Separation by Locally Testable Languages

Thomas Place, Lorijn van Rooijen, Marc Zeitoun

LaBRI Université Bordeaux, CNRS, FRANCE

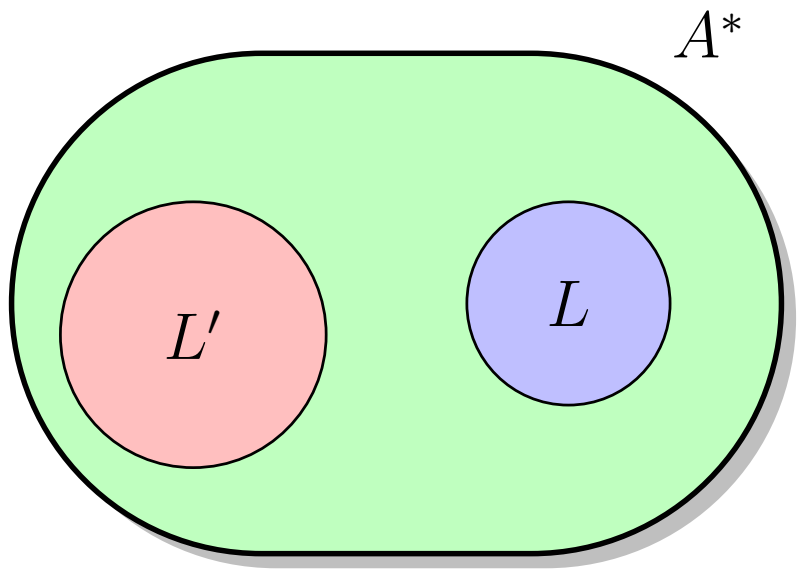
December 13, 2013

What is Separation?

Fix a class **Sep** of regular languages.

Input Regular languages L, L'.

Question Does there exist a separator $K \in$ **Sep** for L, L'?

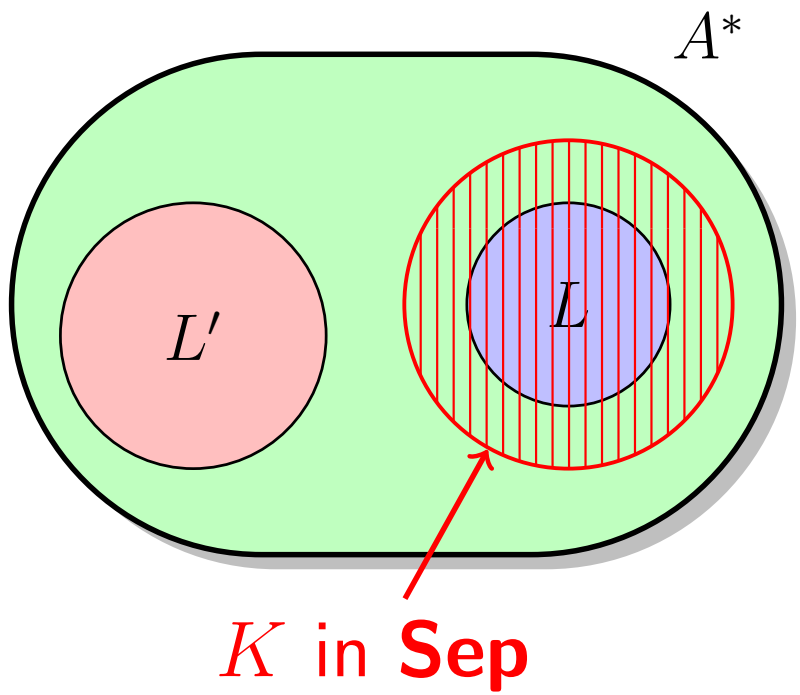


What is Separation?

Fix a class **Sep** of regular languages.

Input Regular languages L, L'.

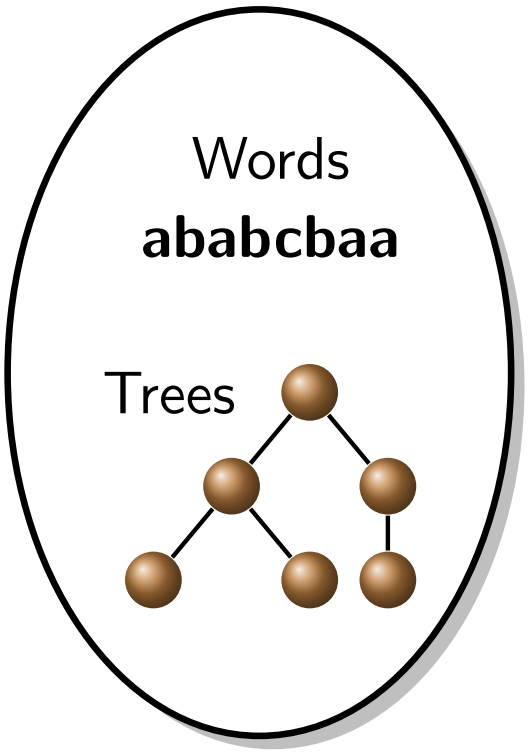
Question Does there exist a separator $K \in$ **Sep** for L, L'?



Separation:
Why should you try it?

