Prediction of Infinite Words with Automata

Tim Smith

LIGM
Université Paris-Est Marne-la-Vallée

EQINOCS workshop, Paris   11 May 2016

Prediction Setting

- We consider an "emitter" and a "predictor".
- The emitter takes no input, but just emits symbols one at a time, continuing indefinitely.
- The predictor receives each symbol output by the emitter, and tries to guess the next symbol.
- We say that the predictor "masters" the emitter if there is a point after which all of the predictor's guesses are correct.

Our Model

- We view the emitter as an infinite word α, i.e., an infinite sequence of symbols drawn from a finite alphabet A.
- We view the predictor as an automaton M whose input is α and whose output is an infinite word $M(\alpha)$. We call each symbol of $M(\alpha)$ a guess.
- M is required to output the i-th symbol of $M(\alpha)$ before it can read the i-th symbol of α.
- If for some $n \geq 1$, for all $i \geq n$, $M(\alpha)[i] = \alpha[i]$, then M masters α.

Prediction Example

- A DFA predictor is a DFA which takes an infinite word as input, and on each transition, tries to guess the next symbol.
- Consider a DFA predictor M which always guesses that the next symbol is a.
- An ultimately periodic word is an infinite word of the form $xy^\omega = xyyy\ldots$ for some x,y in A^*.
- M masters a^ω, ba^ω, aba^ω, bba^ω, ..., i.e., every ultimately periodic word ending in a^ω.

Limitations of DFA predictors

- A purely periodic word is an infinite word of the form $x^\omega = xxx...$ for some x in A^*.
- Theorem: No DFA predictor masters every purely periodic word.
- Proof by contradiction: Suppose there is a DFA predictor M which masters every purely periodic word. Let n be the number of states of M. Then M does not master the purely periodic word $(a^{n+1} b)^\omega$.

Research Direction

- [Smith 2016] Prediction of infinite words with automata CSR 2016 (forthcoming)
- Considers various classes of automata and infinite words in a prediction setting.
- Studies the question of which automata can master which infinite words.
- Motivation: Make connections among automata, infinite words, and learning theory, via the notion of mastery or "learning in the limit" [Gold 1967].

Automata Considered

Class	Name
DFA	deterministic finite automata
DPDA	deterministic pushdown automata
DSA	deterministic stack automata
multi-DFA	multihead deterministic finite automata
sensing multi-DFA	sensing multihead deterministic finite automata

- All of the automata have a one-way input tape.

Infinite Words Considered

Class	Example
purely periodic words	ababab...
ultimately periodic words	abaaaaa...
multilinear words	abaabaaab...

- We have the proper containments:
  - purely periodic $\subset$ ultimately periodic $\subset$ multilinear

Prediction Results

infinite words

automata

∃ masters → ∀	purely periodic	ultimately periodic	multilinear
DFA	×	×	×
DPDA			
DSA			
multi-DFA			
sensing multi-DFA			

Multihead Finite Automata

- Finite automata with one or more input heads on a single tape [Rosenberg 1965].
- We are interested in multi-DFA, the class of one-way multihead deterministic finite automata.

$$\text{multi-DFA} = \bigcup_{k \geq 1} k\text{-DFA}$$

- What are the predictive capabilities of multi-DFA?

Prediction by Multihead Automata

- Theorem: Some multihead DFA masters every ultimately periodic word.
- Construction: Variation of the "tortoise and hare" algorithm. Let M be a two-head DFA which always guesses that the symbols under the heads will match, and
  - if the last guess was correct, M moves each head one square to the right;
  - otherwise, M moves the left head one square to the right and the right head two squares to the right.

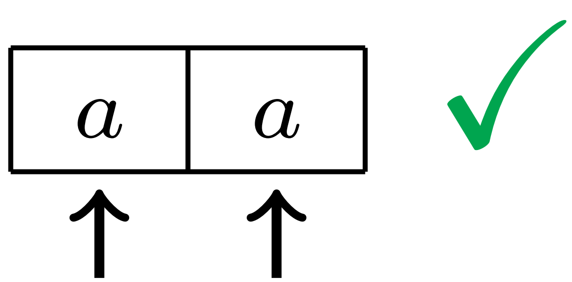
2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$

a	$?$
$\uparrow$	$\uparrow$

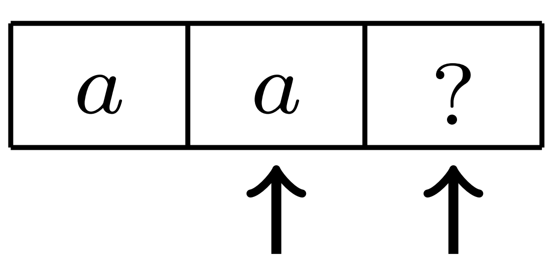
2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$



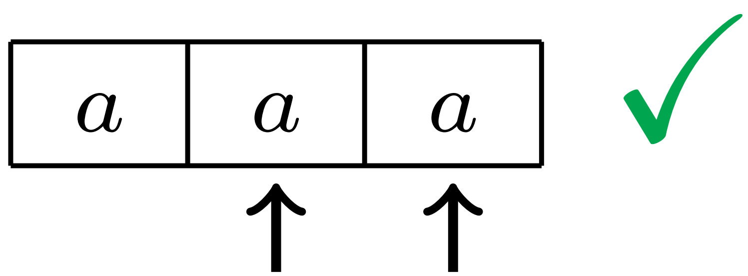
2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$



2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$



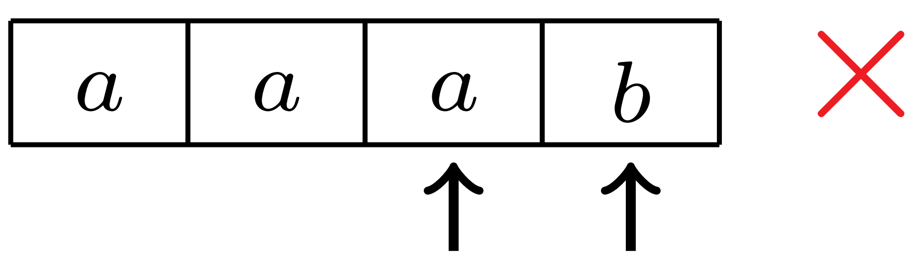
2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$

a	a	a	?
		↑	↑

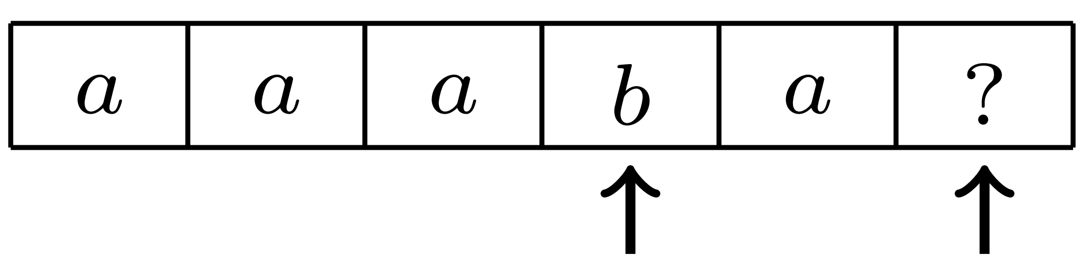
2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$



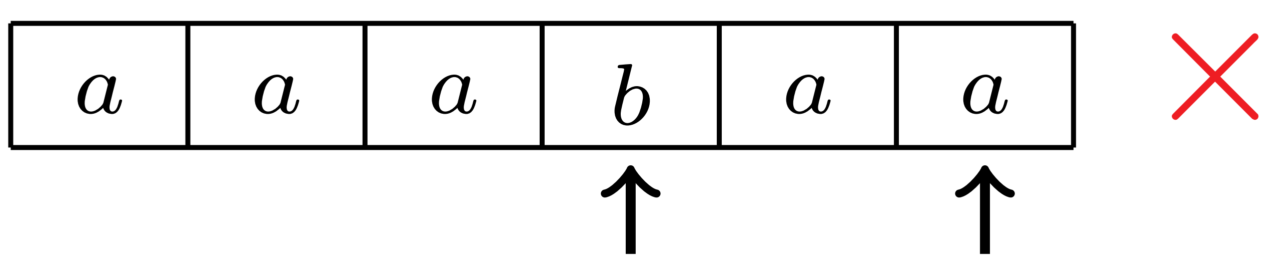
2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$



2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$



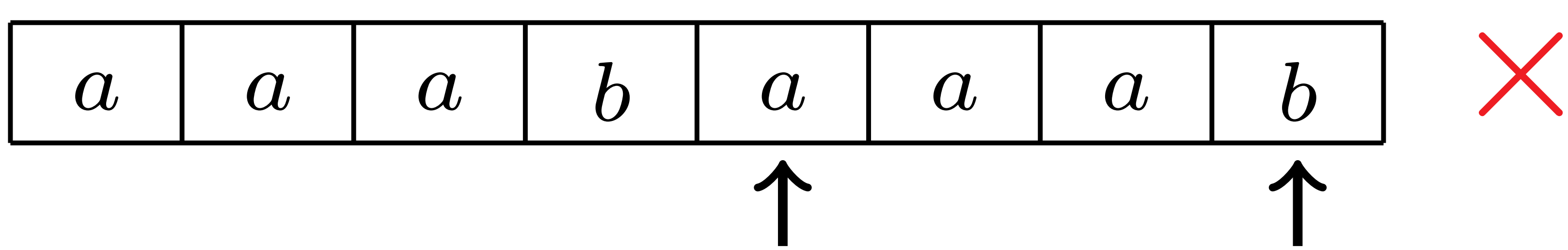
2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$

a	a	a	b	a	a	a	?

