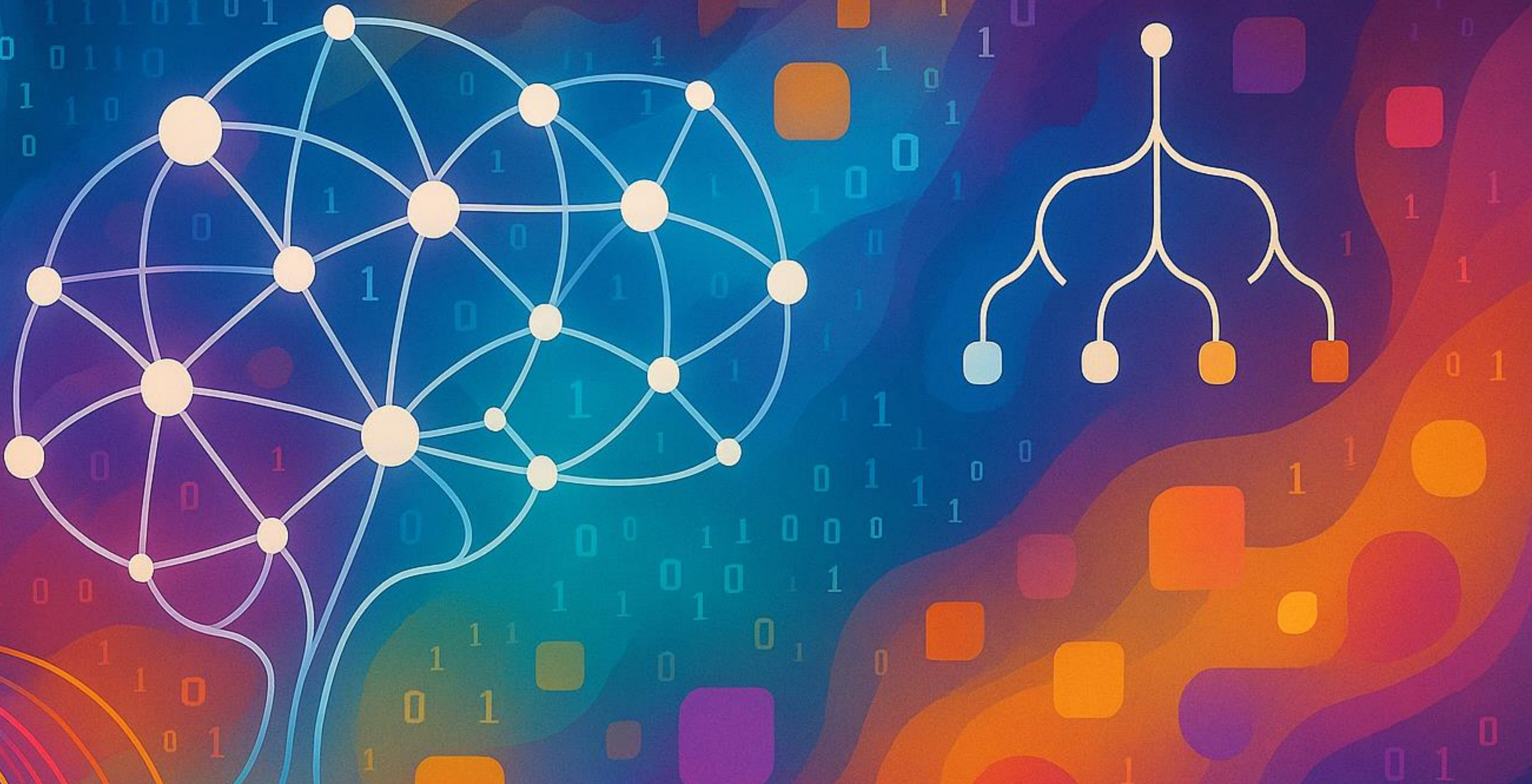


# Tutorial on Language Generation

***Organizers:*** *Moses Charikar, Anay Mehrotra, Charlotte Peale, Chirag Pabbaraju, Grigoris Velegkas*

# Validity–Breadth Trade-Off (Part II)

**This Talk:**

- Lower Bound Proof for "Uniqueness Property"
- Algorithm for Achieving Breadth Infinitely Often

# Lower Bound on Breadth

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound on Breadth

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound on Breadth

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

*Uniqueness condition*: if a generator satisfies $P$ for some $L$, it cannot satisfy $P$ for some other $L'$

# Angluin's Condition for Identification

**Angluin's Condition**[Angluin'80]:

Any collection $\mathcal{L}$ is identifiable in the limit iff every $L \in \mathcal{L}$ has a *finite* tell-tale subset $T_L \subseteq L$, i.e.,

# Angluin's Condition for Identification

**Angluin's Condition** [Angluin'80]:

Any collection $\mathcal{L}$ is identifiable in the limit iff every $L \in \mathcal{L}$ has a *finite* tell-tale subset $T_L \subseteq L$, i.e.,

for any $L' \neq L$ if $T_L \subseteq L'$ then $L'$ is *not* a proper subset of $L$

# Angluin's Condition for Identification

**Angluin's Condition** [Angluin'80]:

Any collection $\mathcal{L}$ is identifiable in the limit iff every $L \in \mathcal{L}$ has a *finite* tell-tale subset $T_L \subseteq L$, i.e.,

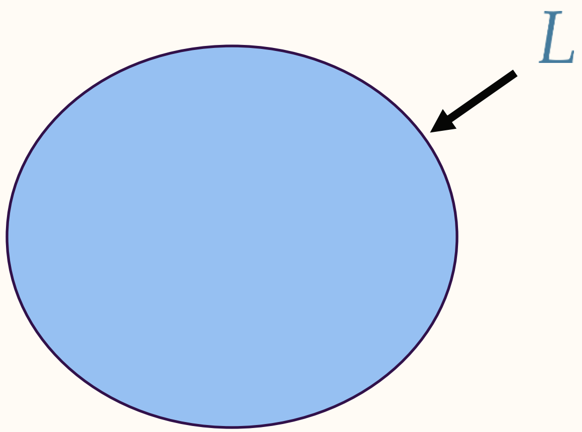
for any $L' \neq L$ if $T_L \subseteq L'$ then $L'$ is *not* a proper subset of $L$

# Angluin's Condition for Identification

**Angluin's Condition** [Angluin'80]:

Any collection $\mathcal{L}$ is identifiable in the limit iff every $L \in \mathcal{L}$ has a *finite* tell-tale subset $T_L \subseteq L$, i.e.,

for any $L' \neq L$ if $T_L \subseteq L'$ then $L'$ is *not* a proper subset of $L$

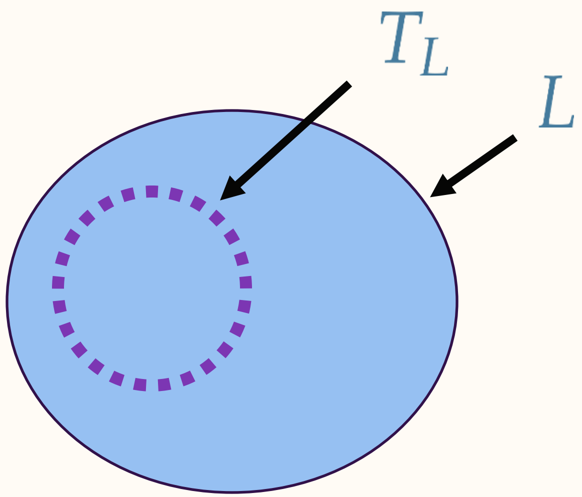


# Angluin's Condition for Identification

**Angluin's Condition** [Angluin'80]:

Any collection $\mathcal{L}$ is identifiable in the limit iff every $L \in \mathcal{L}$ has a *finite* tell-tale subset $T_L \subseteq L$, i.e.,

for any $L' \neq L$ if $T_L \subseteq L'$ then $L'$ is *not* a proper subset of $L$