Objects we consider

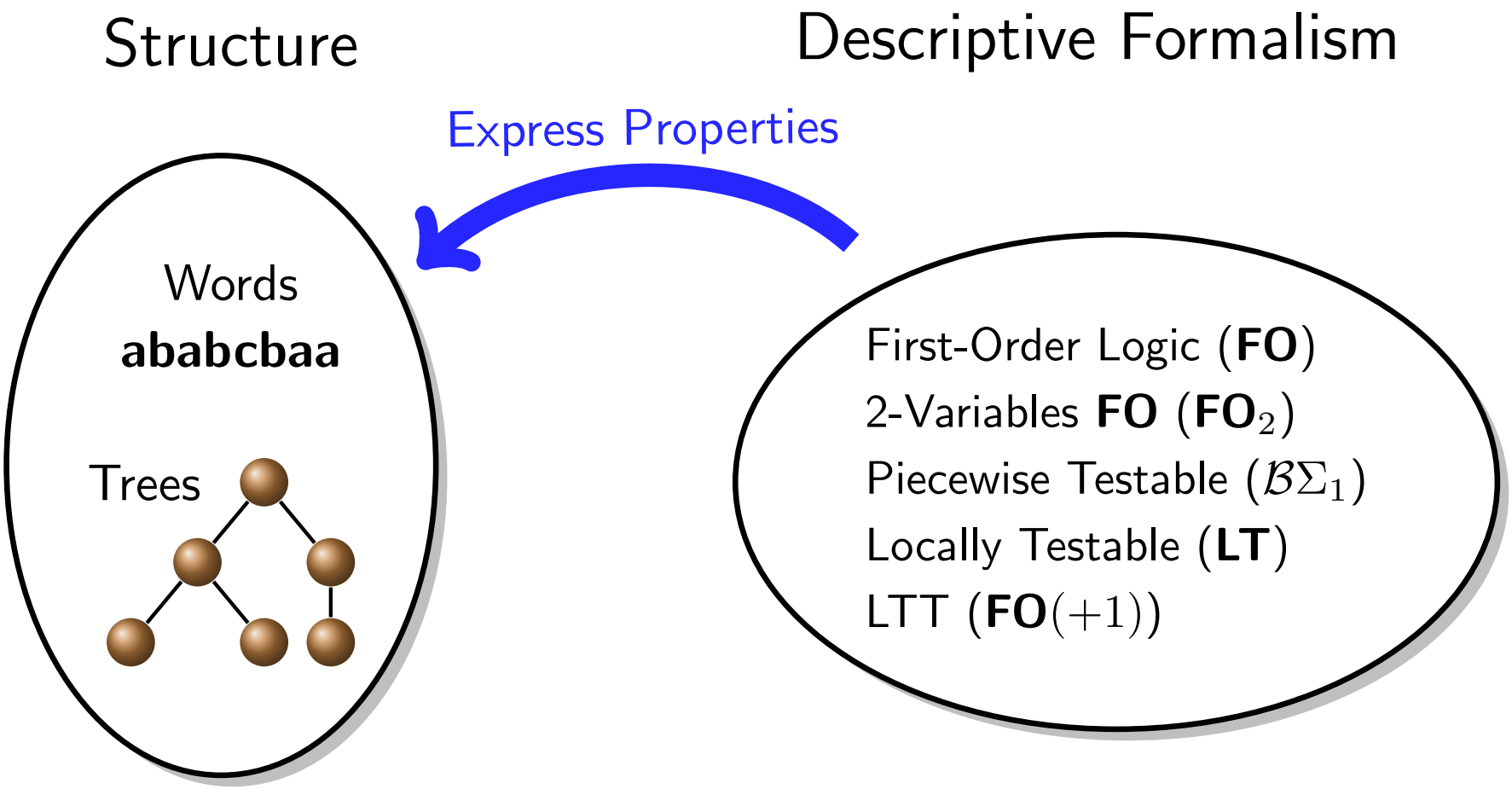
Structure



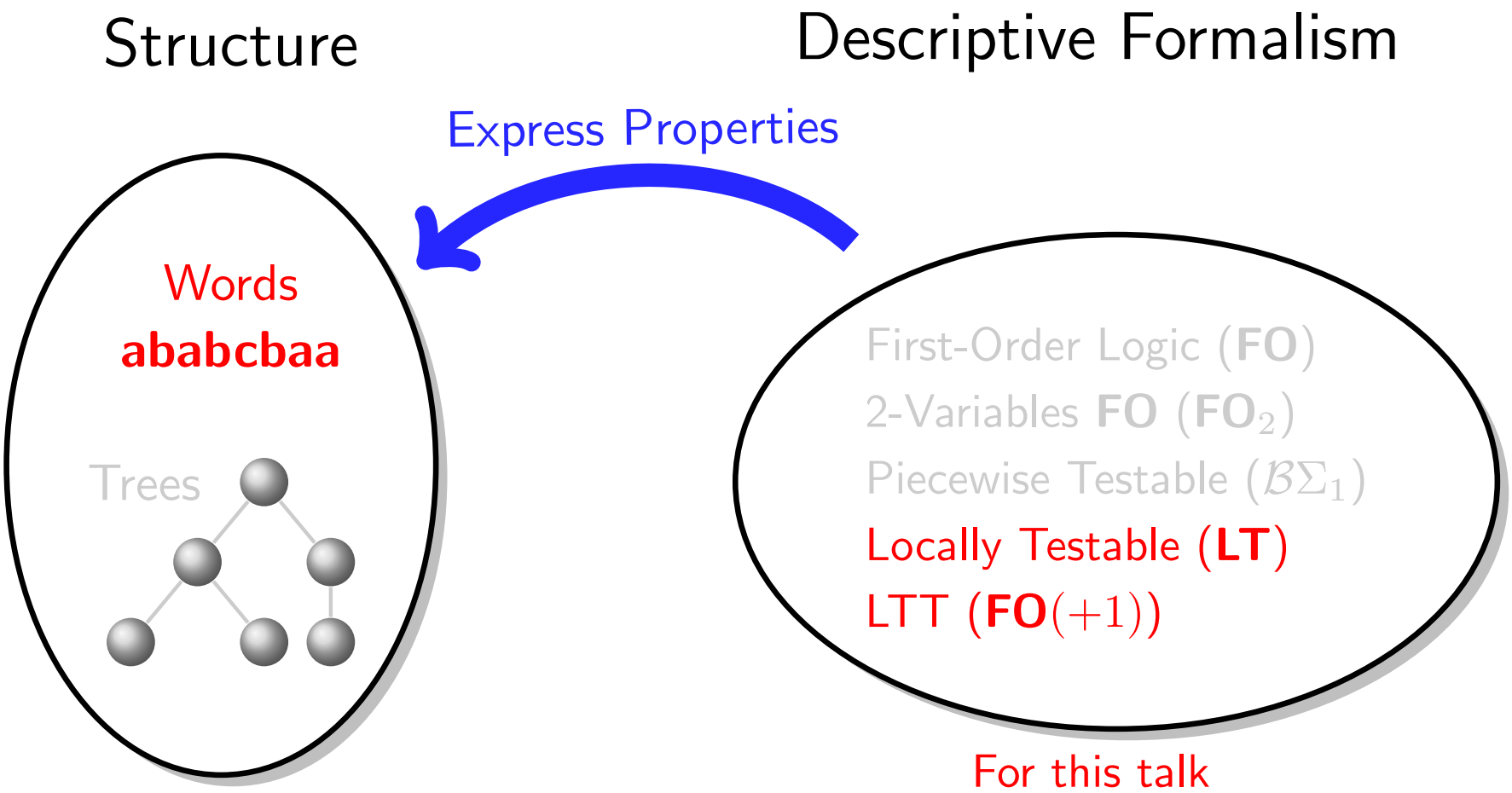
Descriptive Formalism

First-Order Logic (**FO**)
2-Variables **FO** (**FO**$_2$)
Piecewise Testable ($\mathcal{B}\Sigma_1$)
Locally Testable (**LT**)
LTT (**FO**$(+1)$)

Objects we consider



Objects we consider



Expressiveness of Descriptive Formalisms

2 ways to capture expressiveness:

- **Level 1**. Expressive power. *Membership.*

  Can the formalism express an input language?

$$\exists \varphi,\ L = L(\varphi)?$$

Expressiveness of Descriptive Formalisms

2 ways to capture expressiveness:

- **Level 1**. Expressive power. *Membership.*

  Can the formalism express an input language?

$$\exists \varphi,\ L = L(\varphi)?$$

- **Level 2**. Discriminating power. *Separation.*

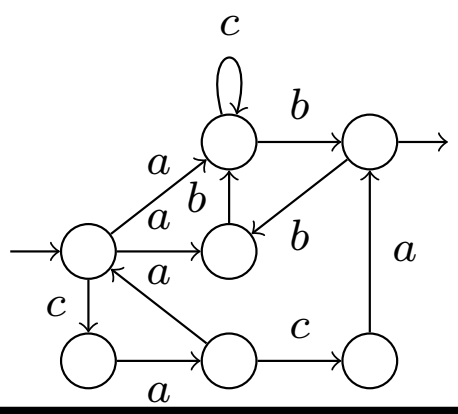
  Can the formalism discriminate between 2 input languages?

$$\exists \varphi, \quad L \subseteq L(\varphi) \ \text{ and } \ L(\varphi) \cap L' = \varnothing?$$

A Reduced Problem: Decidable Characterizations

Decide the following problem:

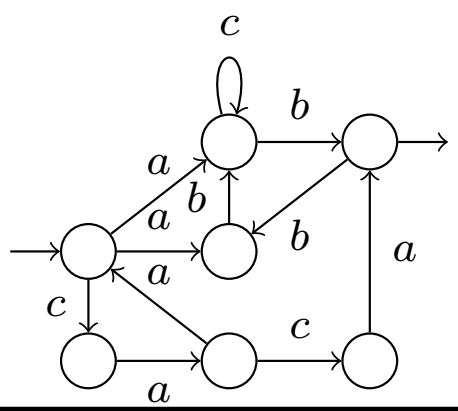
Take a regular language L



A Reduced Problem: Decidable Characterizations

Decide the following problem:

Take a regular language L



Is it an **LT** language ?