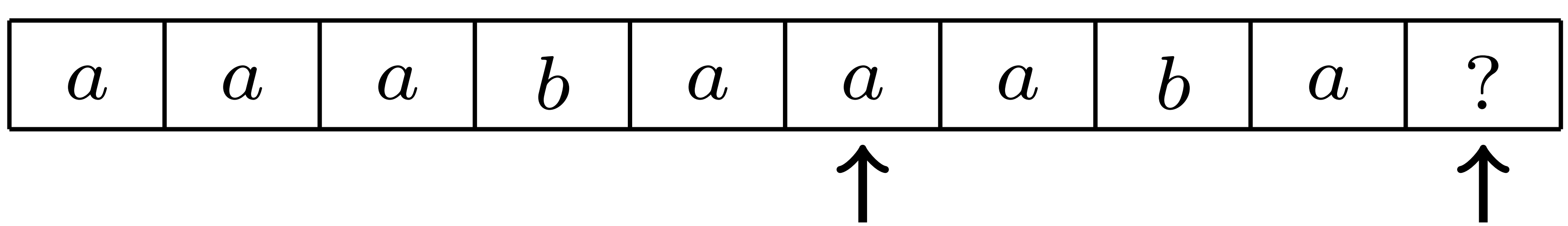
2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$



2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$



2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$

a	a	a	b	a	a	a	b	a	a	✓
					↑				↑	

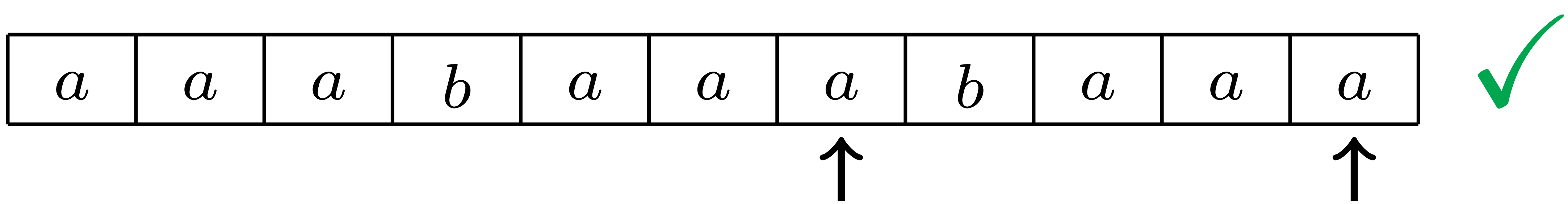
2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$

a	a	a	b	a	a	a	b	a	a	?

2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$



2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^\omega$$

a	a	a	b	a	a	a	b	a	a	a	$?$
							$\uparrow$				$\uparrow$

2-head DFA which masters all ultimately periodic words

$$\alpha = (aaab)^{\omega}$$

a	a	a	b	a	a	a	b	a	a	a	b

Prediction Results

infinite words

∃ masters → ∀	purely periodic	ultimately periodic	multilinear
DFA	✗	✗	✗
DPDA			
DSA			
multi-DFA	✓	✓	
sensing multi-DFA	✓	✓	

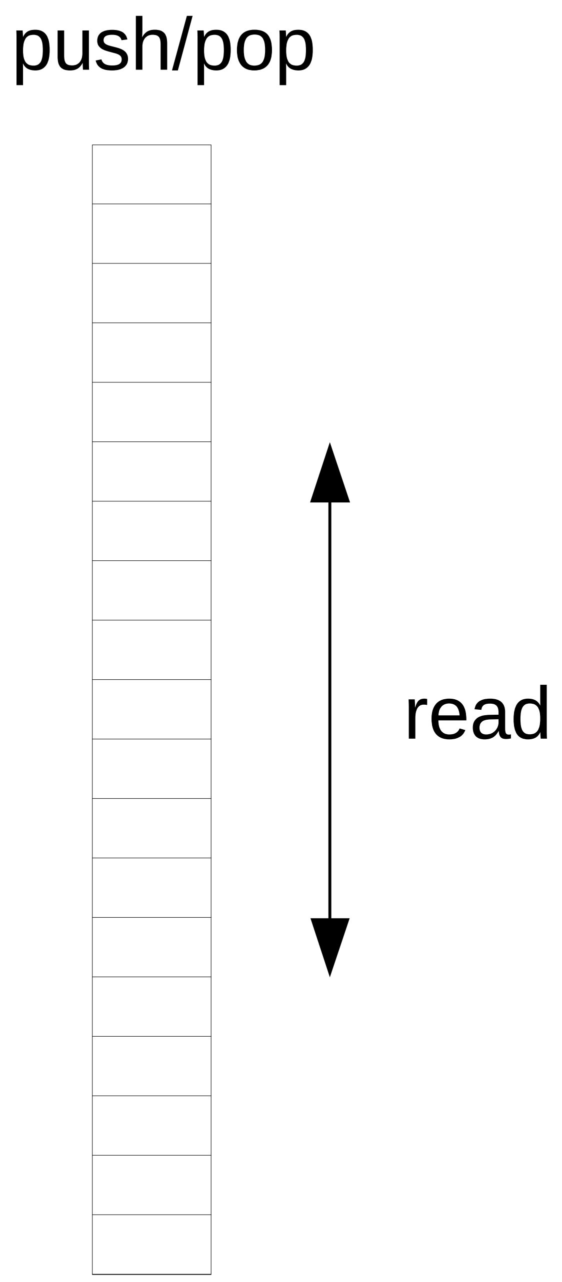
automata

DPDA predictors

- [Smith 2016] No DPDA predictor masters every purely periodic word.
- Proof idea:
  - Suppose there is a DPDA predictor M which masters every purely periodic word. Set n to be very large with respect to the number of states of M and the size of the stack alphabet. Let $\alpha = (a^n b)^\omega$.
  - We show that in some block of consecutive a's, there are configurations C_i and C_j of M with the same state and top-of-stack symbol, such that the stack below the top symbol at C_i is not accessed between C_i and C_j. Then M does not master α.

Stack Automata

- Generalization of pushdown automata due to [Ginsburg, Greibach, & Harrison 1967].
- In addition to pushing and popping at the top of the stack, the stack head can move up and down the stack in read-only mode.
- We consider DSA, the class of one-way deterministic stack automata.

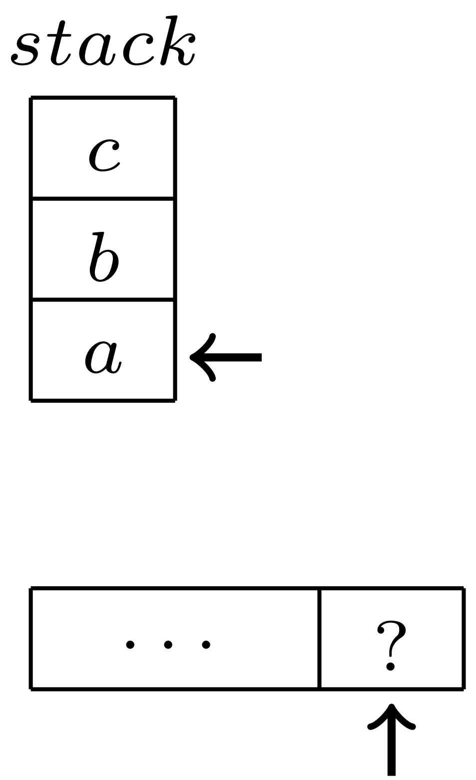


Prediction with Stack Automata

- [Smith 2016] Some DSA predictor masters every purely periodic word.
- Algorithm: The goal is to build up the stack until it holds the period of the word.
    - The stack automaton M makes guesses by repeatedly matching its stack against the input. Call each traversal of the stack a “pass”.
    - In the event of a mismatch, M finishes the current pass, then continues making passes until one succeeds with no mismatches. Then it pushes the next symbol of the input onto the stack and continues as before.
    - Eventually the stack holds the period and M achieves mastery.