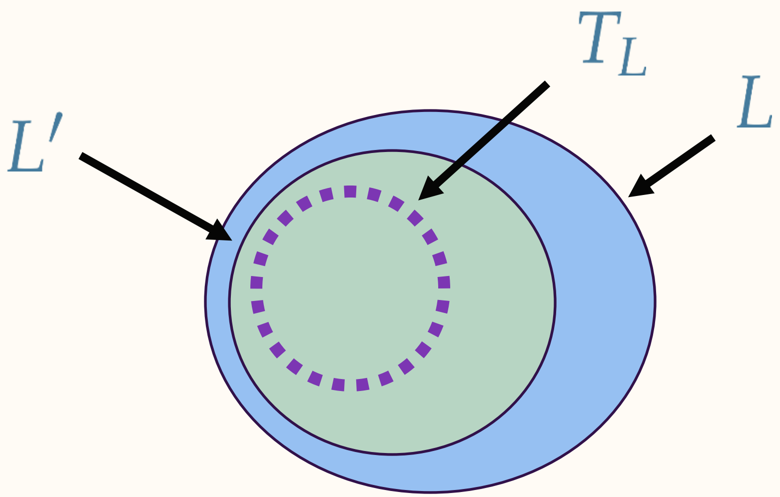


# Angluin's Condition for Identification

**Angluin's Condition** [Angluin'80]:

Any collection $\mathcal{L}$ is identifiable in the limit iff every $L \in \mathcal{L}$ has a *finite* tell-tale subset $T_L \subseteq L$, i.e.,

for any $L' \neq L$ if $T_L \subseteq L'$ then $L'$ is *not* a proper subset of $L$

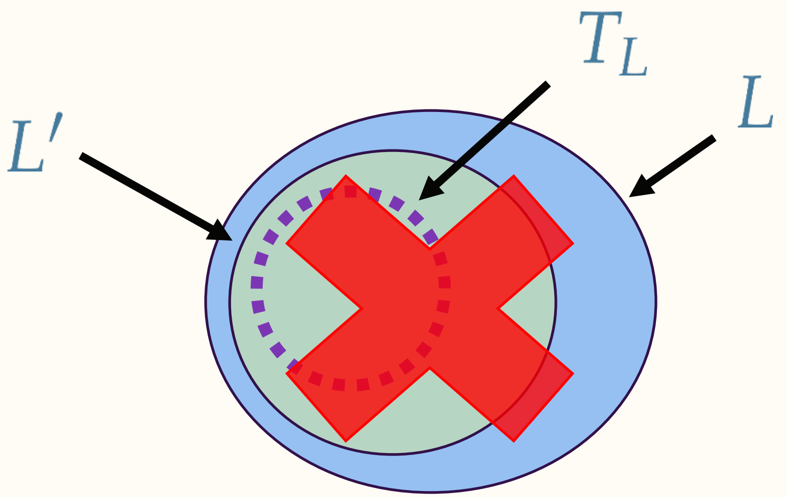


# Angluin's Condition for Identification

**Angluin's Condition** [Angluin'80]:

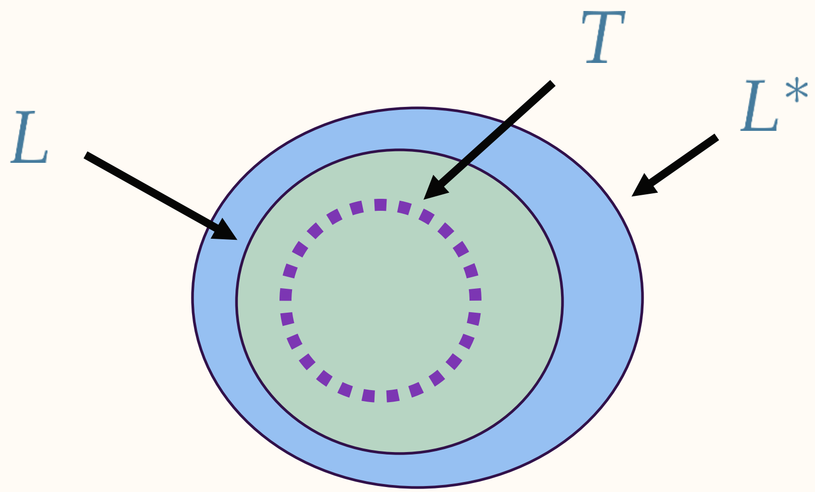
Any collection $\mathcal{L}$ is identifiable in the limit iff every $L \in \mathcal{L}$ has a *finite* tell-tale subset $T_L \subseteq L$, i.e.,

for any $L' \neq L$ if $T_L \subseteq L'$ then $L'$ is *not* a proper subset of $L$



# Angluin's Condition for Identification

- If $L^*$ doesn't have a tell-tale, for every $T$ (finite subset of $L^*$), there is some $L$ such that: i) $T$ is subset of $L$, and ii) $L$ is proper subset of $L^*$



# High Level Proof Structure

# High Level Proof Structure

- Inspiration from Gold's negative result for identification

# High Level Proof Structure

- Inspiration from Gold's negative result for identification
- Goal: given a generator choose enumeration $E$ and target $K$ s.t. generator doesn't achieve $P$ infinitely often

# High Level Proof Structure

- Inspiration from Gold's negative result for identification
- Goal: given a generator choose enumeration $E$ and target $K$ s.t. generator doesn't achieve $P$ infinitely often
- Collection isn't identifiable, there's $L^*$ violating Angluin's condition

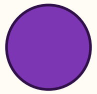
# High Level Proof Structure

- Inspiration from Gold's negative result for identification
- Goal: given a generator choose enumeration $E$ and target $K$ s.t. generator doesn't achieve $P$ infinitely often
- Collection isn't identifiable, there's $L^*$ violating Angluin's condition
- $E$, $K$ are chosen via *diagonalization* by applying the (negation of) Angluin's condition on $L^*$ repeatedly