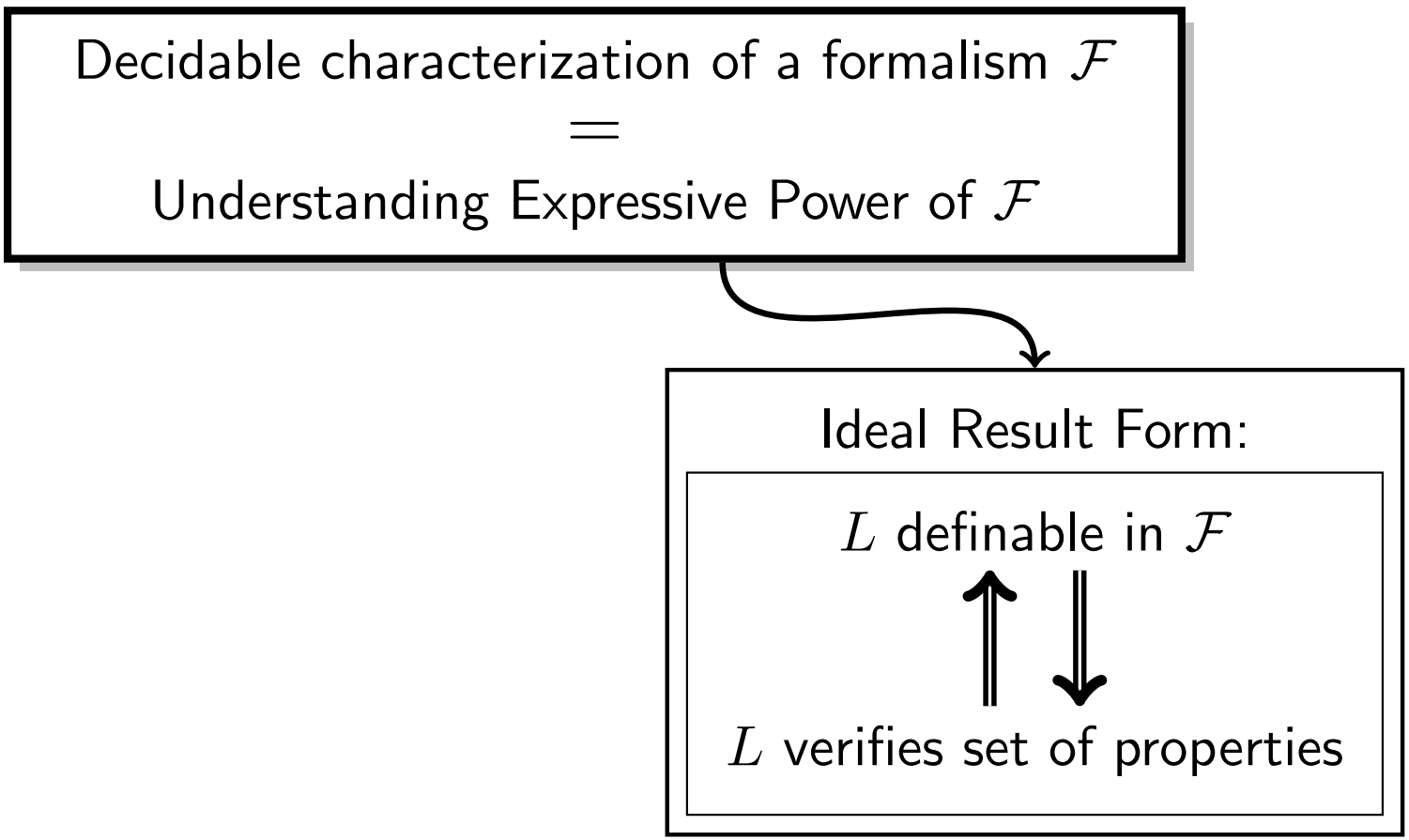
Why do we study these things ?

Decidable characterization of a formalism $\mathcal{F}$

$=$

Understanding Expressive Power of $\mathcal{F}$

Why do we study these things ?

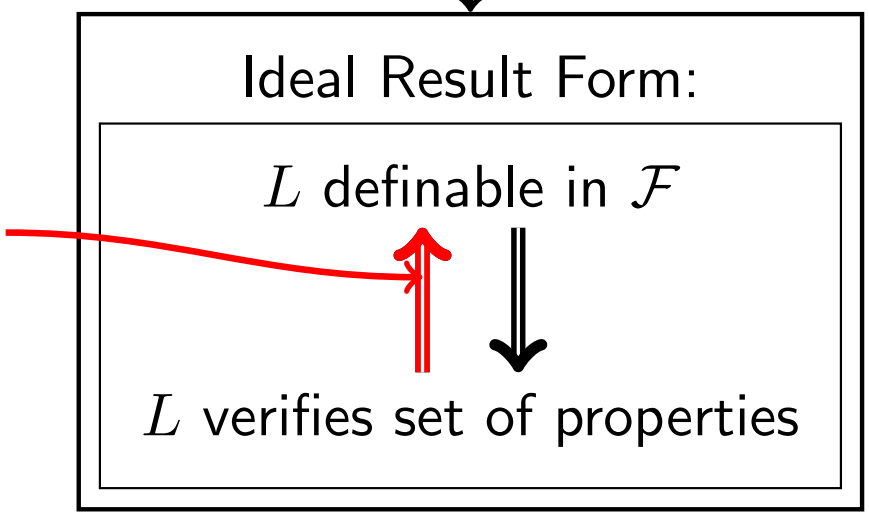


Why do we study these things ?

Decidable characterization of a formalism $\mathcal{F}$
$=$
Understanding Expressive Power of $\mathcal{F}$

Proof = Construction of a Formula,
Hypothesis = Simple Properties

$\Rightarrow$ Normal form for formulas



Why do we study these things ?

Decidable characterization of a formalism $\mathcal{F}$

Simon&Brzozowski '73, McNaughton '74

L a regular language, the following are equivalent:

- L is in **LT**.
- The syntactic monoid of L satisfies $\begin{cases} ese = esese, \\ esete = etese \end{cases}$

P

ties

Why do we study these things ?

Decidable characterization of a formalism $\mathcal{F}$

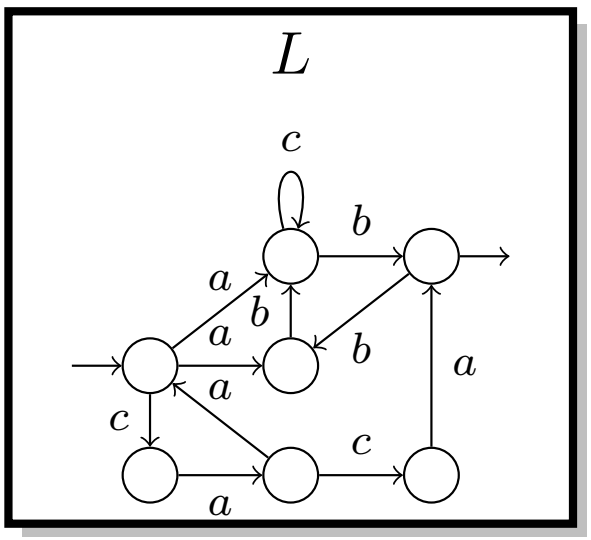
Simon&Brzozowski '73, McNaughton '74

L a regular language, the following are equivalent:

- L is in **LT**.
- The syntactic monoid of L satisfies $\begin{cases} ese = esese, \\ esete = etese \end{cases}$

A counterexample that always happens when not in LT.

Why would we want more ?



If the characterization answer is yes ☺
- All subparts of the automata are in **LT**

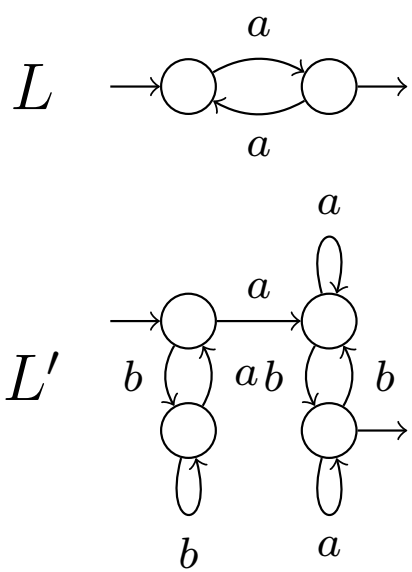
  (provided it is minimal).

If the characterization's answer is no ☹
- We know very little.
- Maybe interesting things can still be said.

Here comes Separation

Decide the following problem:

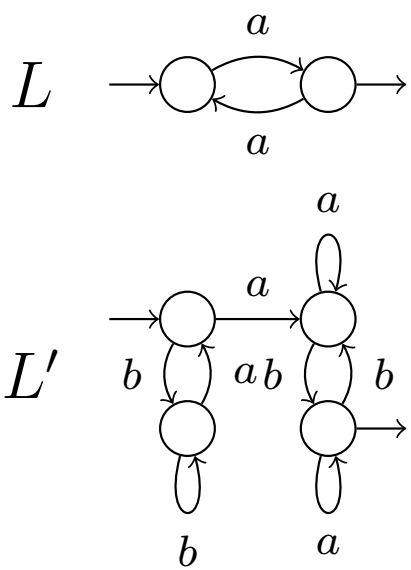
Take two regular languages L, L'



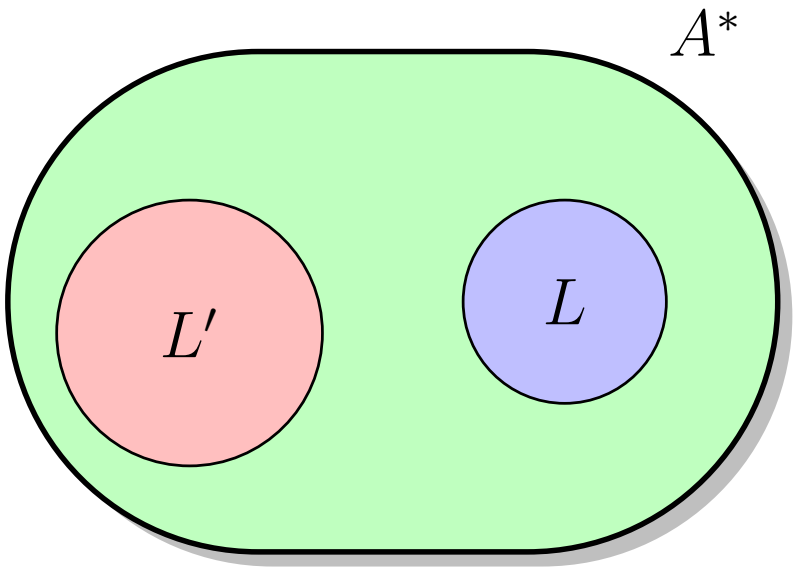
Here comes Separation

Decide the following problem:

Take two regular languages L, L'



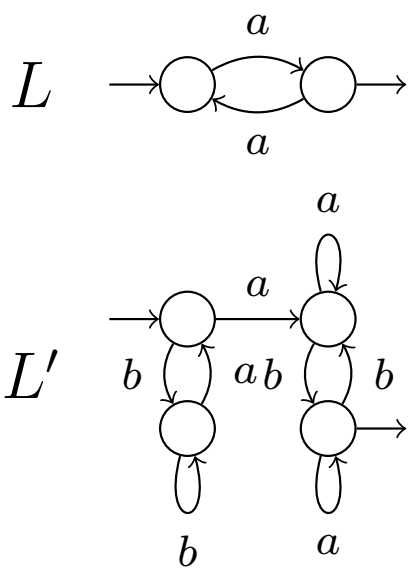
Can L be separated from L' with an **LT** language?



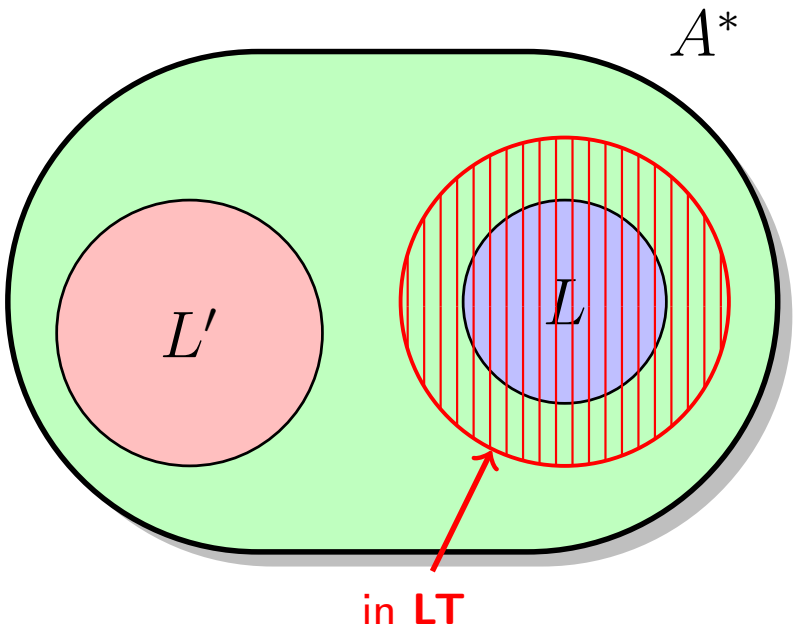
Here comes Separation

Decide the following problem:

Take two regular languages L, L'



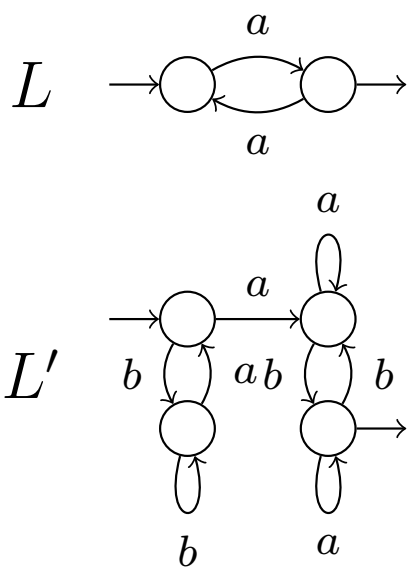
Can L be separated from L' with an **LT** language?



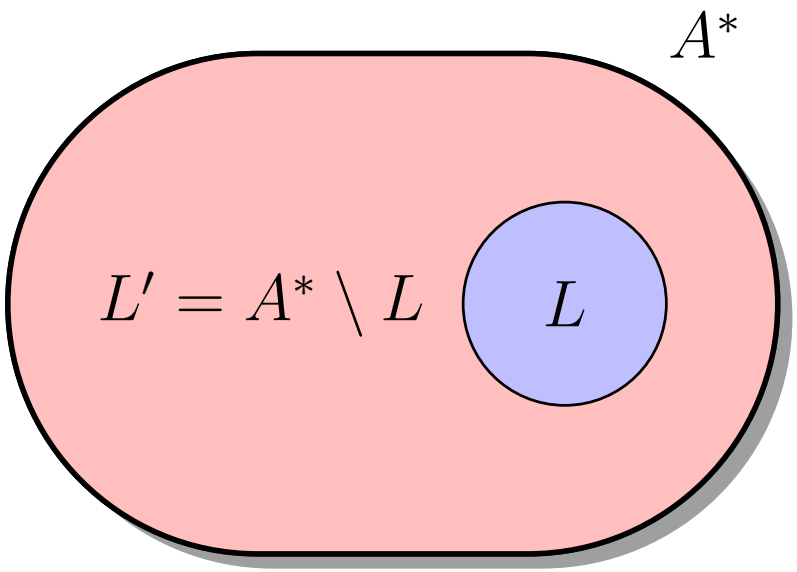
Here comes Separation

Decide the following problem:

Take two regular languages L, L'



Can L be separated from L' with an **LT** language?

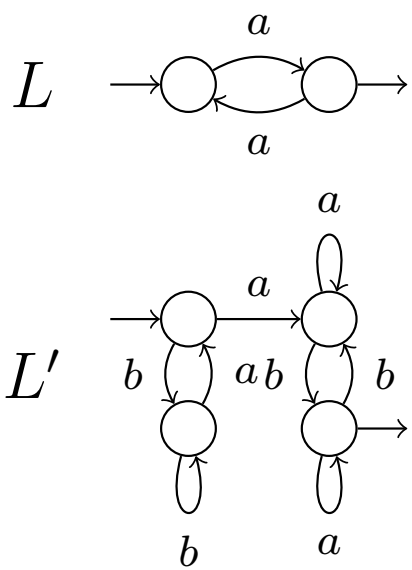


Separation is more general than characterization: $L' = A^* \setminus L$

Here comes Separation

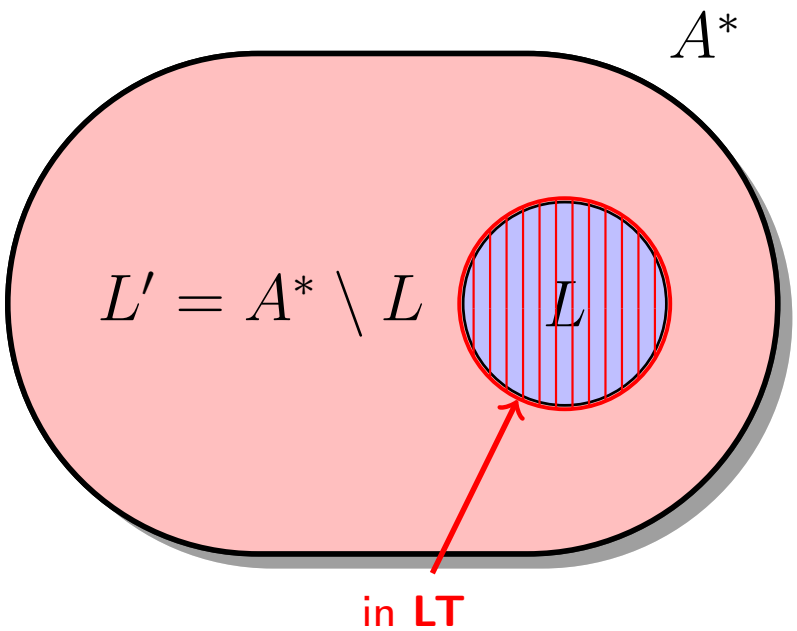
Decide the following problem:

Take two regular languages L, L'



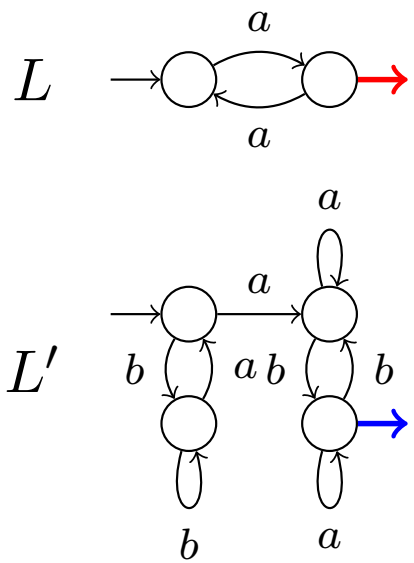
$\Rightarrow$

Can L be separated from L' with an **LT** language?

