Dynamic Membership for Regular Tree Languages

Antoine Amarilli[1]
Joint work with Corentin Barloy, Pawel Gawrychowski, Louis Jachiet, Charles Paperman

September 18, 2024

[1]Inria Lille

Dynamic membership of words to regular languages

- Fix a regular language L

$$L = (ab)^*$$

Dynamic membership of words to regular languages

- Fix a regular language L $\quad L = (ab)^*$
- Given as input a word w, write $n := |w|$ $\quad w = ababab$

Dynamic membership of words to regular languages

- Fix a regular language L $\qquad L = (ab)^*$
- Given as input a word w, write $n := |w|$ $\qquad w = ababab$
- Can decide in $O(n)$ whether $w \in L$

Dynamic membership of words to regular languages

- Fix a regular language L — $L = (ab)^*$
- Given as input a word w, write $n := |w|$ — $w = ababab$
- Can decide in $O(n)$ whether $w \in L$

Our problem: substitute some letters while maintaining membership to L!

Dynamic membership of words to regular languages

- Fix a regular language L — $L = (ab)^*$
- Given as input a word w, write $n := |w|$ — $w = ababab$
- Can decide in $O(n)$ whether $w \in L$

Our problem: substitute some letters while maintaining membership to L!

$$w = ababab \qquad w \in L$$

Dynamic membership of words to regular languages

- Fix a **regular language** L $\qquad L = (ab)^*$
- Given as input a **word** w, write $n := |w|$ $\qquad w = ababab$
- Can decide in $O(n)$ whether $w \in L$

Our problem: **substitute** some letters while **maintaining membership** to L!

$$w = abbbab \qquad w \notin L$$

Dynamic membership of words to regular languages

- Fix a regular language L $\qquad L = (ab)^*$
- Given as input a word w, write $n := |w|$ $\qquad w = ababab$
- Can decide in $O(n)$ whether $w \in L$

Our problem: substitute some letters while maintaining membership to L!

$$w = abbbaa \qquad w \notin L$$

Dynamic membership of words to regular languages

- Fix a regular language L $\qquad L = (ab)^*$
- Given as input a word w, write $n := |w|$ $\qquad w = ababab$
- Can decide in $O(n)$ whether $w \in L$

Our problem: substitute some letters while maintaining membership to L!

$$w = abbbaa \qquad w \notin L$$

What is the complexity of this problem? can we do better than $O(n)$ per update?

Dynamic membership for words: Results (actually for monoids)

- Skovbjerg Frandsen, Miltersen, Skyum, “*Dynamic Word Problems*”, JACM’97
- A., Jachiet, Paperman, “*Dynamic Membership for Regular Languages*”, ICALP’21

Dynamic membership for words: Results (actually for monoids)

- Skovbjerg Frandsen, Miltersen, Skyum, "*Dynamic Word Problems*", JACM'97
- A., Jachiet, Paperman, "*Dynamic Membership for Regular Languages*", ICALP'21

in $O(1)$

in $O(\log \log n)$
not in $O(1)$?

in $\Theta(\log n / \log \log n)$

Dynamic membership for words: Results (actually for monoids)

- Skovbjerg Frandsen, Miltersen, Skyum, "*Dynamic Word Problems*", JACM'97
- A., Jachiet, Paperman, "*Dynamic Membership for Regular Languages*", ICALP'21

in $O(1)$

in $O(\log \log n)$
not in $O(1)$?

in $\Theta(\log n / \log \log n)$

- For some regular languages we can maintain membership in **$O(1)$ per update**

Dynamic membership for words: Results (actually for monoids)