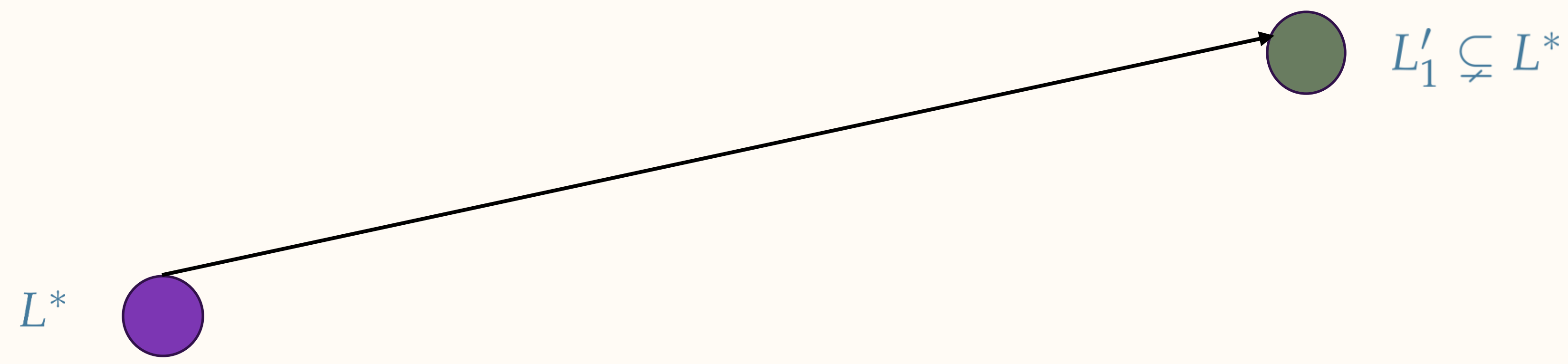
# High Level Proof Structure

# High Level Proof Structure

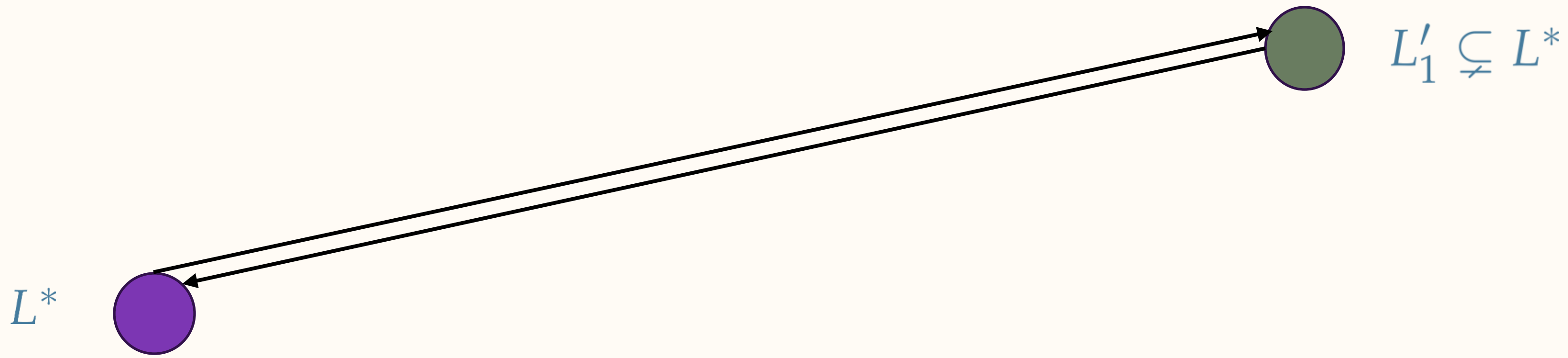
$L^*$



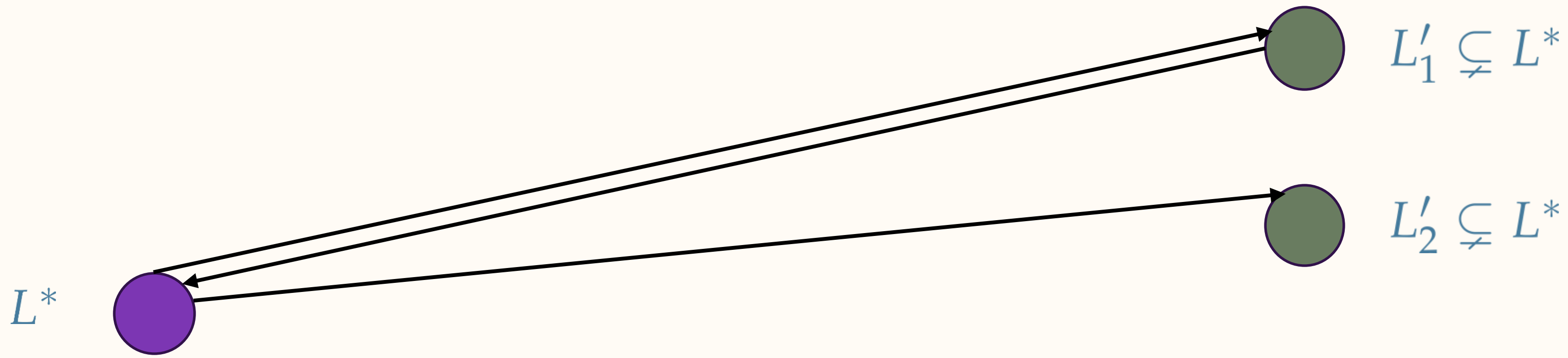
# High Level Proof Structure



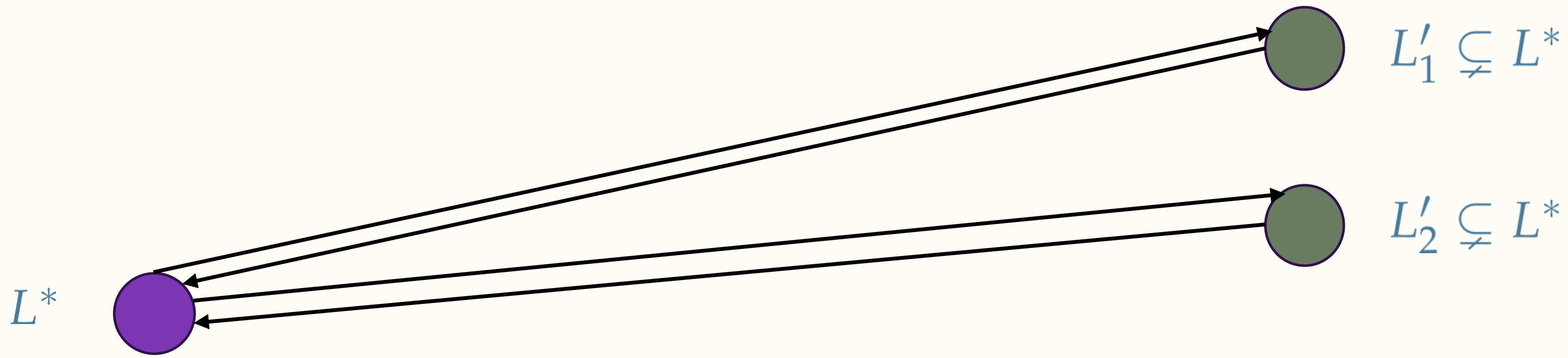
# High Level Proof Structure



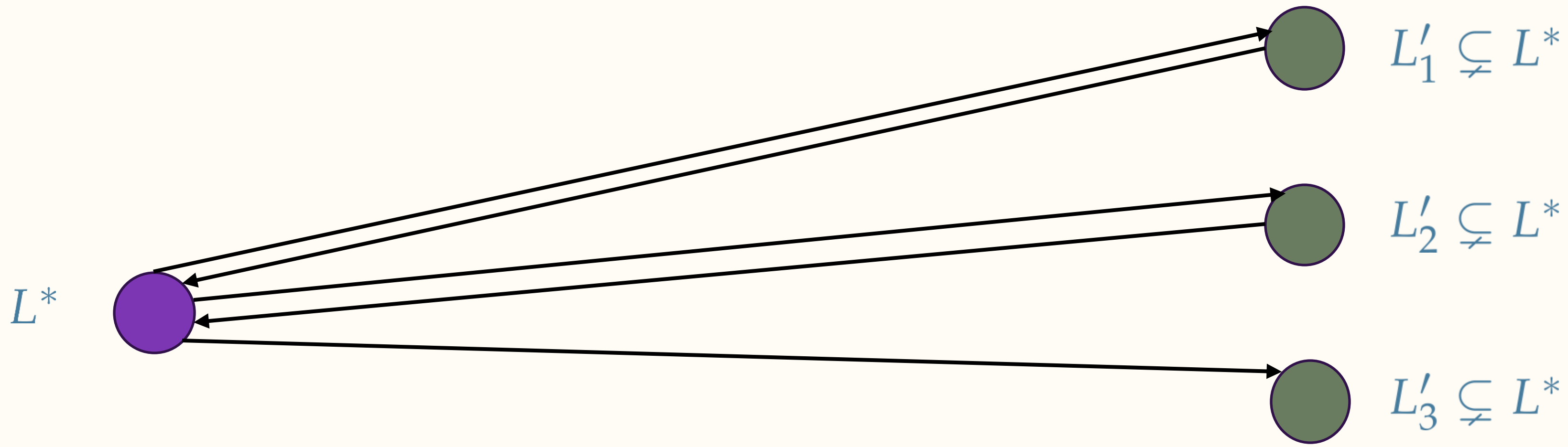
# High Level Proof Structure



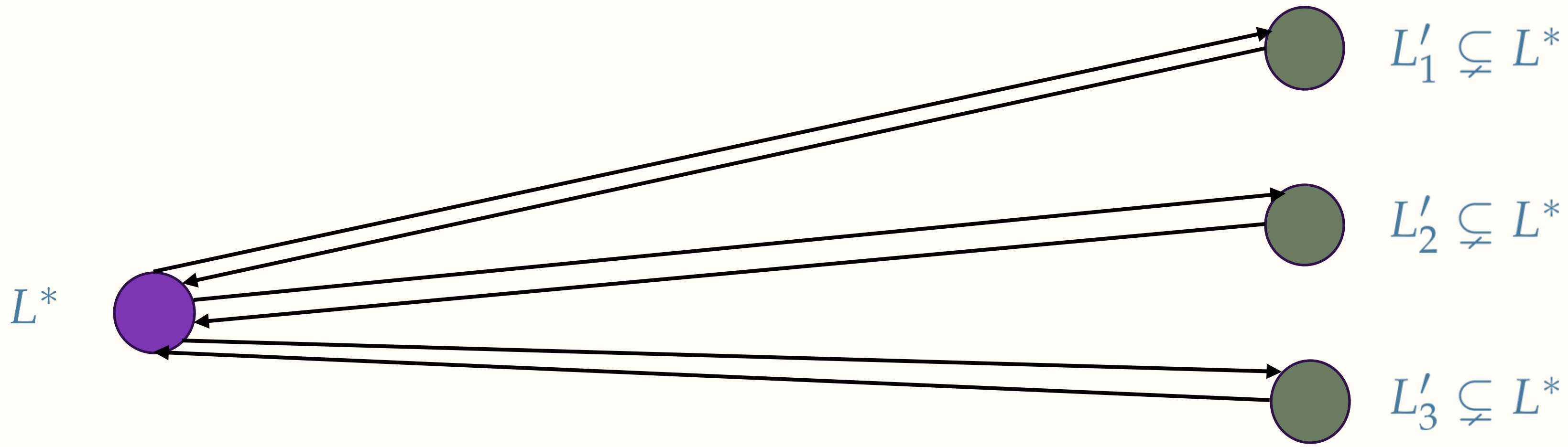
# High Level Proof Structure



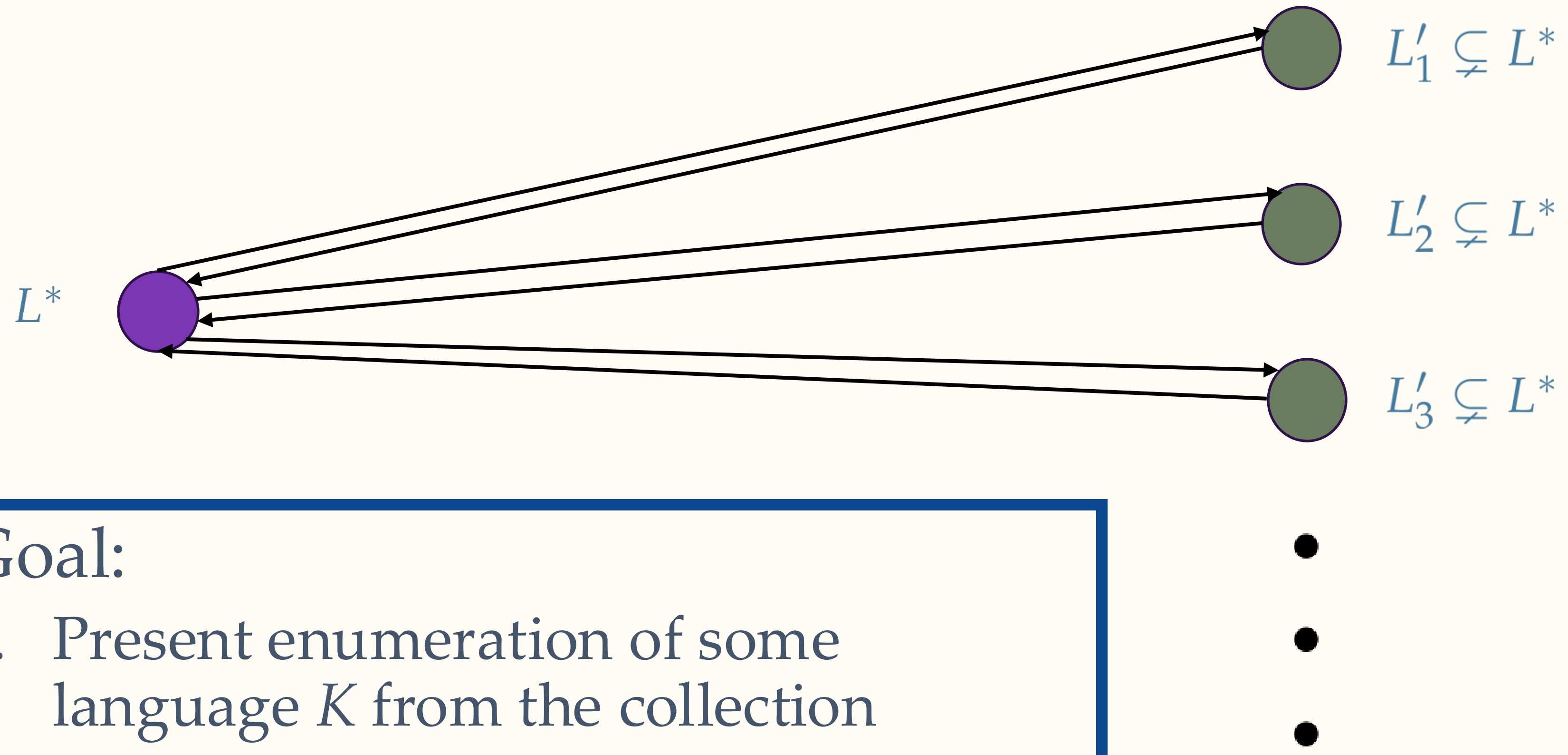
# High Level Proof Structure



# High Level Proof Structure



# High Level Proof Structure




- Goal:
  1. Present enumeration of some language $K$ from the collection
  2. Make the generator miss $P$ infinitely often

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- Since $\mathcal{L}$ isn't identifiable there's $L^*$ that violates Angluin's condition

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- Since $\mathcal{L}$ isn't identifiable there's $L^*$ that violates Angluin's condition
- Adversary starts enumerating $L^*$

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- Since $\mathcal{L}$ isn't identifiable there's $L^*$ that violates Angluin's condition

- Adversary starts enumerating $L^*$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $E^*$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | … |

**Lower Bound on Generation with Breadth** [CP’25], [KMV’24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn’t identifiable, no generator can achieve $P$.

# Lower Bound Proof

- Since $\mathcal{L}$ isn’t identifiable there’s $L^*$ that violates Angluin’s condition

- Adversary starts enumerating $L^*$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $E^*$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | … |

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- Since $\mathcal{L}$ isn't identifiable there's $L^*$ that violates Angluin's condition

- Adversary starts enumerating $L^*$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $E^*$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | ... |

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- Since $\mathcal{L}$ isn't identifiable there's $L^*$ that violates Angluin's condition

- Adversary starts enumerating $L^*$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $E^*$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | ... |

- If generator never satisfies $P$ for $L^*$ then lower bound holds

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- Since $\mathcal{L}$ isn't identifiable there's $L^*$ that violates Angluin's condition