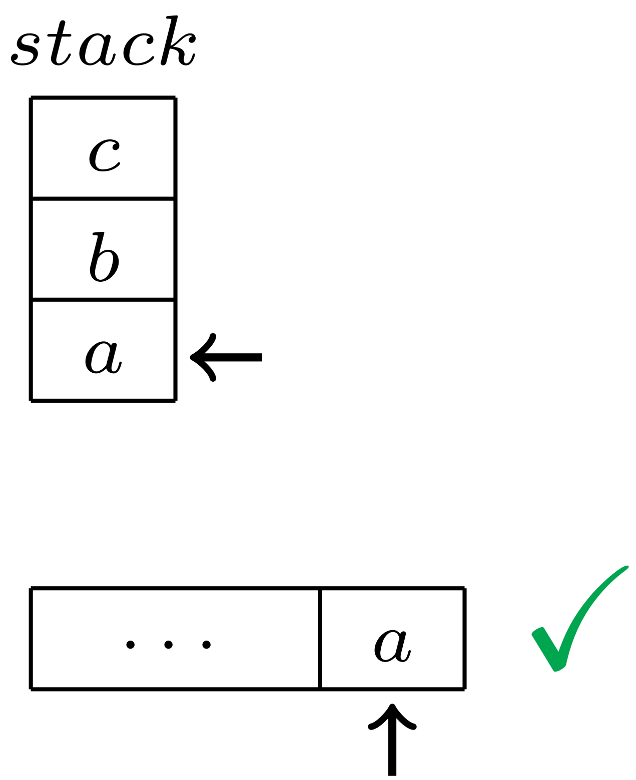
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



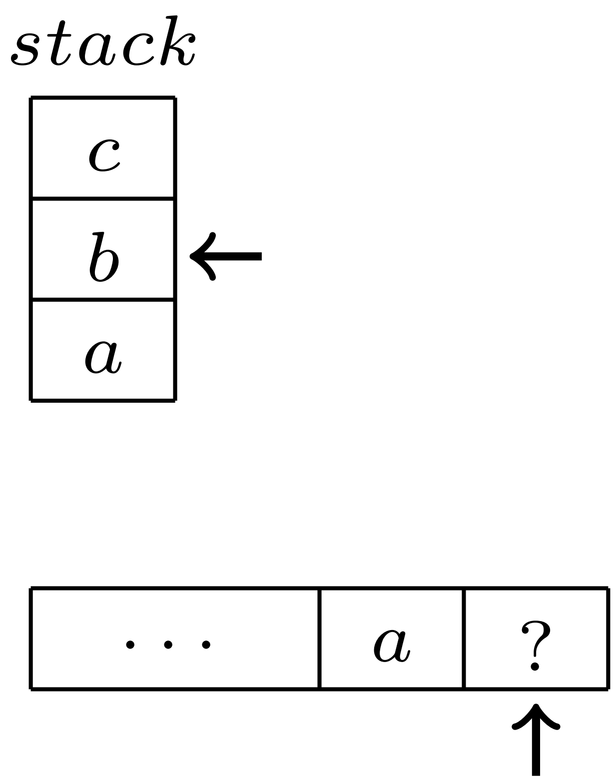
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



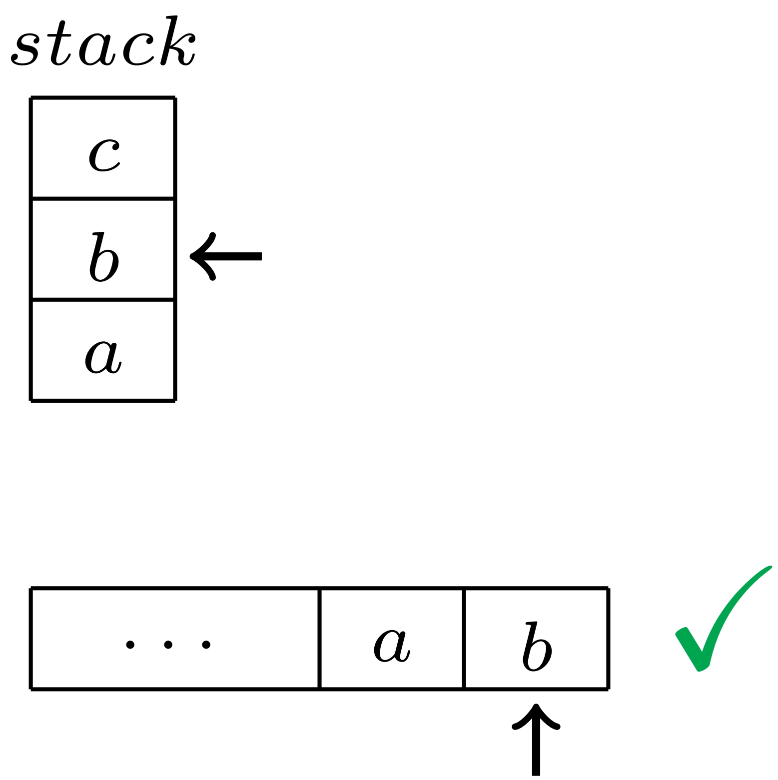
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



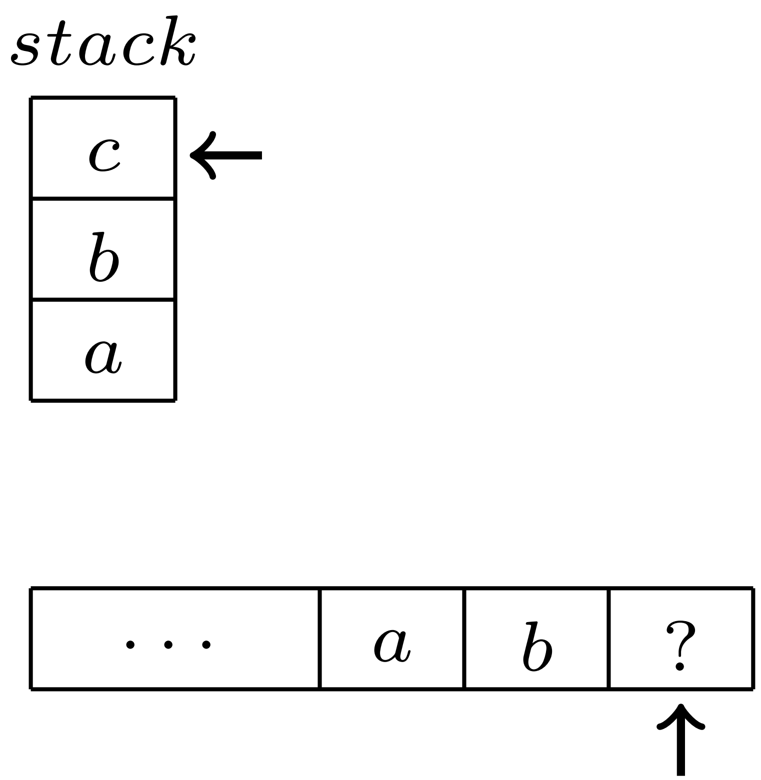
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



