Dynamic Membership for Regular Languages

Antoine Amarilli[1], Louis Jachiet[1], Charles Paperman[2]

September 17, 2021

[1]Télécom Paris

[2]Université de Lille

Problem: dynamic membership for regular languages

- Fix a regular language L
  - $\rightarrow$ E.g., $L = (ab)^*$

Problem: dynamic membership for regular languages

- Fix a regular language L
  - $\rightarrow$ E.g., $L = (ab)^*$
- Read an input word w with $n := |w|$
  - $\rightarrow$ E.g., $w = abbbab$

Problem: dynamic membership for regular languages

- Fix a regular language L
  - $\rightarrow$ E.g., $L = (ab)^*$
- Read an input word w with $n := |w|$
  - $\rightarrow$ E.g., $w = abbbab$
- Preprocess it in $O(n)$
  - $\rightarrow$ E.g., we have $w \notin L$

Problem: dynamic membership for regular languages

- Fix a regular language L
  - $\rightarrow$ E.g., $L = (ab)^*$
- Read an input word w with $n := |w|$
  - $\rightarrow$ E.g., $w = abbbab$
- Preprocess it in $O(n)$
  - $\rightarrow$ E.g., we have $w \notin L$
- Maintain the membership of w to L under substitution updates
  - $\rightarrow$ E.g., replace character at position 3 with a: we now have $w \in L$

Problem: dynamic membership for regular languages

- Fix a regular language L
  - $\rightarrow$ E.g., $L = (ab)^*$
- Read an input word w with $n := |w|$
  - $\rightarrow$ E.g., $w = abbbab$
- Preprocess it in $O(n)$
  - $\rightarrow$ E.g., we have $w \notin L$
- Maintain the membership of w to L under substitution updates
  - $\rightarrow$ E.g., replace character at position 3 with a: we now have $w \in L$
- Model: RAM model with $\Theta(\log n)$ cell size and unit-cost arithmetics

What is the complexity of the problem?

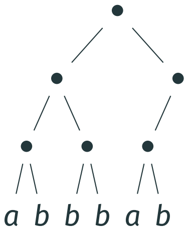
- **Naive algorithm in $O(n)$:** test the whole word again after each update

What is the complexity of the problem?

- **Naive algorithm in $O(n)$:** test the whole word again after each update

- **General-purpose algorithm in $O(\log n)$:**
  - Build a **complete binary tree** on the input word
  - Label the nodes with the **syntactic monoid element** achieved by the subtree rooted at that node
  - When updating a leaf, **recompute** labels upwards

  → Can be improved to $O(\log n / \log \log n)$ via RAM tricks

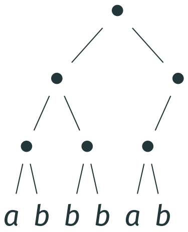


What is the complexity of the problem?

- Naive algorithm in $O(n)$: test the whole word again after each update

- General-purpose algorithm in $O(\log n)$:
  - Build a complete binary tree on the input word
  - Label the nodes with the syntactic monoid element achieved by the subtree rooted at that node
  - When updating a leaf, recompute labels upwards

  → Can be improved to $O(\log n / \log \log n)$ via RAM tricks



- Specific $O(1)$ algorithm for some languages:

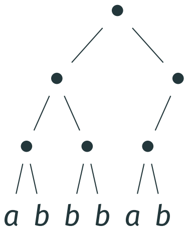
  → E.g., the language a^*: count the number of a's

  → E.g., the language $(ab)^*$ can also be maintained in $O(1)$

What is the complexity of the problem?

- **Naive algorithm in $O(n)$:** test the whole word again after each update
- **General-purpose algorithm in $O(\log n)$:**
  - Build a **complete binary tree** on the input word
  - Label the nodes with the **syntactic monoid element** achieved by the subtree rooted at that node
  - When updating a leaf, **recompute** labels upwards

  → Can be improved to $O(\log n / \log \log n)$ via RAM tricks



- **Specific $O(1)$ algorithm for some languages:**

  → E.g., the language a^*: count the number of a's

  → E.g., the language $(ab)^*$ can also be maintained in $O(1)$

→ **What is the complexity of dynamic membership**, depending on the language?