- Adversary starts enumerating $L^*$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $E^*$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $\dots$ |

- If generator never satisfies $P$ for $L^*$ then lower bound holds

- Generator achieves $P$ at $t_1$ for $L^*$ ; let $S_{t_1}$ be enumerated elements

**Lower Bound on Generation with Breadth** [CP’25], [KMV’24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn’t identifiable, no generator can achieve $P$.

# Lower Bound Proof

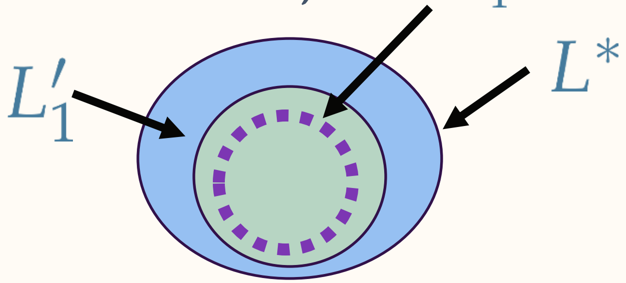
- Since $\mathcal{L}$ isn’t identifiable there’s $L^*$ that violates Angluin’s condition

- Adversary starts enumerating $L^*$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $E^*$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | ... |

- If generator never satisfies $P$ for $L^*$ then lower bound holds

- Generator achieves $P$ at $t_1$ for $L^*$ ; let $S_{t_1}$ be enumerated elements



# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

At time $t_1$

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

At time $t_1$

1. Generator cannot achieve $P$ for $L_1'$ at $t_1$ (uniqueness + strict subset of $L^*$)

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

At time $t_1$

1. Generator cannot achieve $P$ for $L_1'$ at $t_1$ (uniqueness + strict subset of $L^*$)
2. Adversary switches to enumerating $L_1'$ ("valid" move!)

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

## At time $t_1$

1. Generator cannot achieve $P$ for $L'_1$ at $t_1$ (uniqueness + strict subset of $L^*$)
2. Adversary switches to enumerating $L'_1$ ("valid" move!)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $S_{t_1+1}$ | $x_1$ | ... | $x_{t_1}$ | $x_{t_1+1}$ | $x_{t_1+2}$ | $x_{t_1+3}$ | $x_{t_1+4}$ | ... |

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

## At time $t_1$

1. Generator cannot achieve $P$ for $L_1'$ at $t_1$ (uniqueness + strict subset of $L^*$)
2. Adversary switches to enumerating $L_1'$ ("valid" move!)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $S_{t_1+1}$ | $x_1$ | ... | $x_{t_1}$ | $x_{t_1+1}$ | $x_{t_1+2}$ | $x_{t_1+3}$ | $x_{t_1+4}$ | ... |

If generator achieves $P$ at $t_2 > t_1$ for $L_1'$: swap back to $L^*$

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

## At time $t_1$

1. Generator cannot achieve $P$ for $L_1'$ at $t_1$ (uniqueness + strict subset of $L^*$)
2. Adversary switches to enumerating $L_1'$ ("valid" move!)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $S_{t_1+1}$ | $x_1$ | ... | $x_{t_1}$ | $x_{t_1+1}$ | $x_{t_1+2}$ | $x_{t_1+3}$ | $x_{t_1+4}$ | ... |

