Automata over infinite alphabets: Investigations in Fresh-Register Automata

Nikos Tzevelekos
Queen Mary University of London

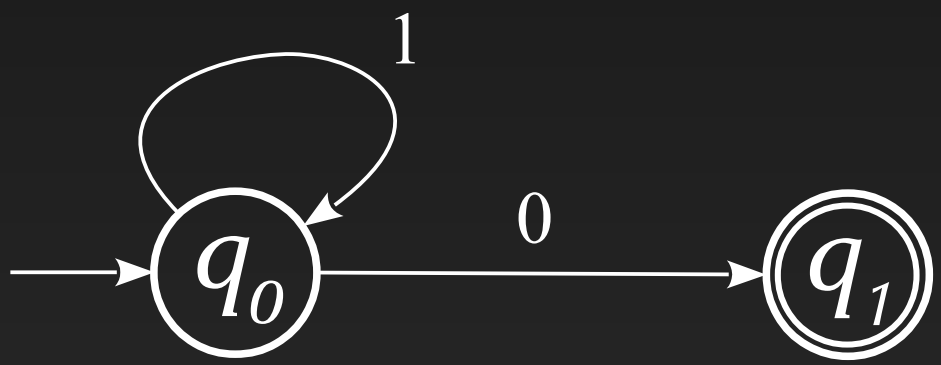
Andrzej Murawski & Steven Ramsay, University of Warwick
Radu Grigore, University of Oxford

10th Panhellenic Logic Symposium, Samos, June 2015

Supported by a Royal Academy of Engineering Research Fellowship

Automata theory

A theory of abstract machines / language acceptors:



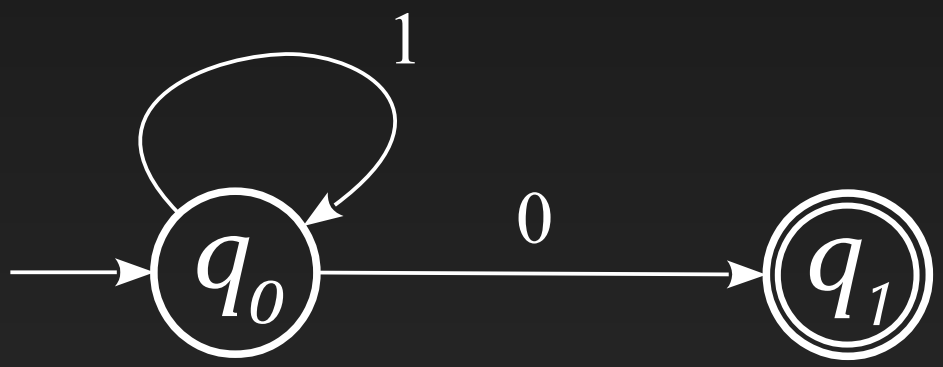
An automaton accepts some $L \subseteq \Sigma^*$

a finite alphabet

(e.g. $\Sigma = \{0,1\}$)

Automata theory

A theory of abstract machines / language acceptors:



An automaton accepts some $L \subseteq \Sigma^*$

a finite alphabet

- classify languages: e.g. regular, context-free, etc.
- algorithmic problems: reachability, membership, universality, inclusion, equivalence, etc.
- useful also for software verification

Why *infinite* alphabets

```
public void foo() {
  // Create new list
  List x = new ArrayList();

  x.add(1); x.add(2);
  Iterator i = x.iterator();
  Iterator j = x.iterator();
  i.next(); i.remove(); j.next();
}
```

Why *infinite* alphabets

```
public void foo() {
   // Create new list
   List x = new ArrayList();

   x.add(1); x.add(2);
   Iterator i = x.iterator();
   Iterator j = x.iterator();
   i.next(); i.remove(); j.next();
}
```

Finite alphabet not satisfactory for modelling or verifying resourceful code (and computation)

Lots of interest also from XML model-checking community!

What this talk is about

1. This talk is about automata over infinite alphabets

2. We take motivation from program modelling & verification

3. We concentrate on infinite alphabet extensions of

- *Finite-State Automata* and
- *Pushdown Automata*

by addition of name registers and freshness oracles

4. We look at their expressiveness and algorithmic properties with focus on non-emptiness (reachability) and bisimilarity

Example formal languages

Let $\Sigma = \{a_1, a_2, \ldots, a_n, \ldots\}$ be an infinite alphabet of names

can only be compared for equality

Example formal languages

Let $\Sigma = \{a_1, a_2, \ldots, a_n, \ldots\}$ be an infinite alphabet of names