Problem 2: Dynamic word problem for monoids

- Dynamic word problem for a fixed monoid M: maintain the product of a word of elements of M under substitution updates

Problem 2: Dynamic word problem for monoids

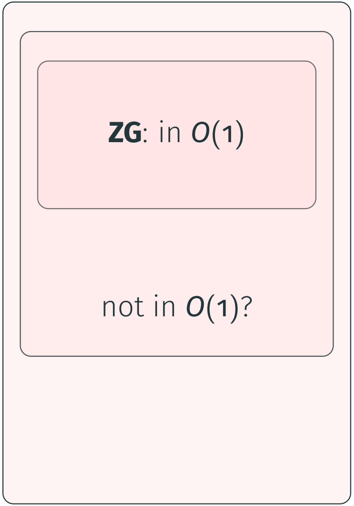
- Dynamic word problem for a fixed monoid M: maintain the product of a word of elements of M under substitution updates
- This is a special case of dynamic membership for regular languages
  - E.g., it assumes that there is a neutral element

Problem 2: Dynamic word problem for monoids

- Dynamic word problem for a fixed monoid M: maintain the product of a word of elements of M under substitution updates
- This is a special case of dynamic membership for regular languages
  - E.g., it assumes that there is a neutral element
- Partial results in [Skovbjerg Frandsen et al., 1997]:

  → in $O(1)$ for commutative monoids
  → in $O(\log \log n)$ for group-free monoids
  → in $\Theta(\log n / \log \log n)$ for a certain monoid class

Results (1/2): dynamic word problem for monoids



- We identify the class **ZG** satisfying $x^{\omega+1}y = yx^{\omega+1}$:
  - For any monoid in **ZG**, the problem is in $O(1)$
  - For any monoid not in **ZG**, we can reduce from a problem that we conjecture is not in $O(1)$

Results (1/2): dynamic word problem for monoids

ZG: in $O(1)$

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

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

- We identify the class **ZG** satisfying $x^{\omega+1}y = yx^{\omega+1}$:
  - For any monoid **in ZG**, the problem is **in $O(1)$**
  - For any monoid **not in ZG**, we can reduce from a problem that we **conjecture is not in $O(1)$**
- We identify the class **SG** satisfying $x^{\omega+1}yx^{\omega} = x^{\omega}yx^{\omega+1}$
  - For any monoid **in SG**, the problem is **in $O(\log\log n)$**
  - For any monoid **not in SG**, it is **in $\Omega(\log n/\log\log n)$** (lower bound of Skovbjerg Frandsen et al.)

Results (2/2): dynamic membership for regular languages

QLZG: in $O(1)$

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

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

- Our results extend to regular language classes called **QLZG** and **QSG**
  - **Q** means: the stable semigroup of the language
  - **L** means: "all submonoids of this semigroup"

Results (2/2): dynamic membership for regular languages

QLZG: in $O(1)$

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

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

- Our results extend to regular language classes called **QLZG** and **QSG**
  - **Q** means: the stable semigroup of the language
  - **L** means: "all submonoids of this semigroup"

→ This yields a conditional trichotomy on the dynamic membership problem

Results (2/2): dynamic membership for regular languages

QLZG: in $O(1)$

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

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

- Our results extend to regular language classes called **QLZG** and **QSG**
  - **Q** means: the stable semigroup of the language
  - **L** means: "all submonoids of this semigroup"

→ This yields a conditional trichotomy on the dynamic membership problem

→ Open problems:
  - Make it unconditional?
  - Identify intermediate cases between **QLZG** and **QSG**?

Results (2/2): dynamic membership for regular languages

QLZG: in $O(1)$

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

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

- Our results extend to regular language classes called **QLZG** and **QSG**
  - **Q** means: the stable semigroup of the language
  - **L** means: "all submonoids of this semigroup"

→ This yields a conditional trichotomy on the dynamic membership problem

→ Open problems:
  - Make it unconditional?
  - Identify intermediate cases between **QLZG** and **QSG**?

Thanks for your attention!

References

Fredman, M. and Saks, M. (1989).
The cell probe complexity of dynamic data structures.
In *STOC*, pages 345–354.

Patrascu, M. (2008).
Lower bound techniques for data structures.
PhD thesis, Massachusetts Institute of Technology.

Skovbjerg Frandsen, G., Miltersen, P. B., and Skyum, S. (1997).
Dynamic word problems.
JACM, 44(2):257–271.