## If generator achieves $P$ at $t_2 > t_1$ for $L_1'$: swap back to $L^*$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $S_{t_2+1}$ | $x_1$ | ... | $x_{t_1}$ | $\boxed{x_{t_1+1}}$ | $x_{t_1+2}$ | $x_{t_1+3}$ | ... |

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

## At time $t_1$

1. Generator cannot achieve $P$ for $L_1'$ at $t_1$ (uniqueness + strict subset of $L^*$)
2. Adversary switches to enumerating $L_1'$ ("valid" move!)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $S_{t_1+1}$ | $x_1$ | ... | $x_{t_1}$ | $x_{t_1+1}$ | $x_{t_1+2}$ | $x_{t_1+3}$ | $x_{t_1+4}$ | ... |

## If generator achieves $P$ at $t_2 > t_1$ for $L_1'$: swap back to $L^*$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $S_{t_2+1}$ | $x_1$ | ... | $x_{t_1}$ | $\boxed{x_{t_1+1}}$ | $x_{t_1+2}$ | $x_{t_1+3}$ | ... |

1. Generator cannot achieve $P$ for $L^*$ at $t_2$

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

## At time $t_1$

1. Generator cannot achieve $P$ for $L_1'$ at $t_1$ (uniqueness + strict subset of $L^*$)
2. Adversary switches to enumerating $L_1'$ ("valid" move!)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $S_{t_1+1}$ | $x_1$ | ... | $x_{t_1}$ | $x_{t_1+1}$ | $x_{t_1+2}$ | $x_{t_1+3}$ | $x_{t_1+4}$ | ... |

## If generator achieves $P$ at $t_2 > t_1$ for $L_1'$: swap back to $L^*$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $S_{t_2+1}$ | $x_1$ | ... | $x_{t_1}$ | $\boxed{x_{t_1+1}}$ | $x_{t_1+2}$ | $x_{t_1+3}$ | ... |

1. Generator cannot achieve $P$ for $L^*$ at $t_2$
2. This is again a valid move

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- If generator doesn't satisfy $P$ for $L^*$ at $t_3 > t_2$ lower bound witnessed

$$\vdots$$

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- If there are infinitely many phases $K = L^*$, valid enumeration for $L^*$

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- If there are infinitely many phases $K = L^*$, valid enumeration for $L^*$
- If there are $j$ phases $K = L_j'$, valid enumeration for $L_j'$

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24]. If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

- If there are infinitely many phases $K = L^*$, valid enumeration for $L^*$
- If there are $j$ phases $K = L'_j$, valid enumeration for $L'_j$
- Infinitely many steps for which $P$ isn't satisfied!

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- If there are infinitely many phases $K = L^*$ , valid enumeration for $L^*$
- If there are $j$ phases $K = L'_j$ , valid enumeration for $L'_j$
- Infinitely many steps for which $P$ isn't satisfied!
  - Either infinitely many transitions in the "bipartite graph"

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

# Lower Bound Proof

- If there are infinitely many phases $K = L^*$ , valid enumeration for $L^*$
- If there are $j$ phases $K = L'_j$ , valid enumeration for $L'_j$
- Infinitely many steps for which $P$ isn't satisfied!
  - Either infinitely many transitions in the "bipartite graph"
  - Or, if we stop at node, it is never satisfied from some point on for this node

# Lower Bound Proof

**Lower Bound on Generation with Breadth** [CP'25], [KMV'24].
If a notion of breadth $P$ satisfies the *uniqueness condition* and $\mathcal{L}$ isn't identifiable, no generator can achieve $P$.

- If there are infinitely many phases $K = L^*$ , valid enumeration for $L^*$
- If there are $j$ phases $K = L'_j$ , valid enumeration for $L'_j$
- Infinitely many steps for which $P$ isn't satisfied!
  - Either infinitely many transitions in the "bipartite graph"
  - Or, if we stop at node, it is never satisfied from some point on for this node
    - Otherwise the adversary would have moved on

# Algorithm for Generation in the Limit

At step t, only consider $L_1, \ldots, L_t$

$L_{n_t}$ : critical language with highest index $n_t \leq t$

Generate a string from $L_{n_t} \setminus S_t$

# Algorithm for Generation in the Limit

At step t, only consider $L_1, \ldots, L_t$

$L_{n_t}$ : critical language with highest index $n_t \leq t$

Generate a string from $L_{n_t} \setminus S_t$

**Proof sketch:**

For large enough $t$, target language $K$ is critical and in $L_1, \ldots, L_t$

# Algorithm for Generation in the Limit

At step t, only consider $L_1, \ldots, L_t$

$L_{n_t}$ : critical language with highest index $n_t \le t$

Generate a string from $L_{n_t} \setminus S_t$

**Proof sketch:**

For large enough *t*, target language $K$ is critical and in $L_1, \ldots, L_t$

$L_{n_t} \subseteq K$ , so any string from $L_{n_t} \setminus S_t$ also belongs to $K \setminus S_t$

# Algorithm for Generation in the Limit

- Alternative view: the KM algorithm produces indices $i_1, i_2, \ldots$
  - After finite $t$ it satisfies $L_{i_t} \subseteq K$

# Algorithm for Generation in the Limit

- Alternative view: the KM algorithm produces indices $i_1, i_2, \ldots$
  - After finite $t$ it satisfies $L_{i_t} \subseteq K$

- Goal: Modify the KM algorithm so that it achieves the previous and
  - For infinitely many $t$ it satisfies $L_{i_t} = K$

# Algorithm for Generation in the Limit

- Alternative view: the KM algorithm produces indices $i_1, i_2, \ldots$
  - After finite $t$ it satisfies $L_{i_t} \subseteq K$

- Goal: Modify the KM algorithm so that it achieves the previous and
  - For infinitely many $t$ it satisfies $L_{i_t} = K$

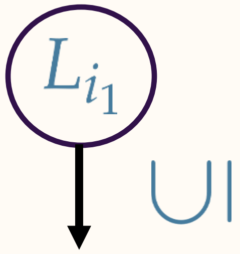
- Idea: The algorithm adds a "backtracking" step in a controlled way