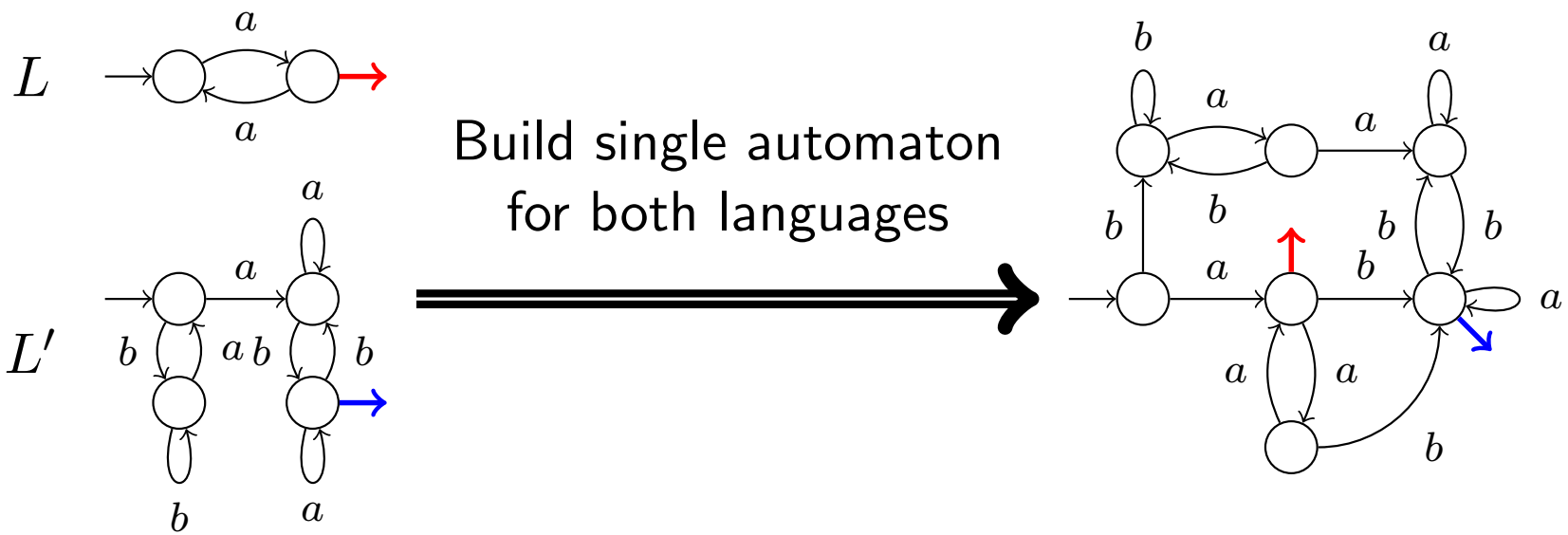


Separation is more general than characterization: $L' = A^* \setminus L$

A more General Problem than Characterization



A more General Problem than Characterization



Problem is now

What properties of this automaton
can be expressed as LT languages?

Advantages & Difficulties of Separation

- Need techniques applying to **all** languages, not just to LT ones
  $\Rightarrow$ More general results, usable in broader context.

Advantages & Difficulties of Separation

- Need techniques applying to **all** languages, not just to LT ones
  $\Rightarrow$ More general results, usable in broader context.

- We do not have the separator in hand.
  $\Rightarrow$ No canonical tool (like the syntactic monoid).

Advantages & Difficulties of Separation

- Need techniques applying to **all** languages, not just to LT ones
  $\Rightarrow$ More general results, usable in broader context.

- We do not have the separator in hand.
  $\Rightarrow$ No canonical tool (like the syntactic monoid).

- Computational complexity can be higher than for membership.

Advantages & Difficulties of Separation

- Need techniques applying to **all** languages, not just to LT ones
  $\Rightarrow$ More general results, usable in broader context.

- We do not have the separator in hand.
  $\Rightarrow$ No canonical tool (like the syntactic monoid).

- Computational complexity can be higher than for membership.

$\looparrowright$ Many classes already solved for separation, including LT, LTT but involved algebraic+topological, **non-constructive** proofs.

Separation by LT languages

Locally Testable Languages

LT = Boolean combinations of

$$uA^*, \qquad A^*u, \qquad A^*uA^*$$

Membership based on presence/absence of suffixes/prefixes/infixes.

Example $A = \{a, b\}$

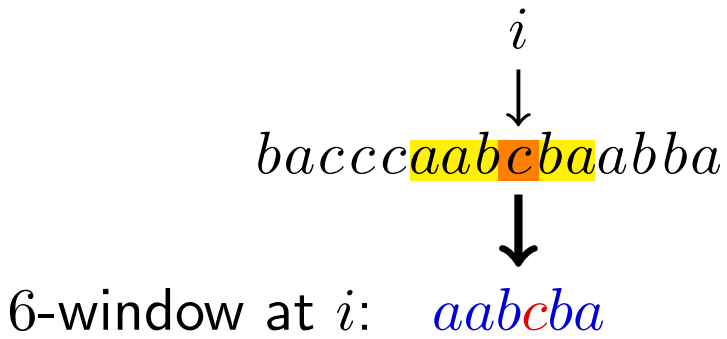
$$(ab)^+ \quad = \quad aA^* \quad \cap \quad A^*b \quad \cap \quad A^* \setminus (A^*aaA^* \cup A^*bbA^*)$$

$\#^*a\#^*b\#^*$ is not LT.

$\#^k a\#^k b\#^k$ and $\#^k b\#^k a\#^k$ have same "windows" of size k.

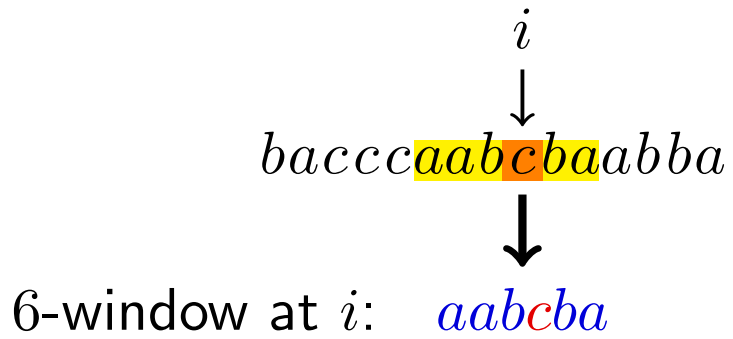
k-windows

- k-window at position i:



k-windows

- k-window at position i:



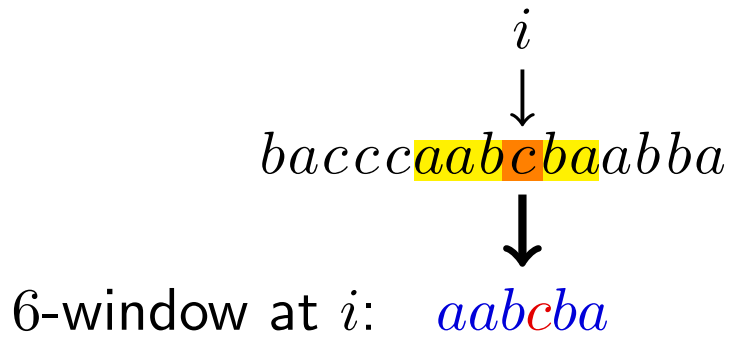
- LT languages determined by (dis)allowed k-windows.

  For $(ab)^+$, take $k = 2$.

$$(ab)^+ \quad = \quad aA^* \quad \cap \quad A^*b \quad \cap \quad A^* \setminus (A^*aaA^* \cup A^*bbA^*)$$

k-windows

- k-window at position i:



- LT languages determined by (dis)allowed k-windows.

For $(ab)^+$, take $k = 2$.

$$(ab)^+ \quad = \quad aA^* \quad \cap \quad A^*b \quad \cap \quad A^* \setminus (A^*aaA^* \cup A^*bbA^*)$$

$u \equiv_k w$ if u, w have the same k-windows.

L is LT $\iff$ L is a union of $\equiv_k$-classes.