- Skovbjerg Frandsen, Miltersen, Skyum, "*Dynamic Word Problems*", JACM'97
- A., Jachiet, Paperman, "*Dynamic Membership for Regular Languages*", ICALP'21

in $O(1)$

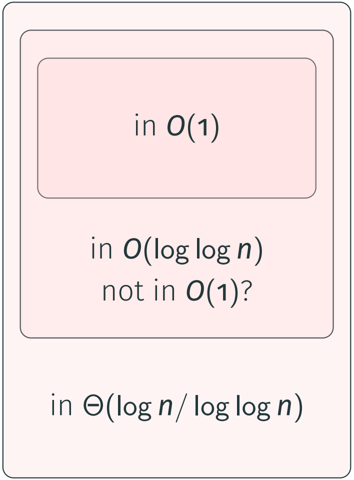
in $O(\log \log n)$
not in $O(1)$?

in $\Theta(\log n / \log \log n)$

- For some regular languages we can maintain membership in $O(1)$ per update
  - finite languages — e.g., $L = ab$

Dynamic membership for words: Results (actually for monoids)

- Skovbjerg Frandsen, Miltersen, Skyum, "*Dynamic Word Problems*", JACM'97
- A., Jachiet, Paperman, "*Dynamic Membership for Regular Languages*", ICALP'21



- For some regular languages we can maintain membership in $O(1)$ **per update**
  - finite languages — e.g., $L = ab$
  - commutative languages — e.g., "even number of c's"

Dynamic membership for words: Results (actually for monoids)

- Skovbjerg Frandsen, Miltersen, Skyum, "*Dynamic Word Problems*", JACM'97
- A., Jachiet, Paperman, "*Dynamic Membership for Regular Languages*", ICALP'21

in $O(1)$

in $O(\log \log n)$
not in $O(1)$?

in $\Theta(\log n / \log \log n)$

- For some regular languages we can maintain membership in **$O(1)$ per update**
  - **finite languages** e.g., $L = ab$
  - **commutative languages** e.g., "even number of c's"
  - "**combinations**" of the two
    e.g., "one a before one b, even number of c's

Dynamic membership for words: Results (actually for monoids)

- Skovbjerg Frandsen, Miltersen, Skyum, "*Dynamic Word Problems*", JACM'97
- A., Jachiet, Paperman, "*Dynamic Membership for Regular Languages*", ICALP'21

in $O(1)$

in $O(\log \log n)$
not in $O(1)$?

in $\Theta(\log n / \log \log n)$

- For some regular languages we can maintain membership in $O(1)$ **per update**
  - finite languages — e.g., $L = ab$
  - commutative languages — e.g., "even number of c's"
  - "combinations" of the two
    e.g., "one a before one b, even number of c's
- For some other languages, $O(\log \log n)$ algorithm

Dynamic membership for words: Results (actually for monoids)

- Skovbjerg Frandsen, Miltersen, Skyum, "*Dynamic Word Problems*", JACM'97
- A., Jachiet, Paperman, "*Dynamic Membership for Regular Languages*", ICALP'21

in $O(1)$

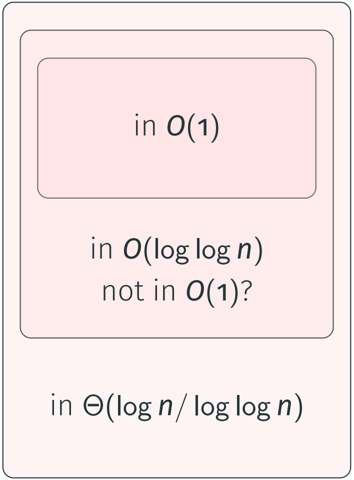
in $O(\log\log n)$
not in $O(1)$?

in $\Theta(\log n / \log\log n)$

- For some regular languages we can maintain membership in **$O(1)$ per update**
  - **finite languages** e.g., $L = ab$
  - **commutative languages** e.g., "even number of c's"
  - "**combinations**" of the two
    e.g., "one a before one b, even number of c's
- For some other languages, $O(\log\log n)$ algorithm
  - all **aperiodic languages** e.g., $L = (ab)^*$