# Chains of Critical Languages

# Chains of Critical Languages

$L_{i_1}$

# Chains of Critical Languages



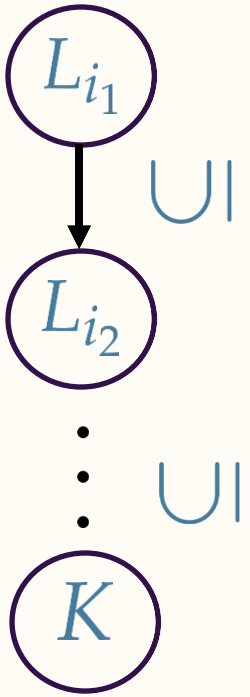
# Chains of Critical Languages

$$
\begin{tikzcd}
L_{i_1} \arrow[d, "\subsetneq"] \\
L_{i_2}
\end{tikzcd}
$$

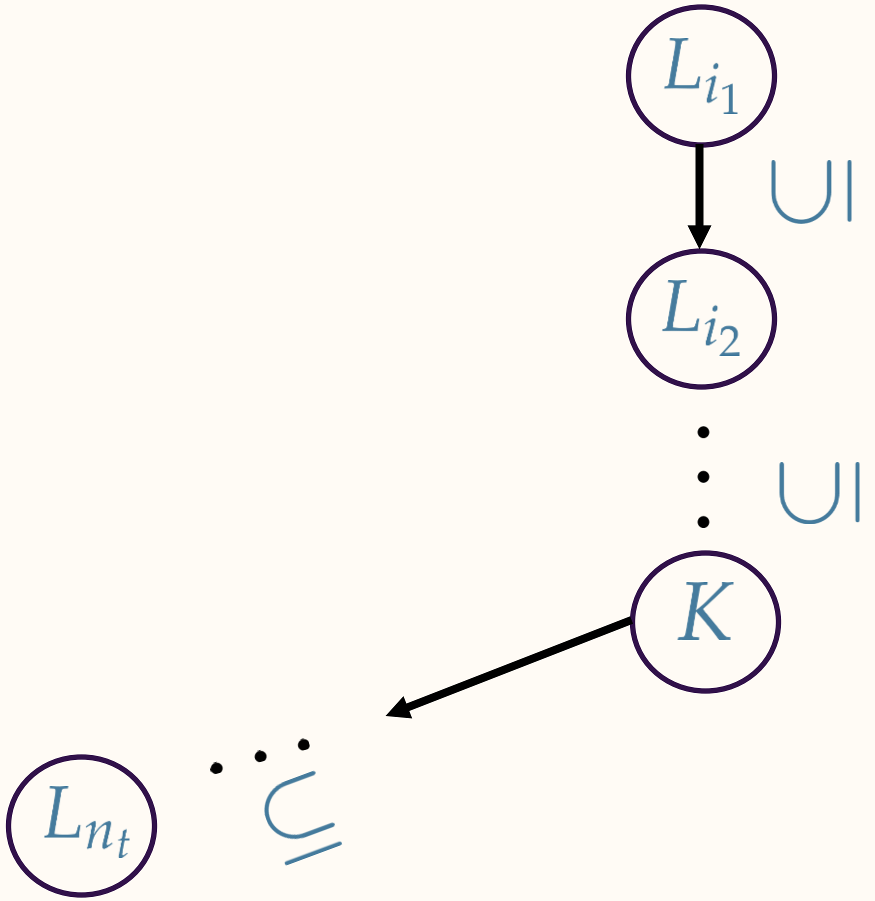
# Chains of Critical Languages

$$\begin{tikzcd}
L_{i_1} \arrow[d, "\rotatebox{-90}{$\supseteq$}"] \\
L_{i_2} \arrow[d, phantom, "\rotatebox{-90}{$\supseteq$}"] \\
\vdots
\end{tikzcd}$$