Separation for LT

- $\mathsf{LT}[k] = \Big\{ L \mid L \text{ is a union of } \equiv_k\text{-classes} \Big\}$.

$$\mathsf{LT} = \bigcup_k \mathsf{LT}[k].$$

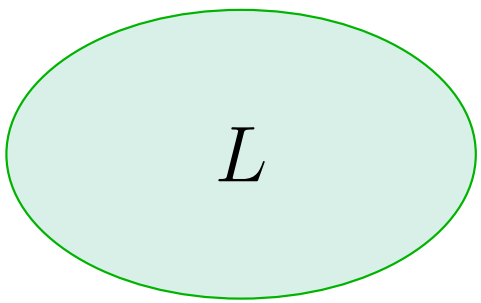
- Deciding $\mathsf{LT}[k]$ separation of regular languages for fixed k?

Separation for LT

- $\mathsf{LT}[k] = \Big\{ L \mid L \text{ is a union of } \equiv_k\text{-classes} \Big\}$.

$$\mathsf{LT} = \bigcup_k \mathsf{LT}[k].$$

- Deciding $\mathsf{LT}[k]$ separation of regular languages for fixed k?
  Easy: $[L]_{\equiv_k}$ = smallest $\mathsf{LT}[k]$ language containing L.

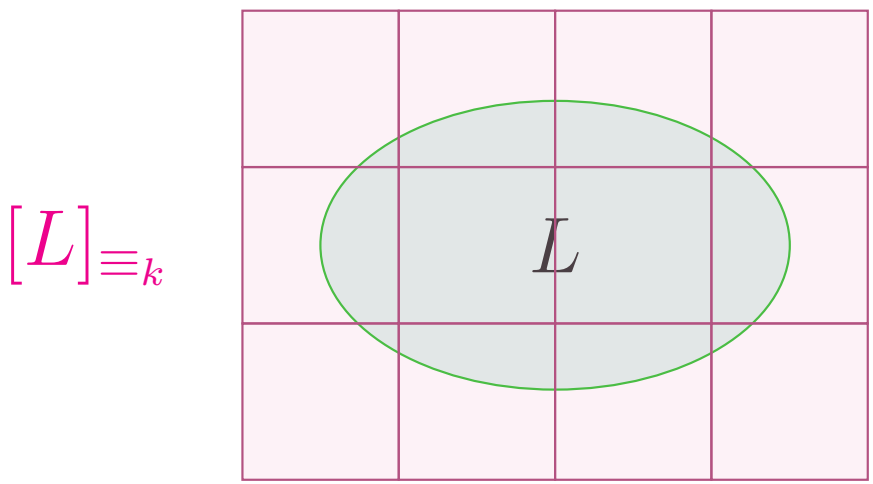


Separation for LT

- $\mathsf{LT}[k] = \Big\{ L \mid L \text{ is a union of } \equiv_k\text{-classes} \Big\}$.

$$\mathsf{LT} = \bigcup_k \mathsf{LT}[k].$$

- Deciding $\mathsf{LT}[k]$ separation of regular languages for fixed k?
  Easy: $[L]_{\equiv_k}$ = smallest $\mathsf{LT}[k]$ language containing L.

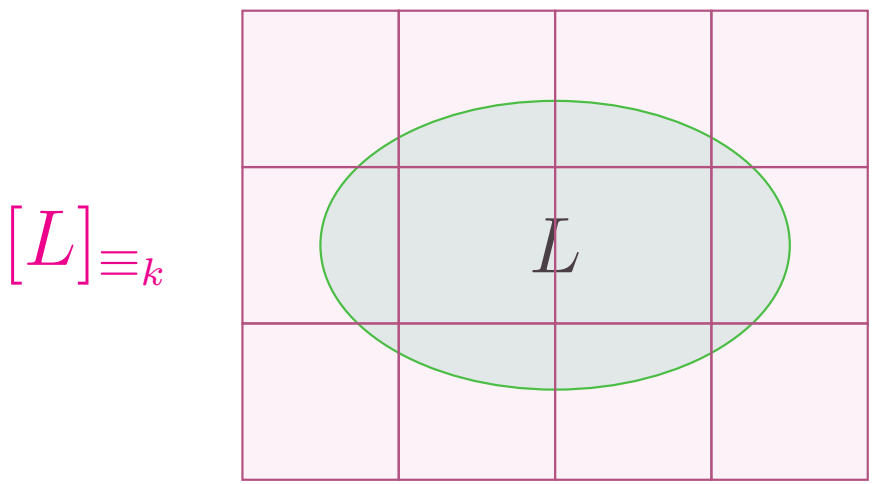


Separation for LT

- $\mathsf{LT}[k] = \Big\{ L \mid L \text{ is a union of } \equiv_k\text{-classes} \Big\}$.

$$\mathsf{LT} = \bigcup_k \mathsf{LT}[k].$$

- Deciding $\mathsf{LT}[k]$ separation of regular languages for fixed k?
  Easy: $[L]_{\equiv_k}$ = smallest $\mathsf{LT}[k]$ language containing L.



- For LT: no such smallest LT language.

Separation for LT: Proof ideas

- Semi-algorithm: enumerate all LT languages K and test

$$L \quad \subseteq \quad K \quad \subseteq \quad A^* \setminus L'$$

Separation for LT: Proof ideas

- Semi-algorithm: enumerate all LT languages K and test

$$L \quad \subseteq \quad K \quad \subseteq \quad A^* \setminus L'$$

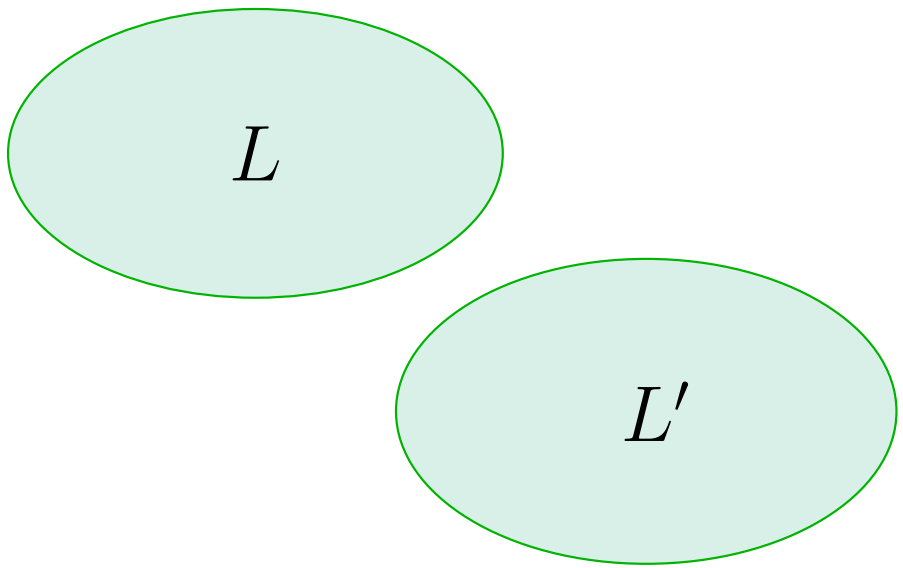
- Need semi-algorithm to witness for non-separability.
- Restricting to a finite number of separators $\rightsquigarrow$ decidability.

Witnesses of non-separability

$u \equiv_k w$ if u, w have the same k-windows.