Dynamic membership for words: Results (actually for monoids)

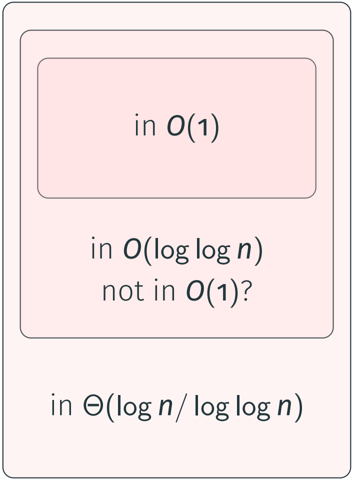
- Skovbjerg Frandsen, Miltersen, Skyum, “*Dynamic Word Problems*”, JACM’97
- A., Jachiet, Paperman, “*Dynamic Membership for Regular Languages*”, ICALP’21



- For some regular languages we can maintain membership in **$O(1)$ per update**
  - **finite languages** e.g., $L = ab$
  - **commutative languages** e.g., “even number of c’s”
  - “**combinations**” of the two
    e.g., “one a before one b, even number of c’s
- For some other languages, **$O(\log\log n)$** algorithm
  - all **aperiodic languages** e.g., $L = (ab)^*$
  - “**combinations**” with commutative languages
    e.g., $L = (ab)^*$ shuffled with an even number of c’s

Dynamic membership for words: Results (actually for monoids)

- Skovbjerg Frandsen, Miltersen, Skyum, “*Dynamic Word Problems*”, JACM’97
- A., Jachiet, Paperman, “*Dynamic Membership for Regular Languages*”, ICALP’21



- For some regular languages we can maintain membership in **$O(1)$ per update**
  - **finite languages** e.g., $L = ab$
  - **commutative languages** e.g., “even number of c’s”
  - “**combinations**” of the two
    e.g., “one a before one b, even number of c’s
- For some other languages, $O(\log \log n)$ algorithm
  - all **aperiodic languages** e.g., $L = (ab)^*$
  - “**combinations**” with commutative languages
    e.g., $L = (ab)^*$ shuffled with an even number of c’s
- All other regular languages: $\Theta(\log n / \log \log n)$

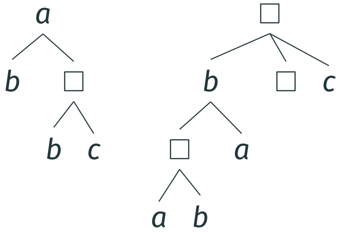
Dynamic membership to tree languages

Innocent question: “What about regular tree languages”?

Dynamic membership to tree languages

Innocent question: “What about regular tree languages”?

- Rooted, ordered, unranked forests

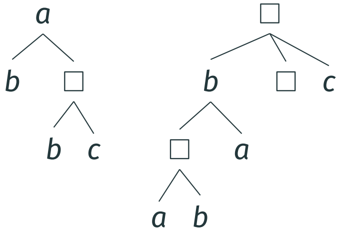


Dynamic membership to tree languages

Innocent question: **"What about regular tree languages"?**

- Rooted, ordered, **unranked forests**
- Label on **nodes** from finite alphabet

e.g., $\Sigma = \{a, b, c, d, \square\}$



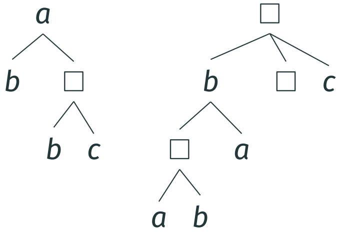
Dynamic membership to tree languages

Innocent question: "What about regular tree languages"?