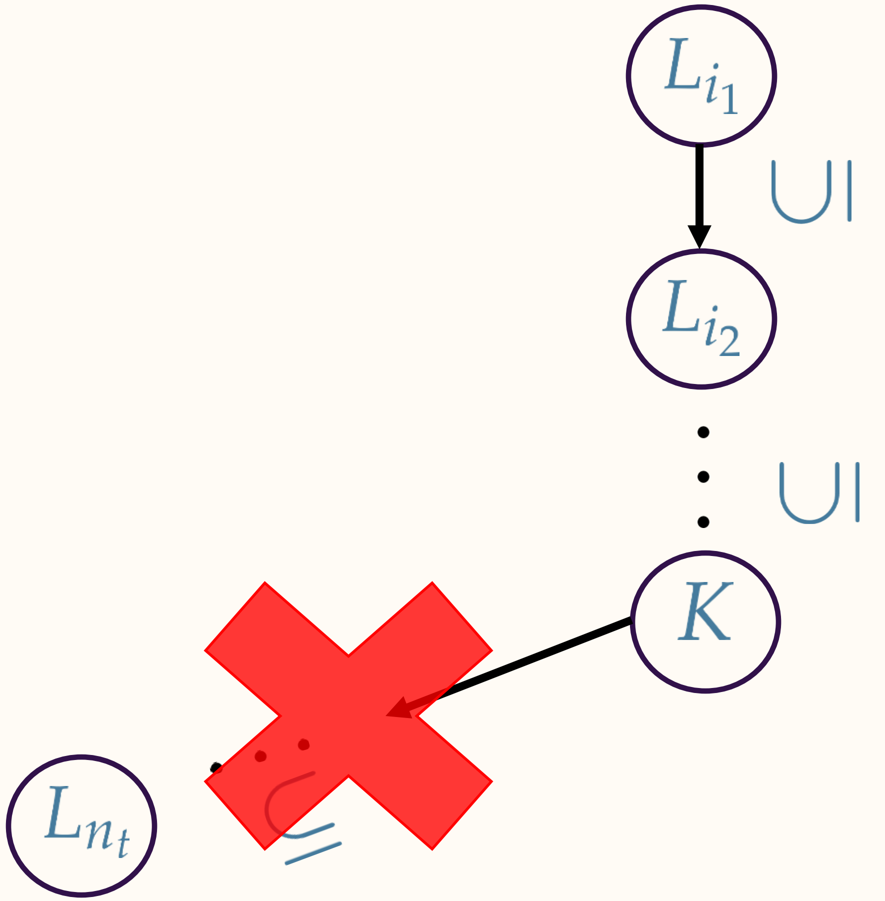
# Chains of Critical Languages



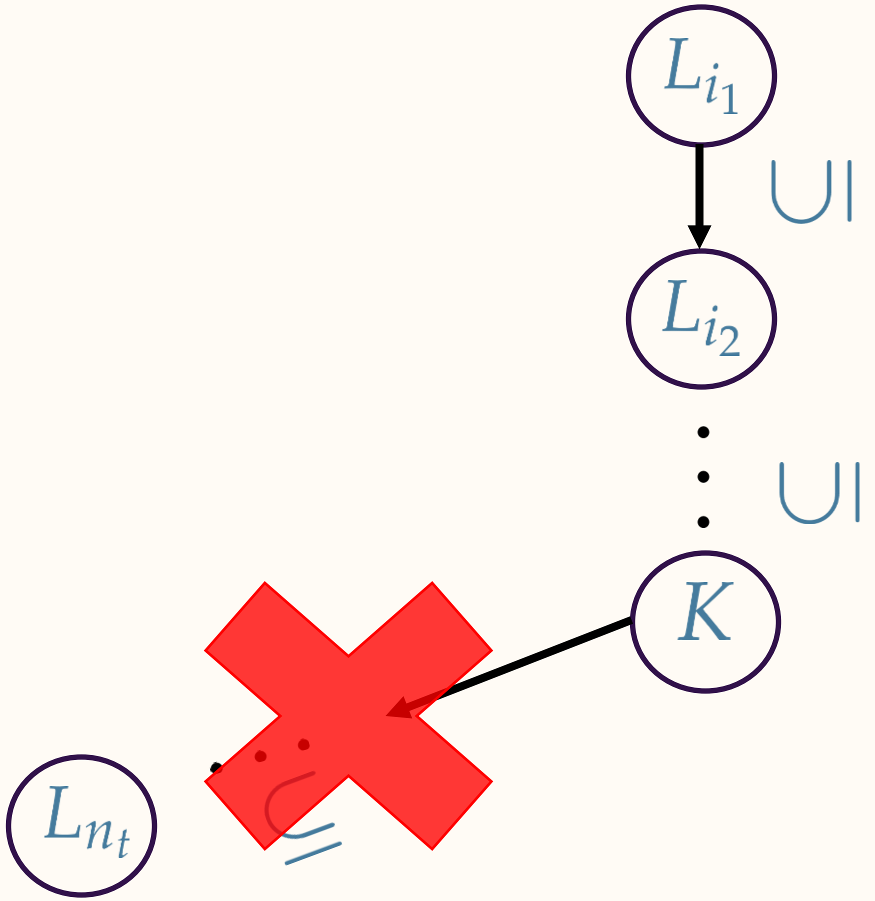
# Chains of Critical Languages



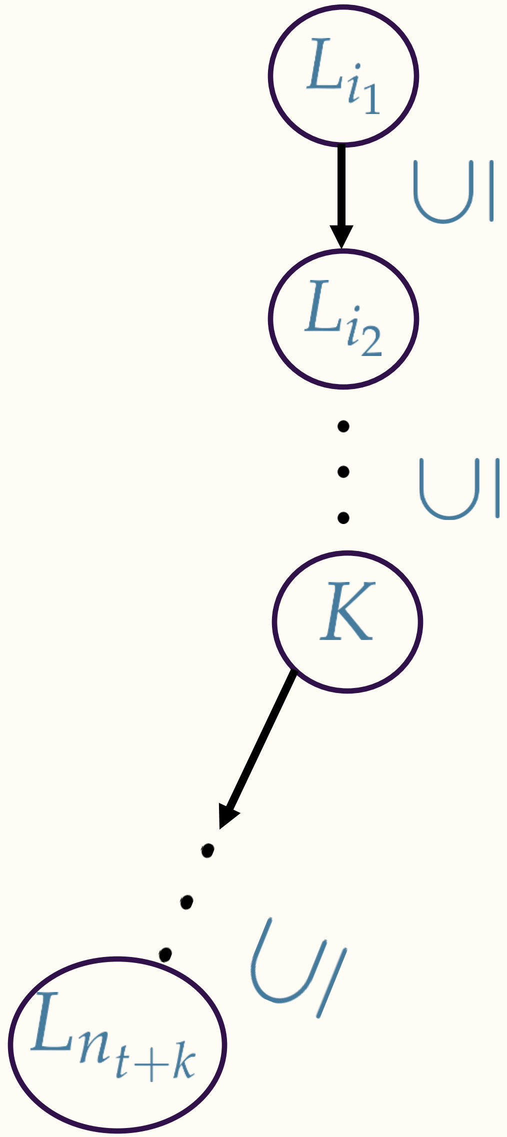
# Chains of Critical Languages



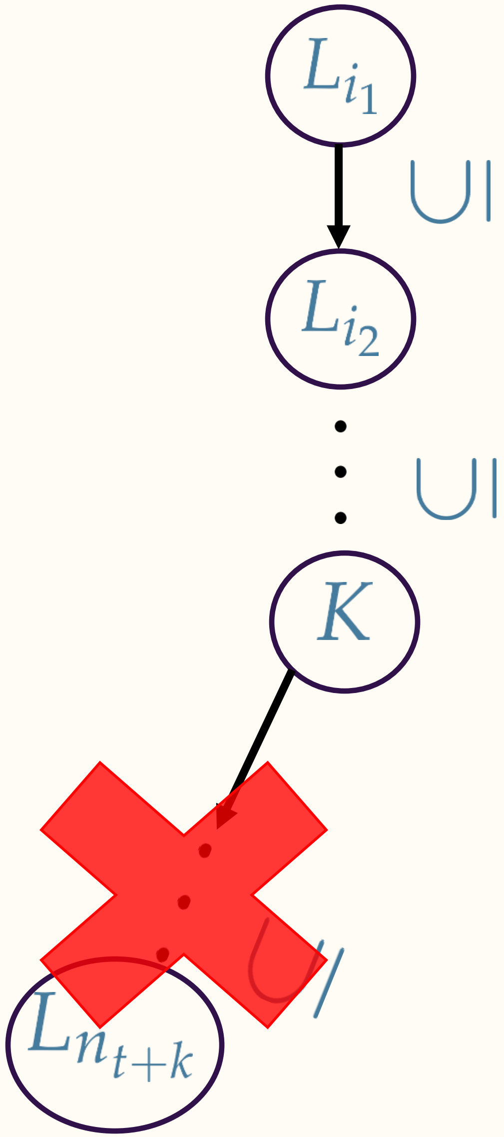
# Chains of Critical Languages



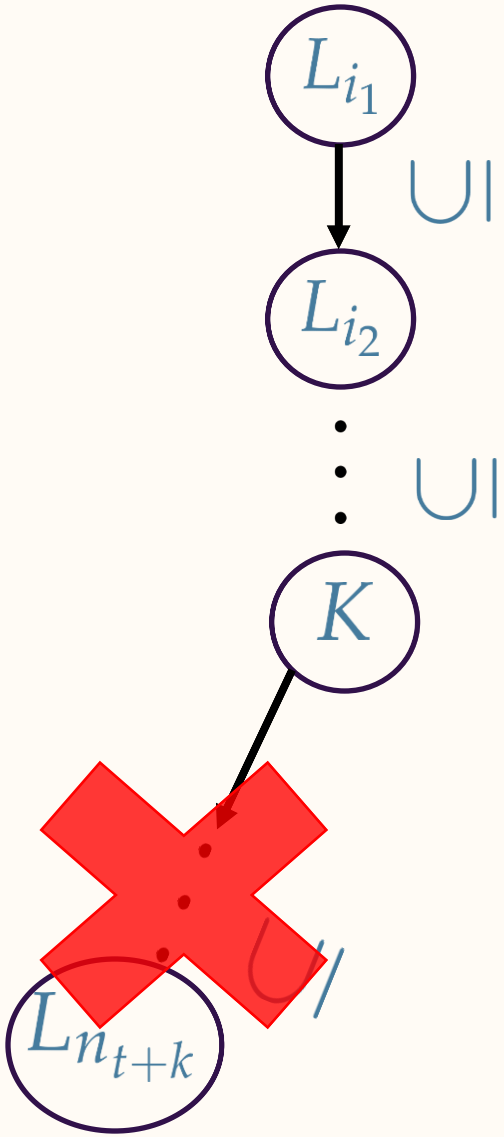
# Chains of Critical Languages



# Chains of Critical Languages



# Chains of Critical Languages



# Algorithm of [Kleinberg and Wei, 2025]

# Algorithm of [Kleinberg and Wei, 2025]

After large enough $t$, the prefix of the chain of critical languages up to $K$ stops changing

# Algorithm of [Kleinberg and Wei, 2025]

After large enough $t$, the prefix of the chain of critical languages up to $K$ stops changing

After large enough $t$, if a critical language gets contradicted, then we know it appears after $K$, hence in the next round we can "backtrack" to the "previous" critical language!

# Algorithm of [Kleinberg and Wei, 2025]

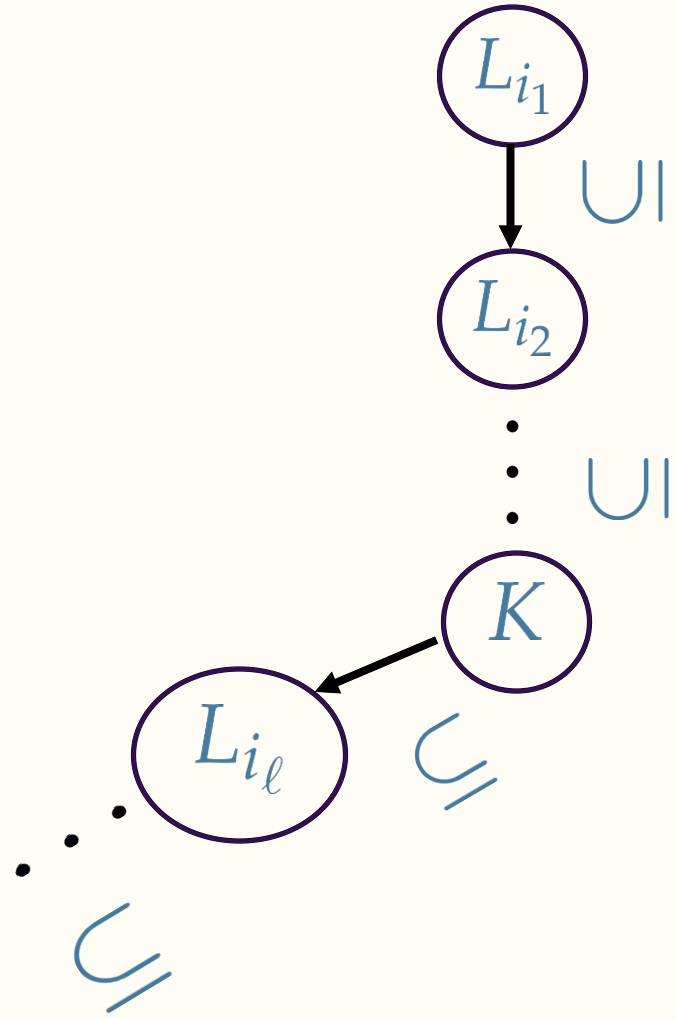
At step t, only consider $L_1, \ldots, L_t$

$L_{n_t}$ : critical language with highest index $n_t \leq t$

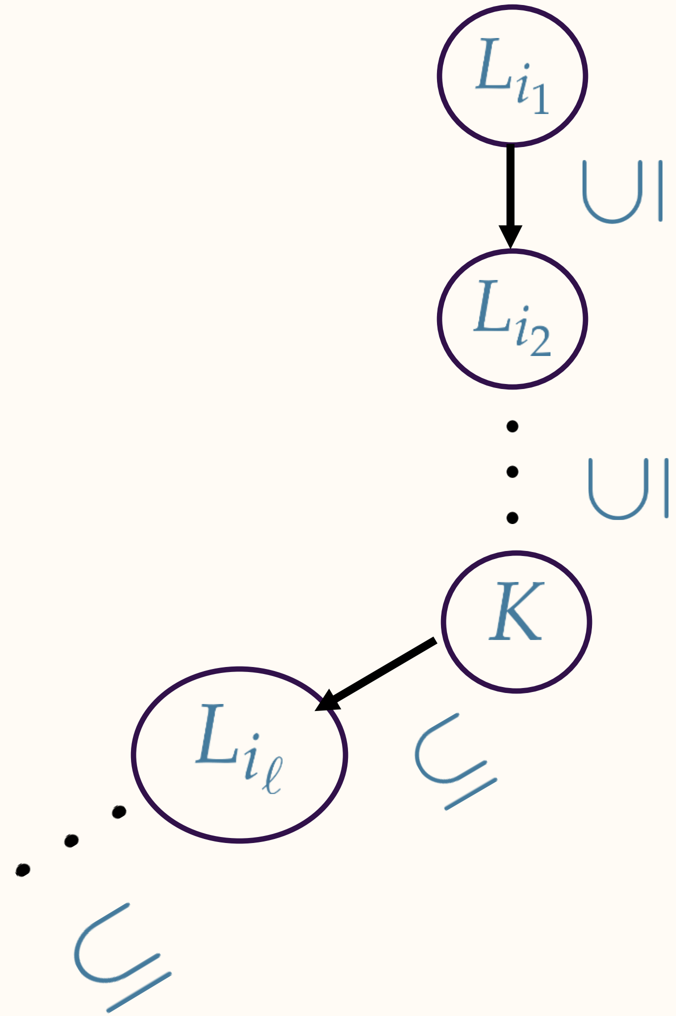
If all critical languages from $t-1$ remain critical, output $n_t$