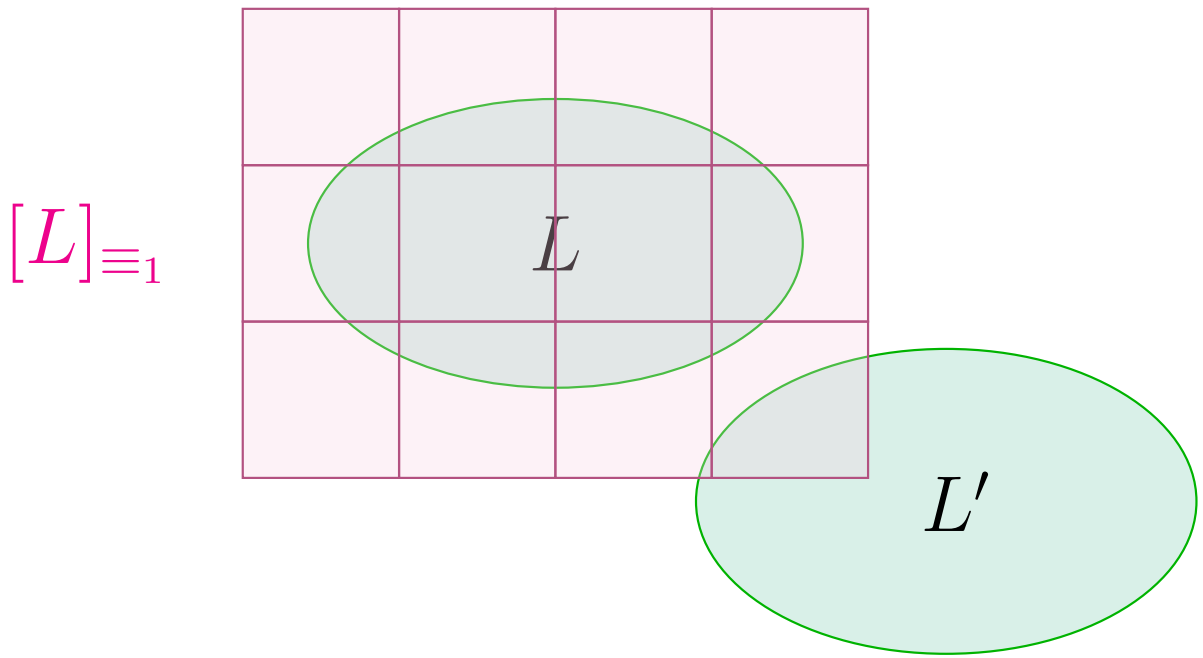
Witnesses of non-separability

$u \equiv_k w$ if u, w have the same k-windows.



Witnesses of non-separability

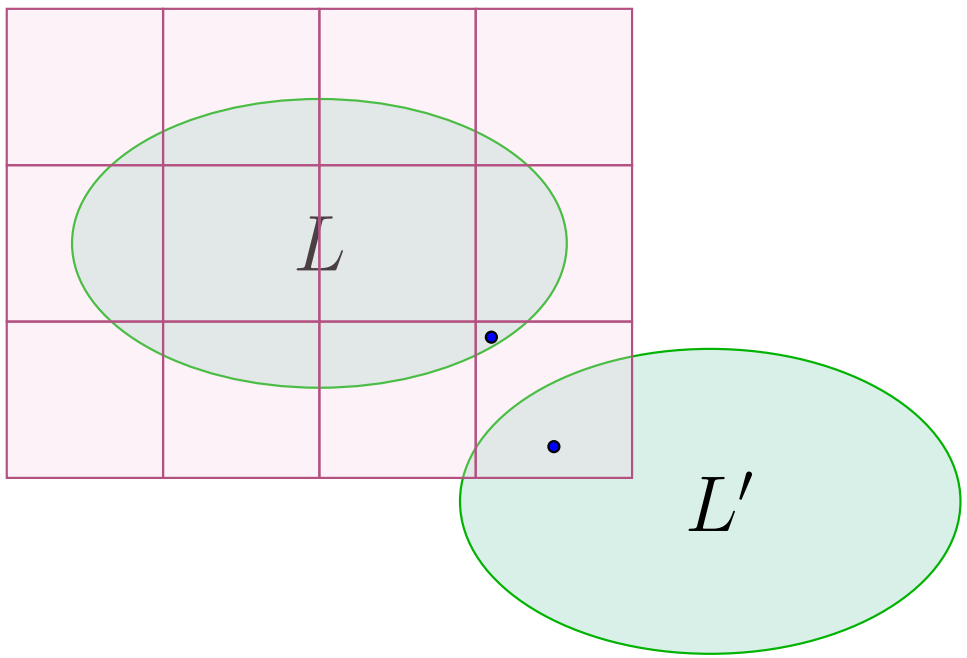
$u \equiv_k w$ if u, w have the same k-windows.



Witnesses of non-separability

$u \equiv_k w$ if u, w have the same k-windows.

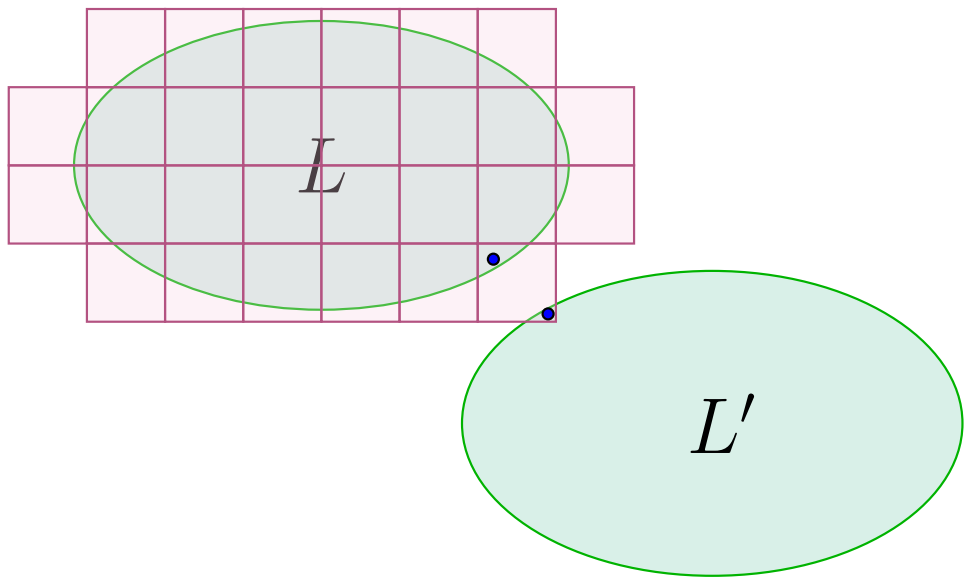
$[L]_{\equiv_1}$



Witnesses of non-separability

$u \equiv_k w$ if u, w have the same k-windows.

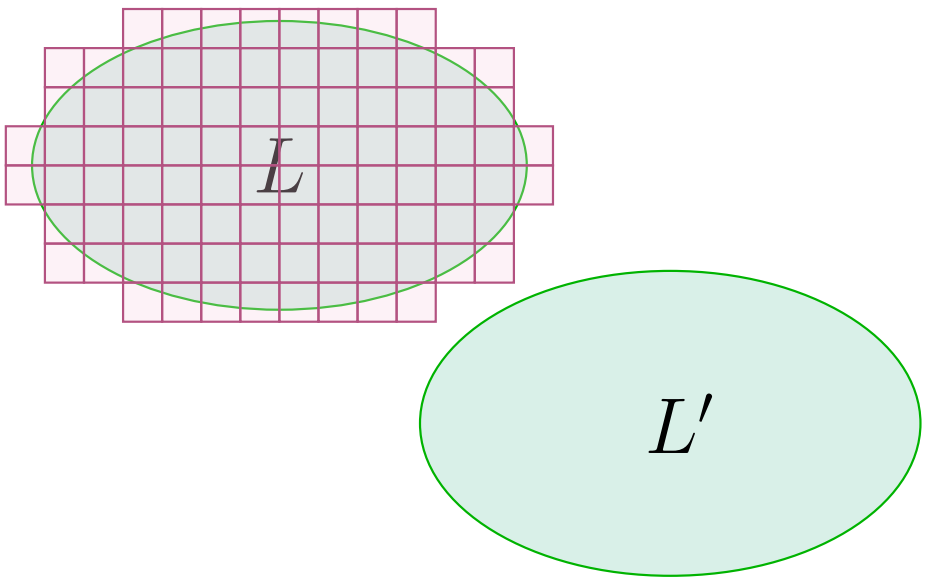
$[L]_{\equiv_2}$



Witnesses of non-separability

$u \equiv_k w$ if u, w have the same k-windows.

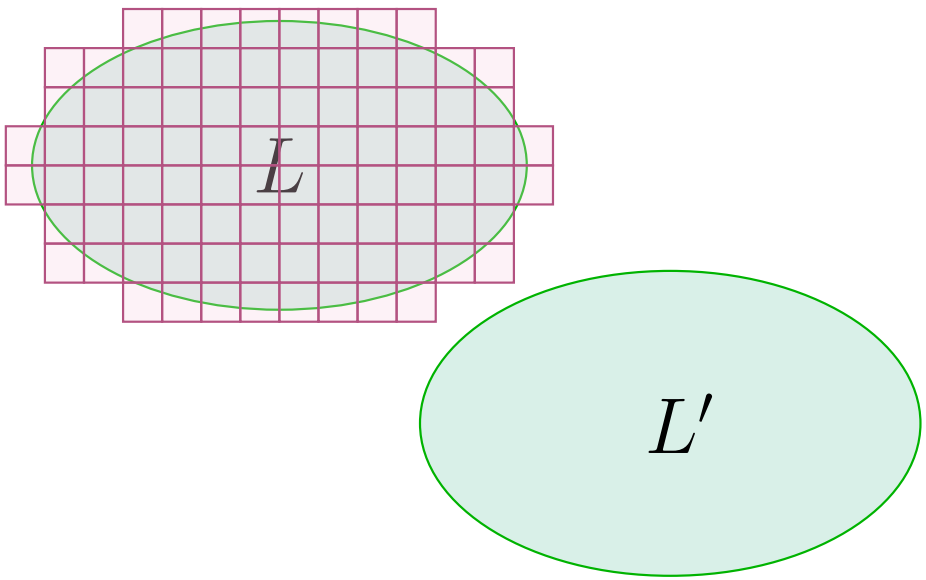
$[L]_{\equiv_3}$



Witnesses of non-separability

$u \equiv_k w$ if u, w have the same k-windows.