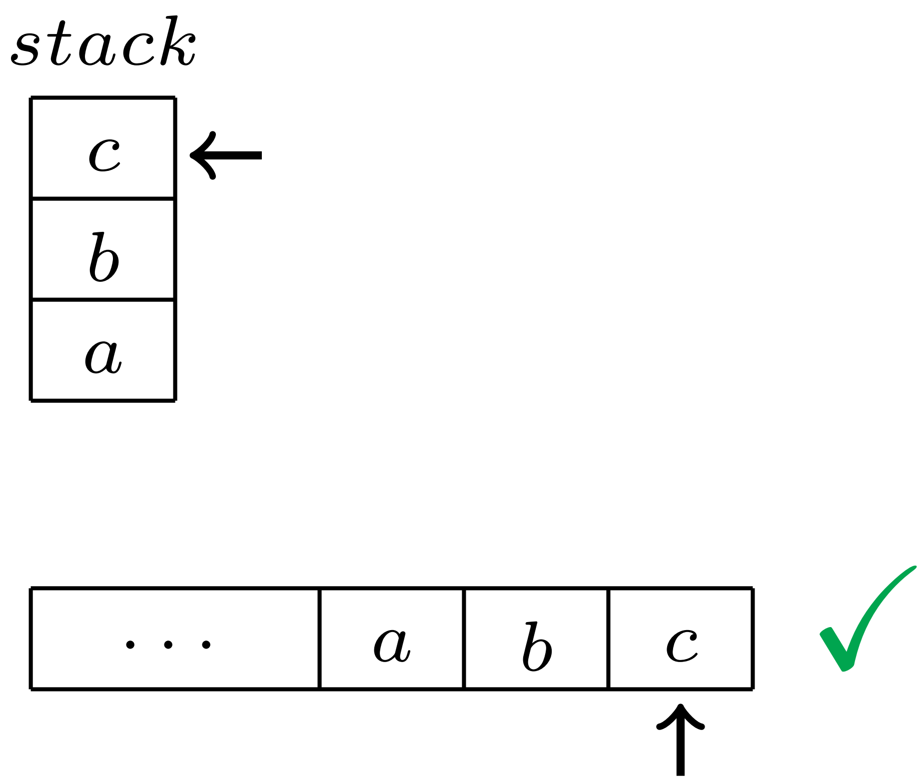
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



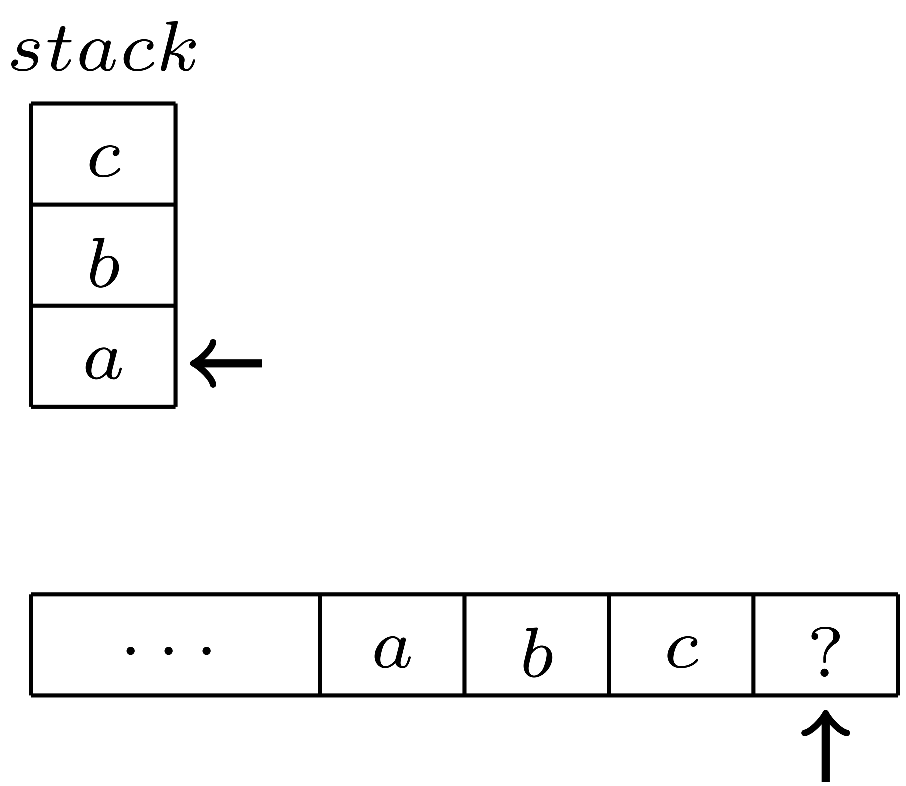
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



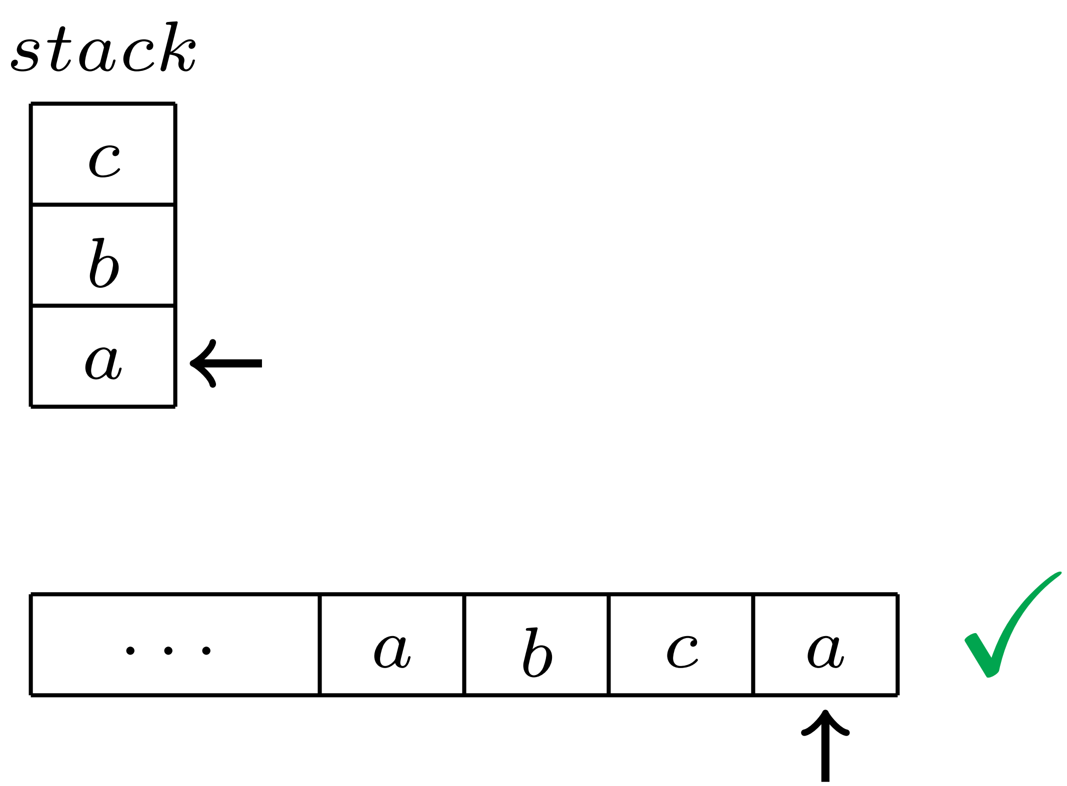
Stack automaton which masters all purely periodic words

$$\alpha = x^{\omega}$$



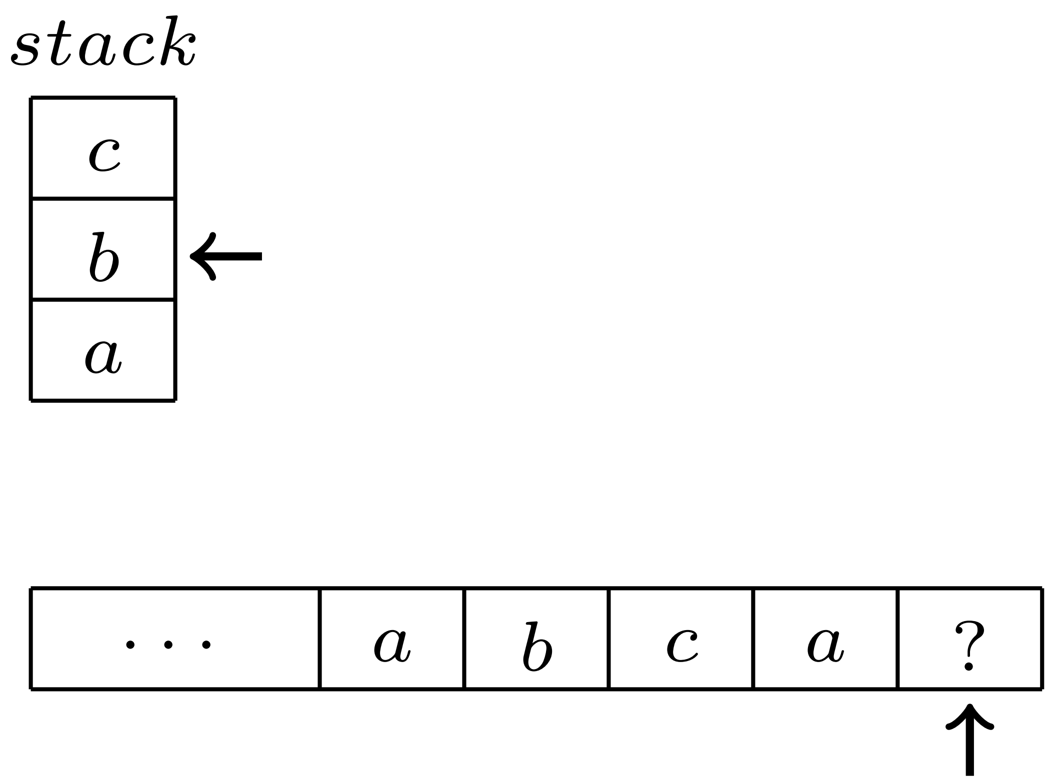
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