Otherwise output the index of the critical language that appears before the first contradicted language

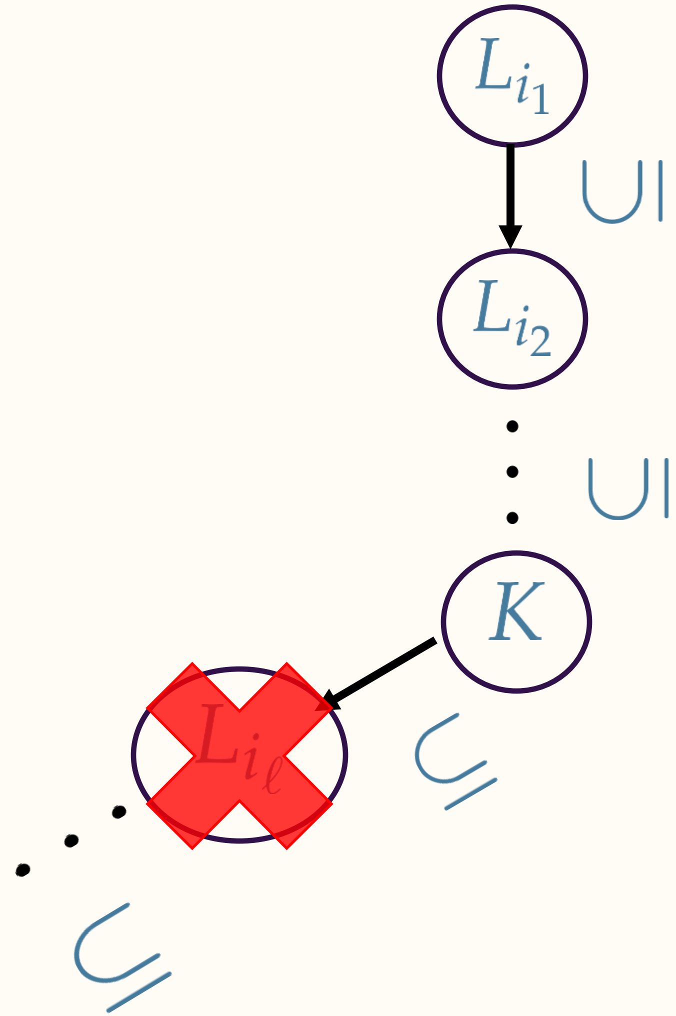
# Chains of Critical Languages



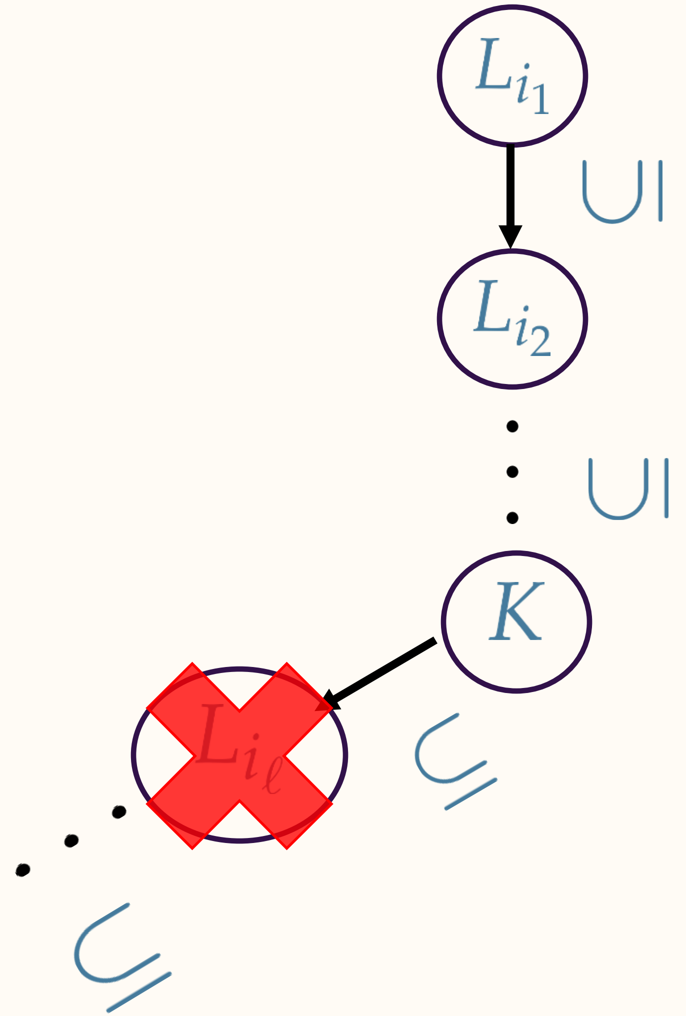
# Chains of Critical Languages



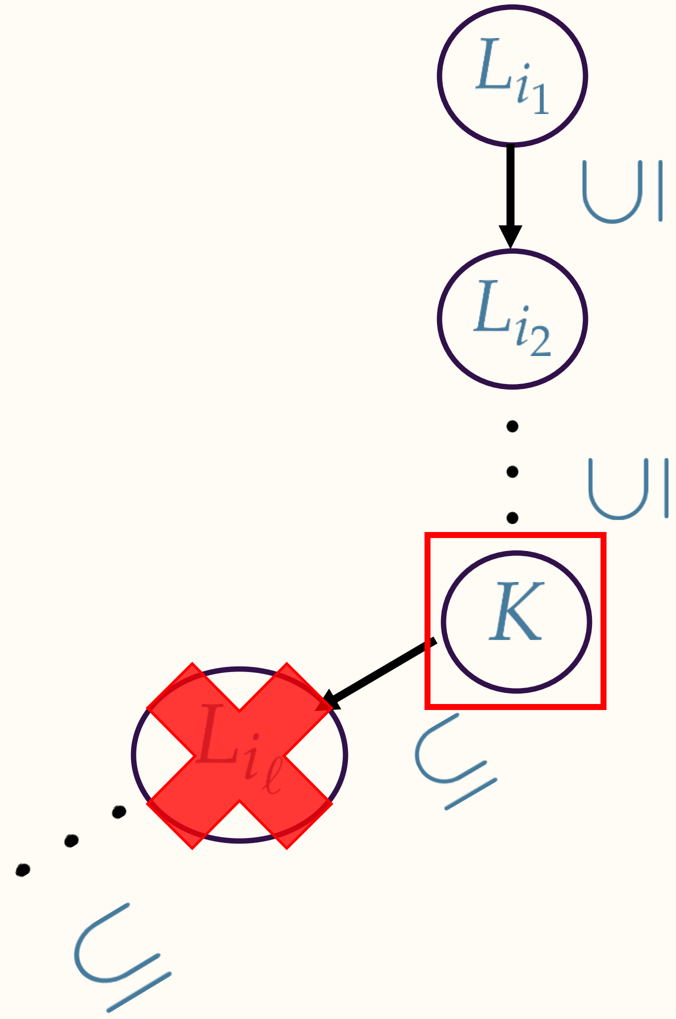
# Chains of Critical Languages



# Chains of Critical Languages



# Chains of Critical Languages



# References


[CP'25] Exploring Facets of Language Generation in the Limit, COLT'25
[HKMV'25] On Union-Closedness of Language Generation, arXiv'25
[KM'24] Language Generation in the Limit, NeurIPS'24
[KW'25] Density Measures for Language Generation, arXiv'25
[KMV'25] On the Limits of Language Generation: … STOC'25
[KMV'24] Characterizations of Language Generation With Breadth, arXiv'24
[PRR'25] Representative Language Generation, ICML'25


*Thank you!*