- Rooted, ordered, unranked forests
- Label on nodes from finite alphabet

  e.g., $\Sigma = \{a, b, c, d, \square\}$
- Regular languages (given, e.g,. as tree automata)

  e.g., "there is an a whose parent is a b"



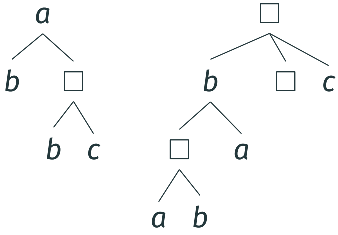
Dynamic membership to tree languages

Innocent question: "What about regular tree languages"?

- Rooted, ordered, unranked forests
- Label on nodes from finite alphabet

  e.g., $\Sigma = \{a, b, c, d, \square\}$
- Regular languages (given, e.g,. as tree automata)

  e.g., "there is an a whose parent is a b"



Some big simplifying assumptions:

- The shape of the tree never changes: we only substitute node labels

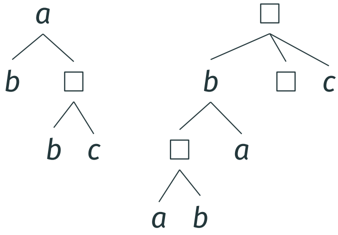
Dynamic membership to tree languages

Innocent question: **"What about regular tree languages"?**

- Rooted, ordered, **unranked forests**
- Label on **nodes** from finite alphabet

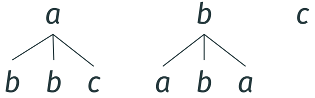
  e.g., $\Sigma = \{a, b, c, d, \square\}$
- Regular languages (given, e.g,. as **tree automata**)

  e.g., "there is an a whose parent is a b"



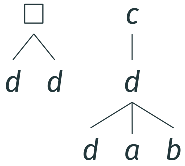
Some big **simplifying assumptions**:

- The **shape** of the tree never changes: we only substitute **node labels**
- **Special neutral label** $\square$ means "replace the node by the list of its children" (avoids "local information")



Fairytale: the $O(1)$ case

We can maintain in $O(1)$:

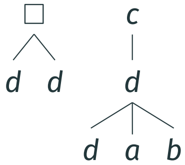


Fairytale: the $O(1)$ case

We can maintain in $\boldsymbol{O(1)}$:

- Commutative regular tree languages

e.g., “even number of $\boldsymbol{d}$’s”



Fairytale: the $O(1)$ case

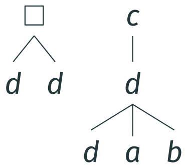
We can maintain in $\boldsymbol{O(1)}$:

- Commutative regular tree languages

  e.g., "even number of $\boldsymbol{d}$'s"

- Finite tree languages on a subalphabet

  e.g., "there is one $\boldsymbol{a}$ and one $\boldsymbol{b}$ and one $\boldsymbol{c}$ which is their LCA"



Fairytale: the $O(1)$ case

We can maintain in $O(1)$:

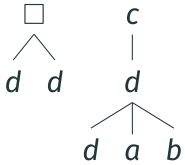
- Commutative regular tree languages

  e.g., "even number of d's"

- Finite tree languages on a subalphabet

  e.g., "there is one a and one b and one c which is their LCA"

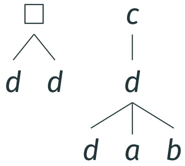
- Boolean combinations thereof



Fairytale: the $O(1)$ case

We can maintain in $O(1)$:

- Commutative regular tree languages
  e.g., "even number of d's"
- Finite tree languages on a subalphabet
  e.g., "there is one a and one b and one c which is their LCA"
- Boolean combinations thereof

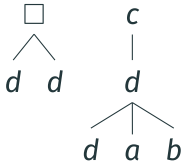


For all other languages, conditional non-$O(1)$ lower bound like for word languages

Fairytale: the $O(1)$ case

We can maintain in $O(1)$:

- Commutative regular tree languages
  e.g., "even number of d's"
- Finite tree languages on a subalphabet
  e.g., "there is one a and one b and one c which is their LCA"
- Boolean combinations thereof