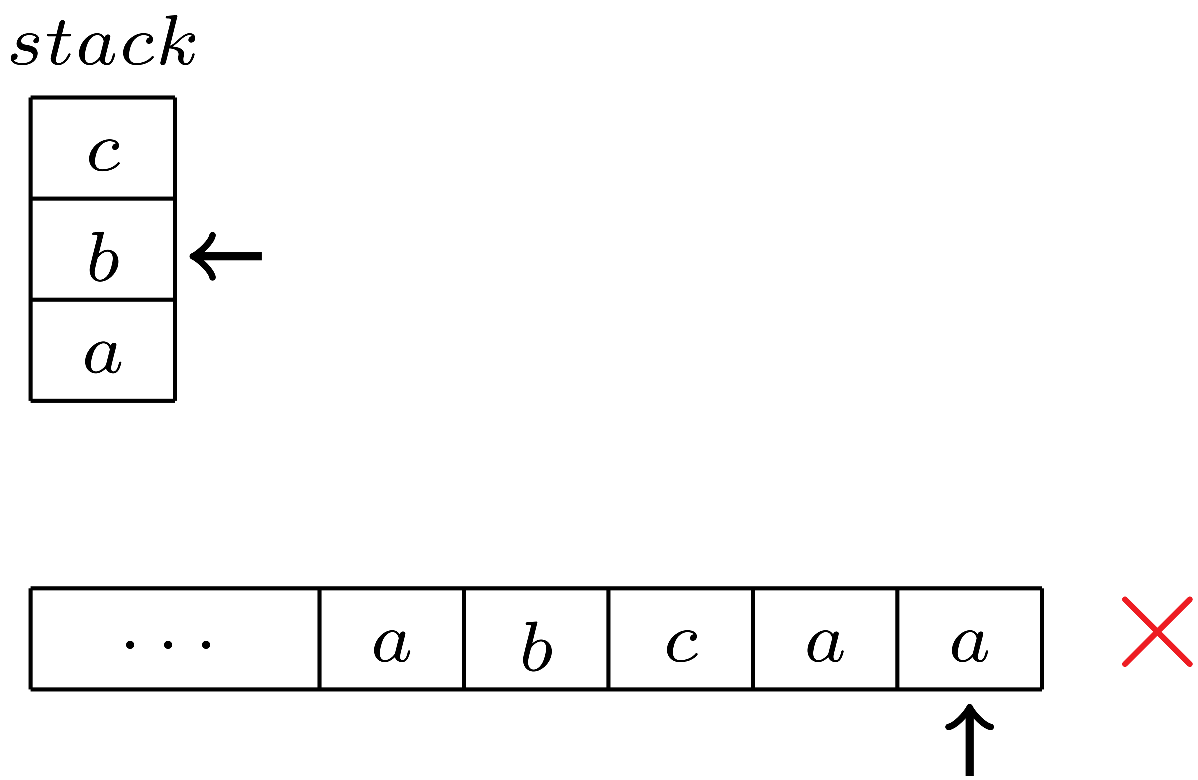
Stack automaton which masters all purely periodic words

$$\alpha = x^{\omega}$$



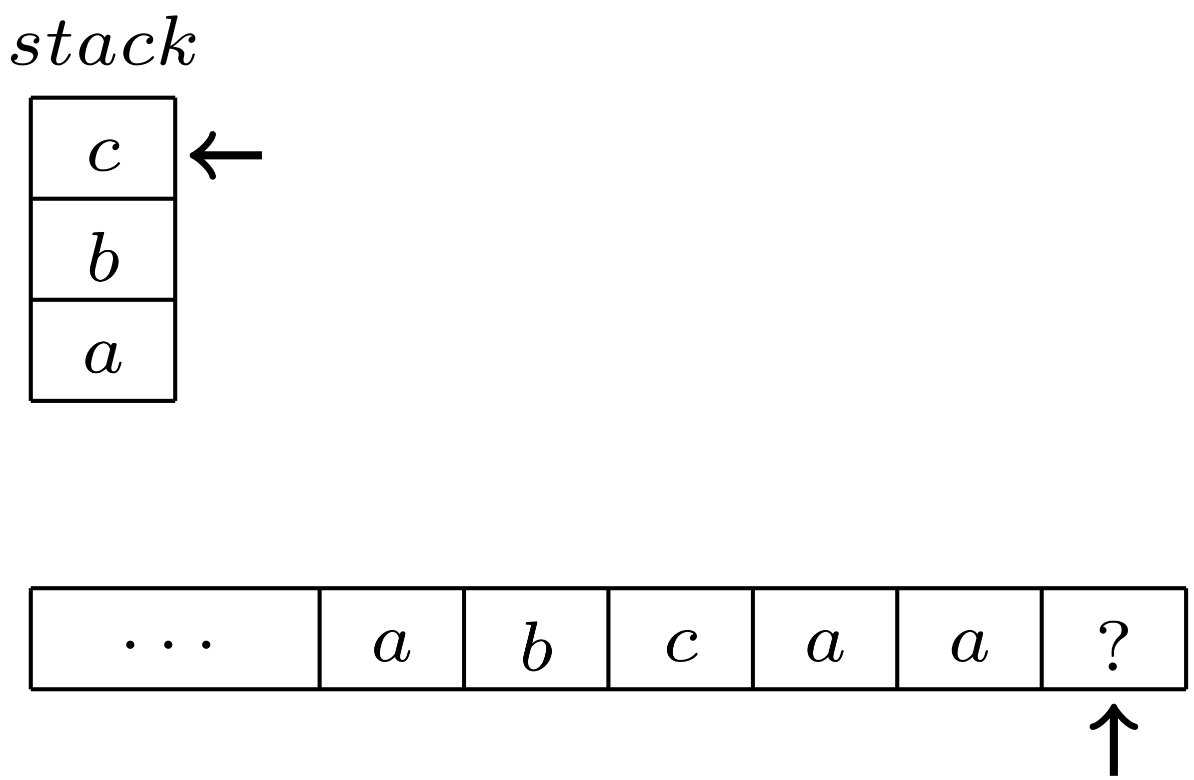
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



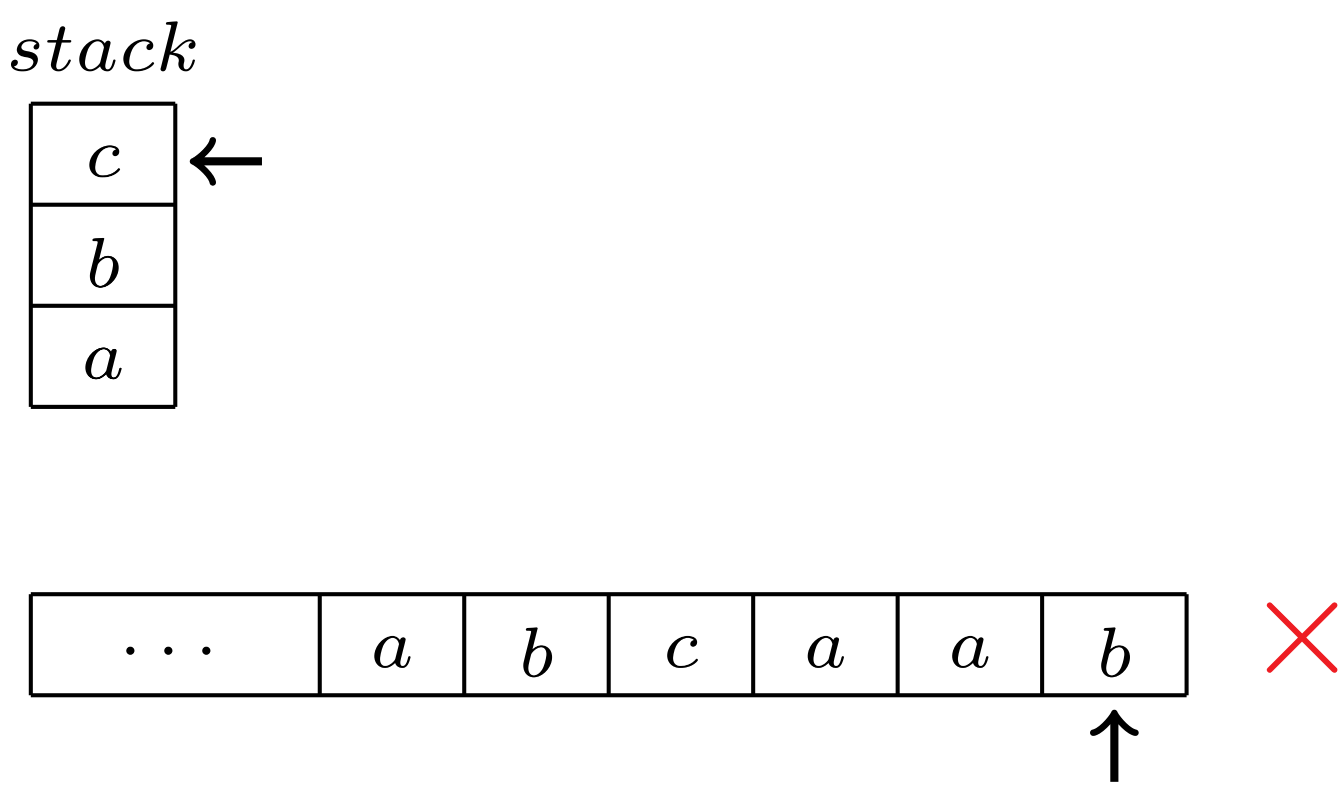
Stack automaton which masters all purely periodic words

