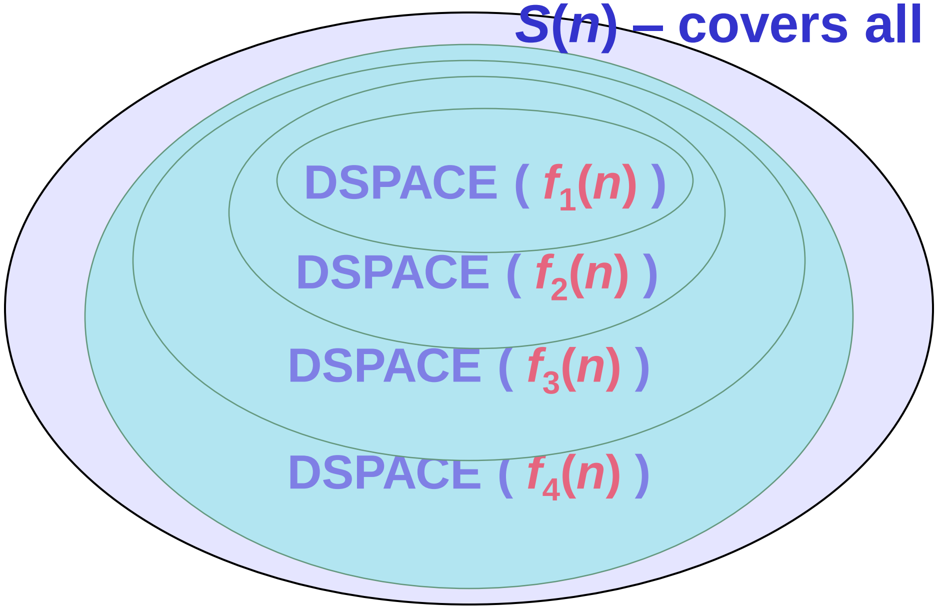


Union of language classes

- $\{f_i(n) \mid i=1, 2, \ldots\}$ – a set of recursive functions
- If we can construct a TM M that outputs the list
  - TM M_1, M_2, ... compute $f_1(n)$, $f_2(n)$, ...
- and if for all n and i it holds that
  - $f_i(n) < f_{i+1}(n)$
- then there is a recursive function $S(n)$ for which

$$\text{DSPACE}(\ S(n)\ ) = \bigcup_{i \geq 1} \text{DSPACE}(\ f_i(n)\ )$$

$S(n)$ – covers all languages for all classes $f_i(n)$



Union of language classes

- $\{f_i(n) \mid i=1, 2, ...\}$ – a set of recursive functions
- If we can construct a TM M that outputs the list
  - TM M_1, M_2, ... compute $f_1(n)$, $f_2(n)$, ...
- and if for all n and i it holds that
  - $f_i(n) < f_{i+1}(n)$
- then there is a recursive function $S(n)$ for which

$$\text{DSPACE}(\, S(n)\,) = \bigcup_{i \geq 1} \text{DSPACE}(\, f_i(n)\,)$$

$S(n)$ – does not cover any other langauge