For all other languages, conditional non-$O(1)$ lower bound like for word languages

[See PhD of Corentin Barloy, Section 7.2]

Horror story: the $O(\log \log n)$ case

- After $O(1)$, we wanted to generalize the $O(\log \log n)$ class

Horror story: the $O(\log \log n)$ case

- After $O(1)$, we wanted to generalize the $O(\log \log n)$ class
- On words, all aperiodic languages are $O(\log \log n)$

Horror story: the $O(\log \log n)$ case

- After $O(1)$, we wanted to generalize the $O(\log \log n)$ class
- On words, all aperiodic languages are $O(\log \log n)$
- What about trees?

Horror story: the $O(\log \log n)$ case

- After $O(1)$, we wanted to generalize the $O(\log \log n)$ class
- On words, all aperiodic languages are $O(\log \log n)$
- What about trees?

Existential marked ancestor problem: “there is one b with an a ancestor”

Horror story: the $O(\log \log n)$ case

- After $O(1)$, we wanted to generalize the $O(\log \log n)$ class
- On words, all aperiodic languages are $O(\log \log n)$
- What about trees?

Existential marked ancestor problem: "there is one b with an a ancestor"

This (aperiodic) problem has an unconditional $\Omega(\log n / \log \log n)$ lower bound!
(from Alstrup et al., FOCS'98)

What can we do in $O(\log \log n)$?

We can still do many things in $O(\log \log n)$!

What can we do in $O(\log \log n)$?

We can still do many things in $O(\log \log n)$!

- Whatever can be done in $O(\log \log n)$ on the XML serialization (intuitively, separation by a $O(\log \log n)$ regular word language)

What can we do in $O(\log \log n)$?

We can still do many things in $O(\log \log n)$!

- Whatever can be done in $O(\log \log n)$ on the XML serialization (intuitively, separation by a $O(\log \log n)$ regular word language)
  - The sequence of leaves is in an aperiodic word language
  - The branch of first children is an aperiodic word language

What can we do in $O(\log\log n)$?

We can still do many things in $O(\log\log n)$!

- Whatever can be done in $O(\log\log n)$ on the XML serialization (intuitively, separation by a $O(\log\log n)$ regular word language)
  - The sequence of leaves is in an aperiodic word language
  - The branch of first children is an aperiodic word language
- Maintaining degree information
  - All internal nodes have exactly 42 children

What can we do in $O(\log \log n)$?

We can still do many things in $O(\log \log n)$!

- Whatever can be done in $O(\log \log n)$ on the XML serialization (intuitively, separation by a $O(\log \log n)$ regular word language)
  - The sequence of leaves is in an aperiodic word language
  - The branch of first children is an aperiodic word language
- Maintaining degree information
  - All internal nodes have exactly 42 children
- Incomprehensible things for non-aperiodic languages (details omitted)

What can we do in $O(\log \log n)$?

We can still do many things in $O(\log \log n)$!

- Whatever can be done in $O(\log \log n)$ on the XML serialization (intuitively, separation by a $O(\log \log n)$ regular word language)
  - The sequence of leaves is in an aperiodic word language
  - The branch of first children is an aperiodic word language
- Maintaining degree information
  - All internal nodes have exactly 42 children
- Incomprehensible things for non-aperiodic languages (details omitted)

But we are still missing a characterization of the $O(\log \log n)$ boundary...