$$\alpha = x^{\omega}$$



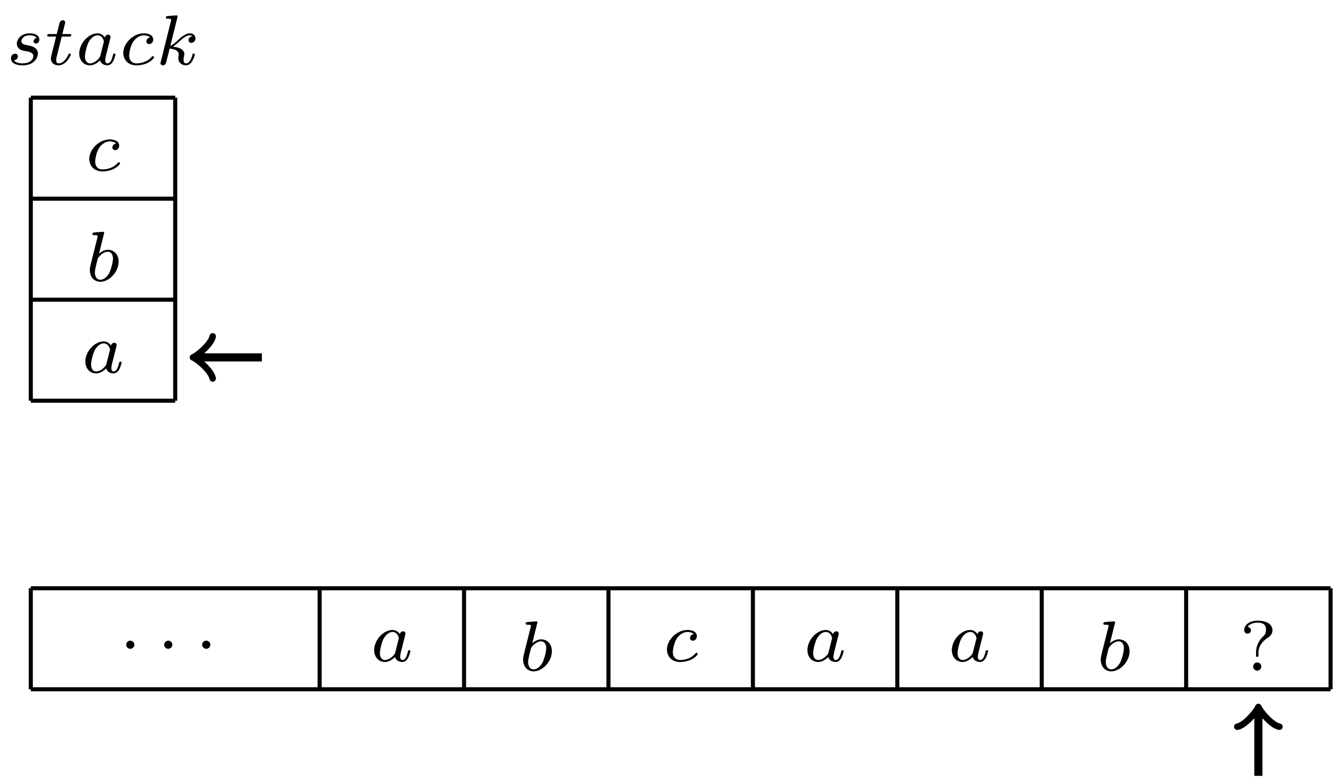
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



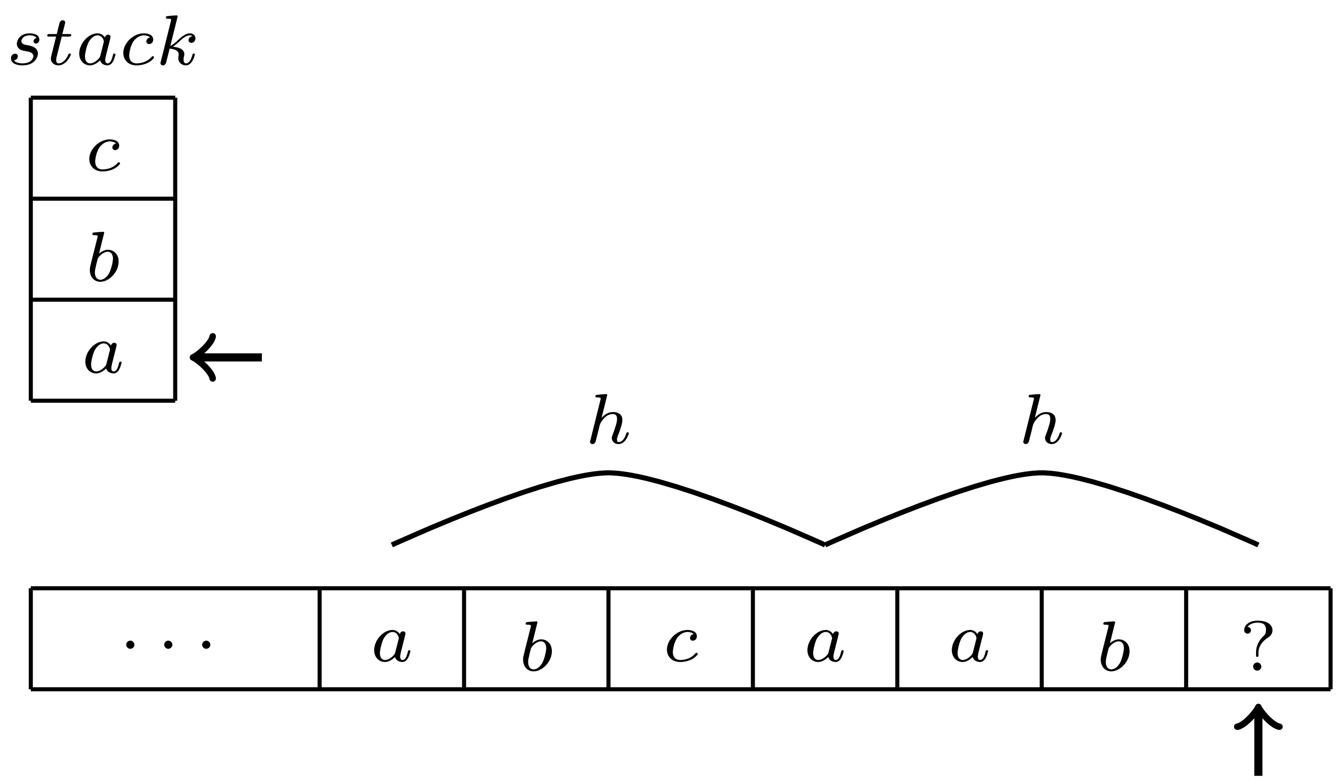
Stack automaton which masters all purely periodic words