$[L]_{\equiv_3}$



When do we stop?

Bounding the window

- L, L' are not separable iff.

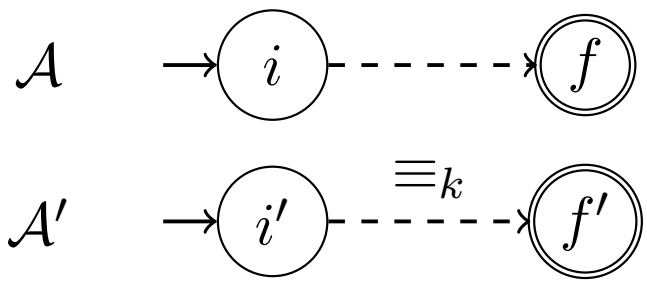
$$\forall k \qquad \exists w \in L,\ \exists w' \in L' \qquad w \equiv_k w'.$$

Bounding the window

- L, L' are not separable iff.

$$\forall k \qquad \exists w \in L,\ \exists w' \in L' \qquad w \equiv_k w'.$$

- Abstraction k-witnesses of non-separability: tuples (i, f, i', f')



- L, L' are k-separable $\iff$ $\{k\text{-witnesses}\} = \varnothing$.
- $\{k+1\text{-witnesses}\} \subseteq \{k\text{-witnesses}\}$.
- Stabilization index?

k-loops

- **Main technique for LT**:

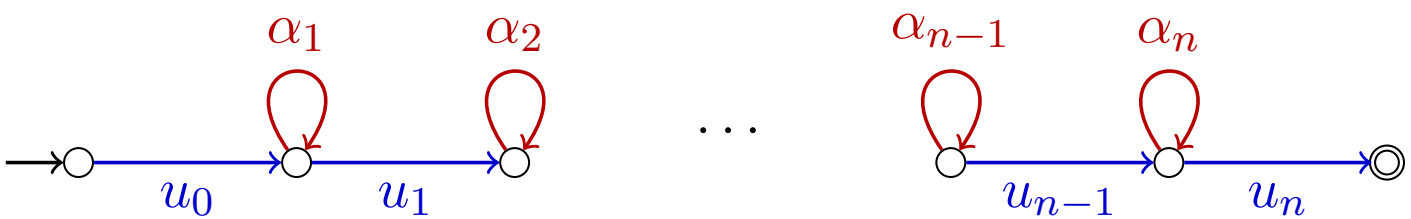
$$\begin{cases} k \text{ large} \\ w \in L, \quad w' \in L' \\ w \equiv_k w' \end{cases} \quad \Longrightarrow \quad \begin{cases} \text{for all } \ell, \text{ build} \\ z \in L, \quad z' \in L' \\ z \equiv_\ell z' \end{cases}$$

k-loops

- **Main technique for LT**:

$$\begin{cases} k \text{ large} \\ w \in L, \quad w' \in L' \\ w \equiv_k w' \end{cases} \implies \begin{cases} \text{for all } \ell, \text{ build} \\ z \in L, \quad z' \in L' \\ z \equiv_\ell z' \end{cases}$$

- z, z' built by pumping infixes labeling loops in $\mathcal{A}$, $\mathcal{A}'$.
- k-loop = infix that can be pumped
  - labeling loops in $\mathcal{A}$, $\mathcal{A}'$,
  - occurring inside $k/2$-window.



k-loops

- **Main technique for LT**:

$$\begin{cases} k \text{ large} \\ w \in L, \quad w' \in L' \\ w \equiv_k w' \end{cases} \implies \begin{cases} \text{for all } \ell, \text{ build} \\ z \in L, \quad z' \in L' \\ z \equiv_\ell z' \end{cases}$$

- z, z' built by pumping infixes labeling loops in $\mathcal{A}$, $\mathcal{A}'$.
- k-loop = infix that can be pumped
  - labeling loops in $\mathcal{A}$, $\mathcal{A}'$,
  - occurring inside $k/2$-window.

Lemma (Pumping)

If $k > 4|Q||Q'|$, $\exists$ *a* k*-loop within* $k/4$ *consecutive positions.*

k-loops

Building $z \in L, z' \in L'$ such that $z \equiv_\ell z'$

- Insert smallest k-loops in w and w' whenever possible.

$$w = u_0 \cdot u_1 \dots u_{n-1} \cdot u_n$$
$$z = u_0 \cdot \alpha_1^\ell \cdot u_1 \alpha_2^\ell \cdots \alpha_{n-1}^\ell \cdot u_{n-1} \cdot \alpha_n^\ell \cdot u_n$$

and same for w' with u'_i, α'_i.

k-loops

Building $z \in L, z' \in L'$ such that $z \equiv_\ell z'$

- Insert smallest k-loops in w and w' whenever possible.

$$w = u_0 \cdot u_1 \dots u_{n-1} \cdot u_n$$
$$z = u_0 \cdot \alpha_1^\ell \cdot u_1 \alpha_2^\ell \cdots \alpha_{n-1}^\ell \cdot u_{n-1} \cdot \alpha_n^\ell \cdot u_n$$

and same for w' with u'_i, α'_i.

Lemma (Recall: $k > 4|Q||Q'|$)

1. $w \equiv_k w' \implies$ *Same sets of triples* $\{(\alpha_i, u_i, \alpha_{i+1}) \mid i\}$, *and* $\{(\alpha'_j, u'_j, \alpha'_{j+1}) \mid j\}$.
2. *Therefore, z and z' are not distinguishable by $\equiv_\ell$.*

Algorithms

- Alg 1: Test all separators up to window size $k = 4|Q|.|Q'| + 1$.
- Alg 2: Test for common pattern occurring in both $\mathcal{A}$ and $\mathcal{A}'$

with same tuples $(\alpha_i, u_i, \alpha_{i+1})$, (u_0, α_1), (α_p, u_p).

Theorem

1. *L, L' not LT-separatable* **iff** *$\mathcal{A}$, $\mathcal{A}'$ have a common pattern.*
2. *Common pattern can be detected in* NEXPTIME.
3. *Common pattern detection is* NP*-hard*
   Compare with: testing if L is in LT is PTIME.

Locally Threshold Testable Languages

- LTT = LT + ability to **count** infixes up to threshold.

  Example At least 7 occurrences of ab.
- $u \equiv_k^d w$ if u, w have same **number** of k-windows, up to thr. d.

Locally Threshold Testable Languages

- LTT = LT + ability to **count** infixes up to threshold.

  Example At least 7 occurrences of ab.
- $u \equiv_k^d w$ if u, w have same **number** of k-windows, up to thr. d.

- For fixed d, **previous bound on** k still works.
- Existence of threshold can be expressed in Presburger logic.

Locally Threshold Testable Languages

- LTT = LT + ability to **count** infixes up to threshold.

  Example At least 7 occurrences of ab.
- $u \equiv_k^d w$ if u, w have same **number** of k-windows, up to thr. d.

- For fixed d, **previous bound on** k still works.
- Existence of threshold can be expressed in Presburger logic.

Theorem

1. *L, L' not LTT-separatable* **iff** *$\mathcal{A}$, $\mathcal{A}'$ have a common pattern.*
2. *A common pattern can be detected in* 2-NEXPSPACE.
3. *Common pattern detection is* NP*-hard*
   Compare with: testing if L is in LTT is PTIME.

Conclusion

Summary

- Separability by LT and LTT is decidable, elementary tools.
- Complexity bounds (not tight).
- Some more results for context-free input languages.

Recent and further work

- Same scheme can be adapted for Sep $=$ B$\Sigma_1(<)$, FO$^2(<)$, FO
- Other important classes, eg, quantifier alternation hierarchies.
- Other structures: eg, trees, VPAs.
- Generic results to transfer decidability of separability.
- Links between membership and separation.