Current roadblock

Consider the following language on alphabet $\Sigma = \{a, b, \square\}$:

Current roadblock

Consider the following language on alphabet $\Sigma = \{a, b, \Box\}$:

- There is exactly one b and it is a leaf

Current roadblock

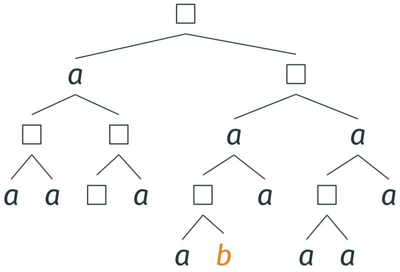
Consider the following language on alphabet $\Sigma = \{a, b, \square\}$:

- There is exactly **one b** and it is **a leaf**
- In the subtree induced by the a's, all internal nodes have **degree 3**

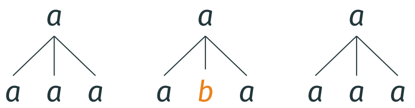
Current roadblock

Consider the following language on alphabet $\Sigma = \{a, b, \square\}$:

- There is exactly **one b** and it is **a leaf**
- In the subtree induced by the a's, all internal nodes have **degree 3**



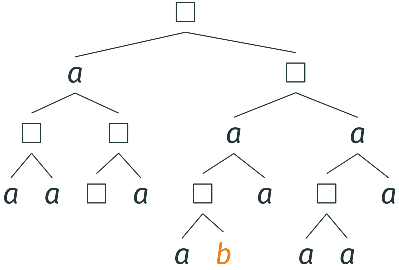
stands for



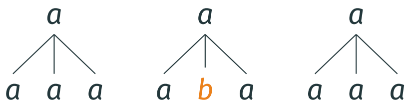
Current roadblock

Consider the following language on alphabet $\Sigma = \{a, b, \square\}$:

- There is exactly **one b** and it is **a leaf**
- In the subtree induced by the a's, all internal nodes have **degree 3**



stands for

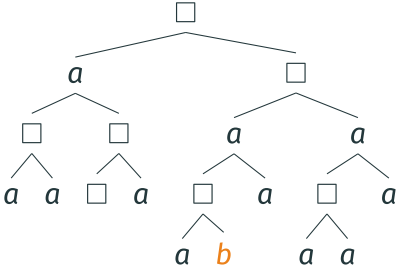


This can be maintained under substitutions in $O(\log \log n)$ per update. But add...

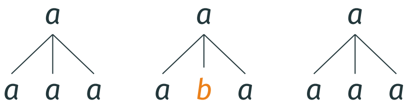
Current roadblock

Consider the following language on alphabet $\Sigma = \{a, b, \square\}$:

- There is exactly **one b** and it is **a leaf**
- In the subtree induced by the a's, all internal nodes have **degree 3**



stands for



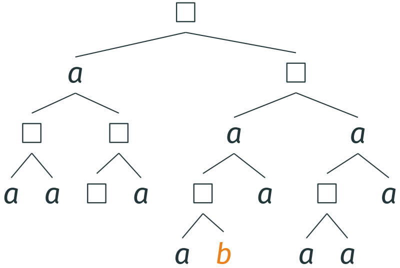
This can be maintained under substitutions in $O(\log \log n)$ per update. But add...

- Going upwards from b we always reach the **second child** of our parent

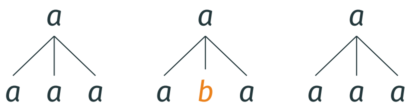
Current roadblock

Consider the following language on alphabet $\Sigma = \{a, b, \square\}$:

- There is exactly **one b** and it is **a leaf**
- In the subtree induced by the a's, all internal nodes have **degree 3**



stands for



This can be maintained under substitutions in $O(\log \log n)$ per update. But add...

- Going upwards from b we always reach the **second child** of our parent

What is the complexity?? no known reduction from existential marked ancestor

References i

Alstrup, S., Husfeldt, T., and Rauhe, T. (1998).
Marked ancestor problems.
In *FOCS*.

Amarilli, A., Jachiet, L., and Paperman, C. (2021).
Dynamic membership for regular languages.
In *ICALP*.

Barloy, C. (2024).
On the complexity of regular languages.
PhD thesis, Université de Lille.

References ii

Skovbjerg Frandsen, G., Miltersen, P. B., and Skyum, S. (1997).
Dynamic word problems.
JACM, 44(2).