$$\alpha = x^{\omega}$$



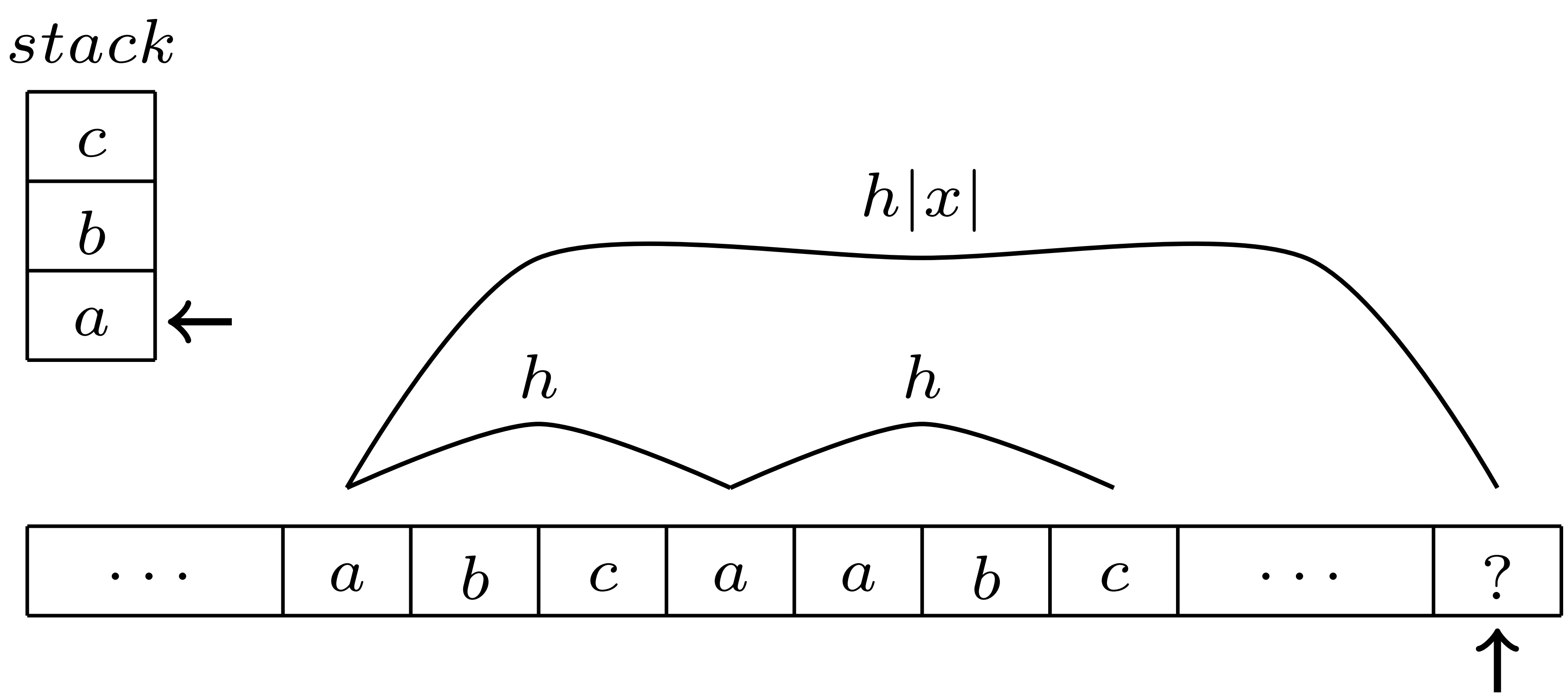
Stack automaton which masters all purely periodic words

$$\alpha = x^{\omega}$$



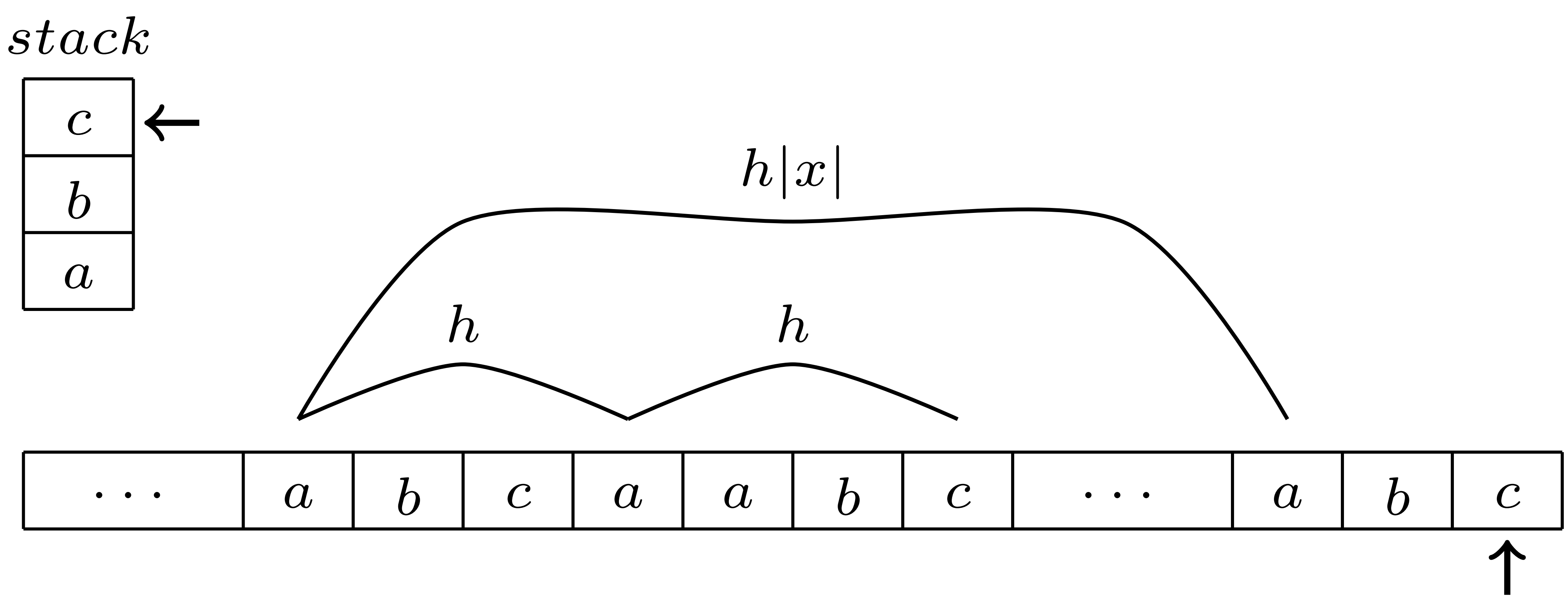
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



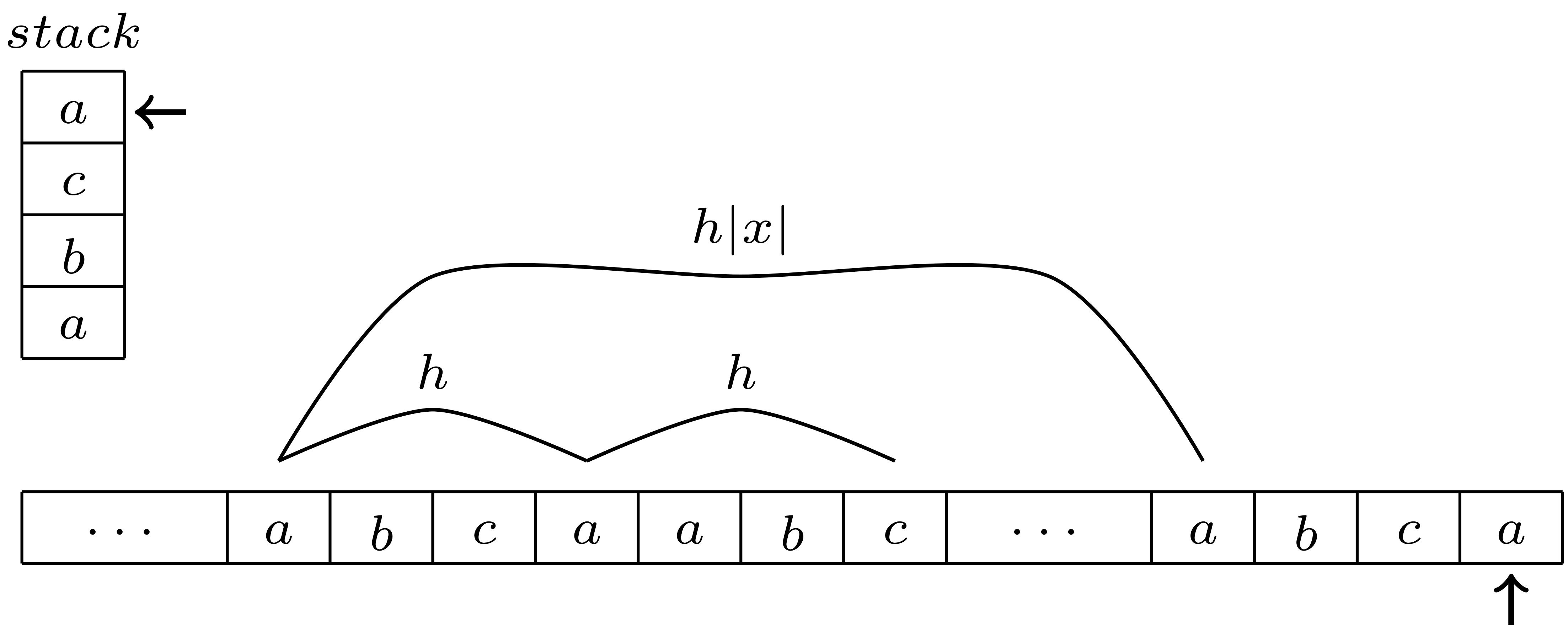
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



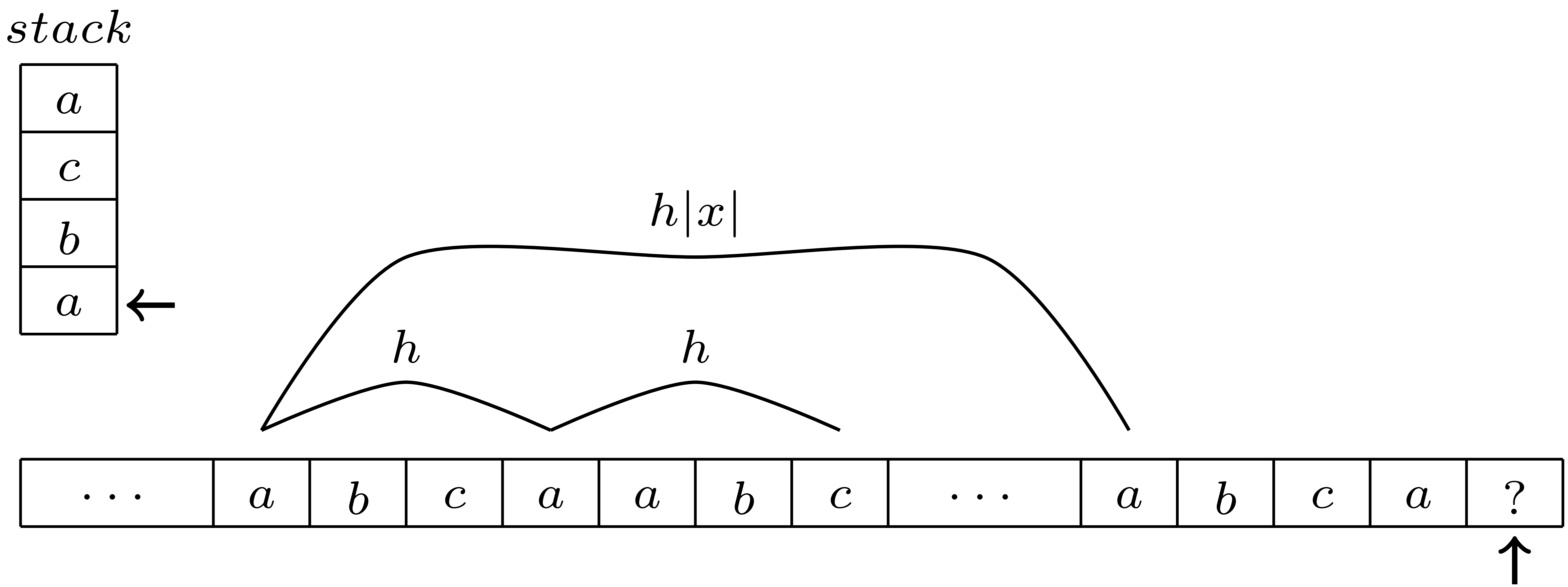
Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



Stack automaton which masters all purely periodic words

$$\alpha = x^\omega$$



Prediction Results

infinite words

automata

∃ masters → ∀	purely periodic	ultimately periodic	multilinear
DFA	×	×	×
DPDA	×	×	×
DSA	✓	?	?
multi-DFA	✓	✓	
sensing multi-DFA	✓	✓	

Multilinear Words

- An infinite word α is multilinear if it has the form

$$q \prod_{n \geq 1} \prod_{i \geq 1}^{m} p_i s_i^n$$

- Thus, α is broken into blocks, each consisting of m segments of the form $p_i s_i^n$.

- Example: $\prod_{n \geq 1} \mathtt{ab}^n \mathtt{c}^n = \mathtt{abcabbccabbbccc} \cdots$

- [Endrullis et al. 2011], [Smith 2013]

- Normal form (unless α is ultimately periodic):

  - $p_i \neq \varepsilon$, $s_i \neq \varepsilon$, $s_i[1] \neq p_{i+1}[1]$, and $s_m[1] \neq p_1[1]$

Predicting Multilinear Words

- We have seen that there is a two-head DFA which masters every ultimately periodic word.
- Can some multihead DFA master every multilinear word? Open problem.
- We consider sensing multihead DFAs, an extension of multihead DFAs able to sense, for each pair of heads, whether those two heads are at the same input position.
- [Smith 2016] Some sensing multihead DFA masters every multilinear word.

Algorithm which masters every multilinear word

- Uses a 10-head sensing DFA. Alternates between two procedures, correction and matching, with an increasing threshold k.

```
k = 0
loop
    k += 1
    correction procedure
    matching procedure
```

- The correction procedure tries to line up certain heads at segment boundaries so that the number of segments separating the heads is a multiple of m.
- The matching procedure tries to master the input α on the assumption that the correction procedure has successfully lined up the heads.

Correction Procedure

- Tries to line up the heads h_1, h_2, h_3, and h_4 to be k segments apart.
- k is a threshold which increases each time the procedure is entered.
- When the procedure is entered, $h_1 < h_2 < h_3 < h_4$.
- Uses a subroutine **advance** whose successful operation depends on k.

move h_1 until $h_1 = h_4$

advance h_1 by 1 segment

move h_2 until $h_2 = h1$

advance h_2 by k segments

move h_3 until $h_3 = h_2$

advance h_3 by k segments

move h_4 until $h_4 = h_3$

advance h_4 by k segments

advance subroutine

- Tries to advance a given head h_i past its current segment $p_j s_j^n$, leaving h_i at p_{j+1}.
- Uses a threshold k which increases between calls to the subroutine.
- Follows tortoise and hare algorithm until the number of consecutive correct guesses reaches k.
- Finally, moves t and h_i together until they disagree.

```
move t until t = h_i
move h_i
correct = 0
while correct < k
    if α[t] = α[h_i]
        correct += 1
    else
        correct = 0
        move h_i
    move t and h_i
while α[t] = α[h_i]
    move t and h_i
```

Matching Procedure

- Tries to master the multilinear word α.
- Works if h_1, h_2, h_3, and h_4 are a multiple of m segments apart, where m is the number of segments per block of α.
- Uses h_1, h_2, and h_3 to coordinate and predict $\alpha[h_4]$.
- If any guess is incorrect, exits so that the correction procedure can be called again.

```
loop
    move h3a until h3a = h3
    while α[h1] = α[h2] = α[h3] = α[h4]
        move h1, h2, h3a, h3
        move h4, guessing α[h2]
        exit procedure if guess was wrong
    while α[h2] = α[h3] = α[h4]
        move h2, h3
        move h4, guessing α[h3]
        exit procedure if guess was wrong
    while α[h3a] = α[h3] = α[h4]
        move h3a, h3
        move h4, guessing α[h3a]
        exit procedure if guess was wrong
    while h3a ≠ h3 and α[h3a] = α[h4]
        move h3a
        move h4, guessing α[h3a]
        exit procedure if guess was wrong
```

Prediction Results

infinite words

automata

∃ masters → ∀	purely periodic	ultimately periodic	multilinear
DFA	×	×	×
DPDA	×	×	×
DSA	✓	?	?
multi-DFA	✓	✓	?
sensing multi-DFA	✓	✓	✓

Further Work

- Consider other classes of automata and infinite words to see what connections can be made among them in a prediction setting.
- Open problems:
  - Can some DSA master every ultimately periodic word?
  - Can some (non-sensing) multi-DFA master every multilinear word?

Thank you!