- $L_0 = \{\, abc \in \Sigma^3 \mid a \neq b, c \in \{a, b\} \,\}$
- $L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$
- $L_2 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \,\}$
- $L_{\text{new}} = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$
- $L_4 = \{\, a_1 \ldots a_n b_1 \ldots b_m \in \Sigma^* \mid \forall i \neq j, i' \neq j'.\ a_i \neq a_j, b_{i'} \neq b_{j'} \,\}$
- $L_5 = \{\, a_1 a_2 \ldots a_n a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$

Automata theory in infinite alphabets

Let $\Sigma = \{a_1, a_2, \ldots, a_n, \ldots\}$ be an infinite alphabet of names

Automata theory in infinite alphabets

Let $\Sigma = \{a_1, a_2, \ldots, a_n, \ldots\}$ be an infinite alphabet of names

- examine languages over Σ^*
  - or, languages over $(F \cup \Sigma)^*$
  - or, languages over $(F \times \Sigma)^*$
    - usually called *data words* (XML)

a finite set of constants

Automata theory in infinite alphabets

Let $\Sigma = \{a_1, a_2, \ldots, a_n, \ldots\}$ be an infinite alphabet of names

- examine languages over Σ^*
  - or, languages over $(F \cup \Sigma)^*$
  - or, languages over $(F \times \Sigma)^*$
    - usually called *data words* (XML)

a finite set of constants

- look for notions of regularity, context-freeness, etc.
- devise effective algorithms for non-emptiness, membership, etc.

[Kaminski & Francez '94]

Register Automata (RA)

Let $\Sigma = \{a_1, a_2, \ldots, a_n, \ldots\}$ be an infinite alphabet of names

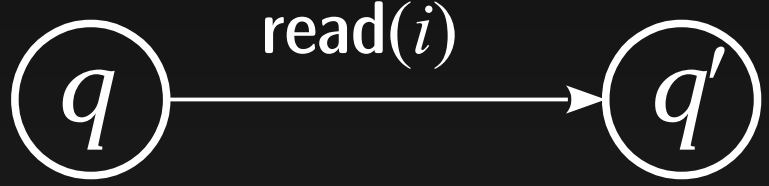
finitely many (say R) registers

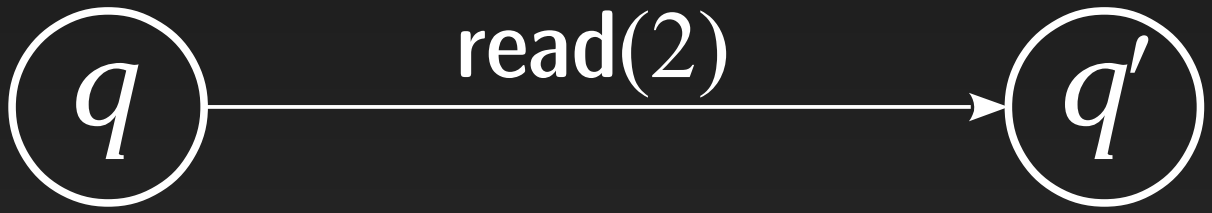
registers store names

Label λ of the form:

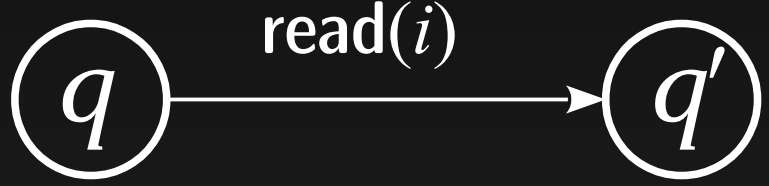
- **read**(i), $i \in \{1,\ldots,R\}$
- **fresh**(i), $i \in \{1,\ldots,R\}$

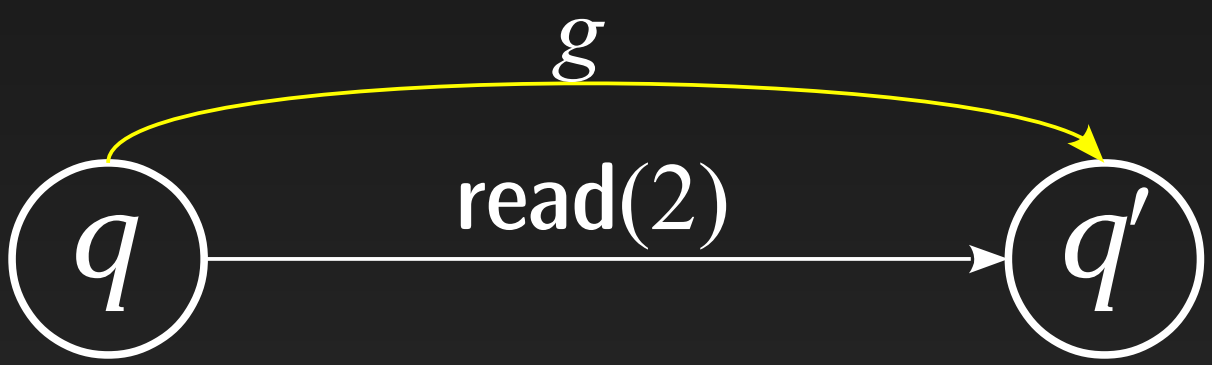
Transitions:



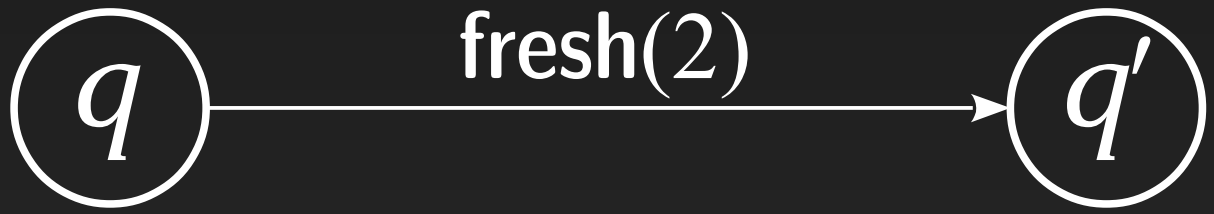


Transitions:



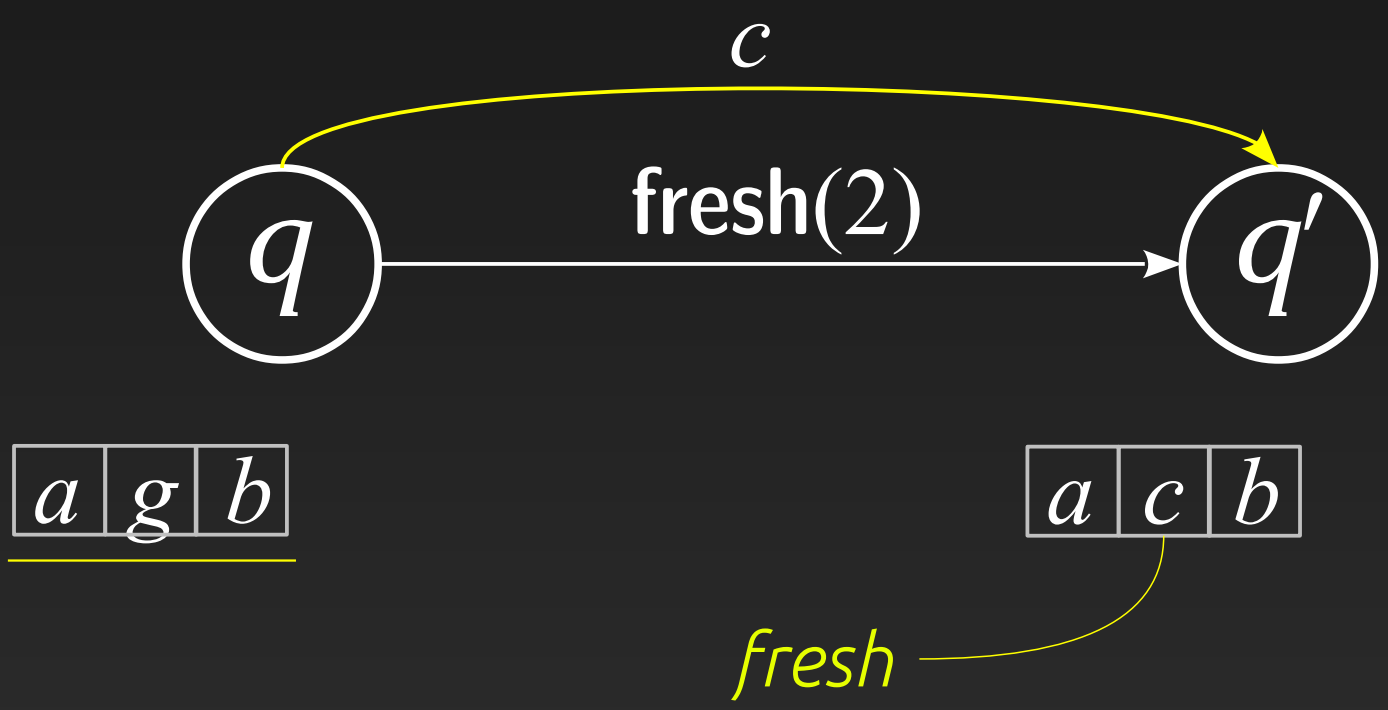


Transitions:



a	g	b

Transitions:

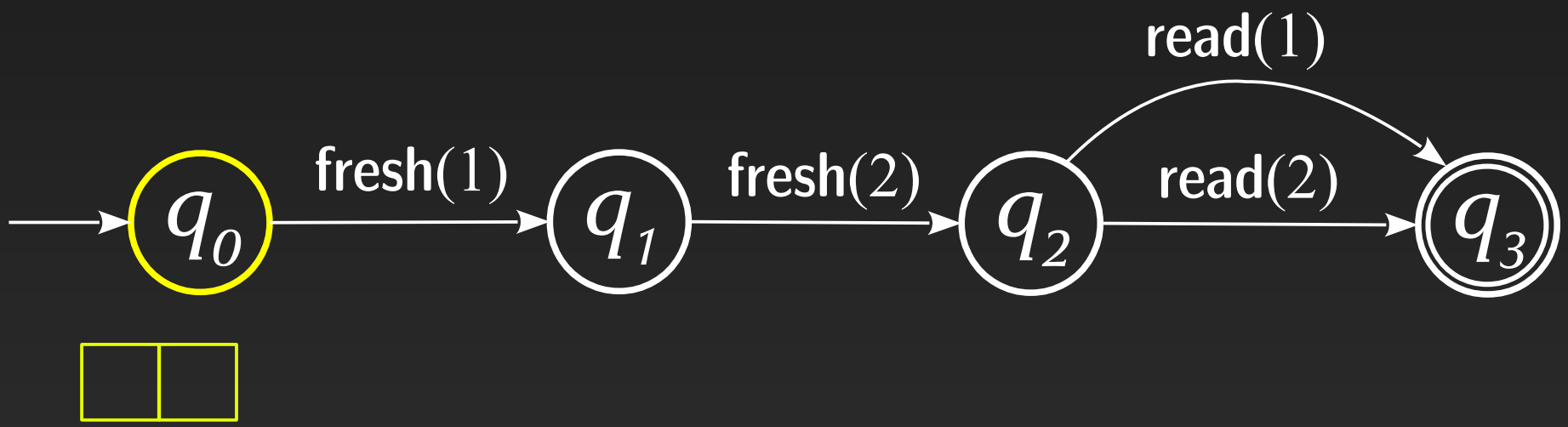


Example

$$L_0 = \{\, abc \in \Sigma^3 \mid a \neq b, c \in \{a,b\} \,\}$$

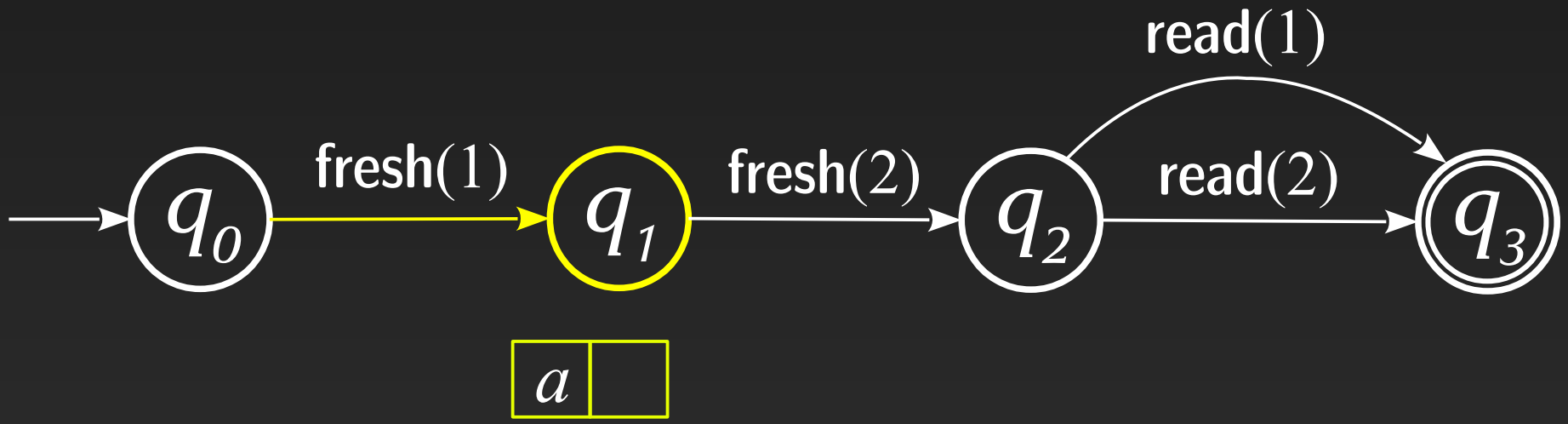
Example

$$L_0 = \{\, abc \in \Sigma^3 \mid a \neq b, c \in \{a,b\} \,\}$$



Example

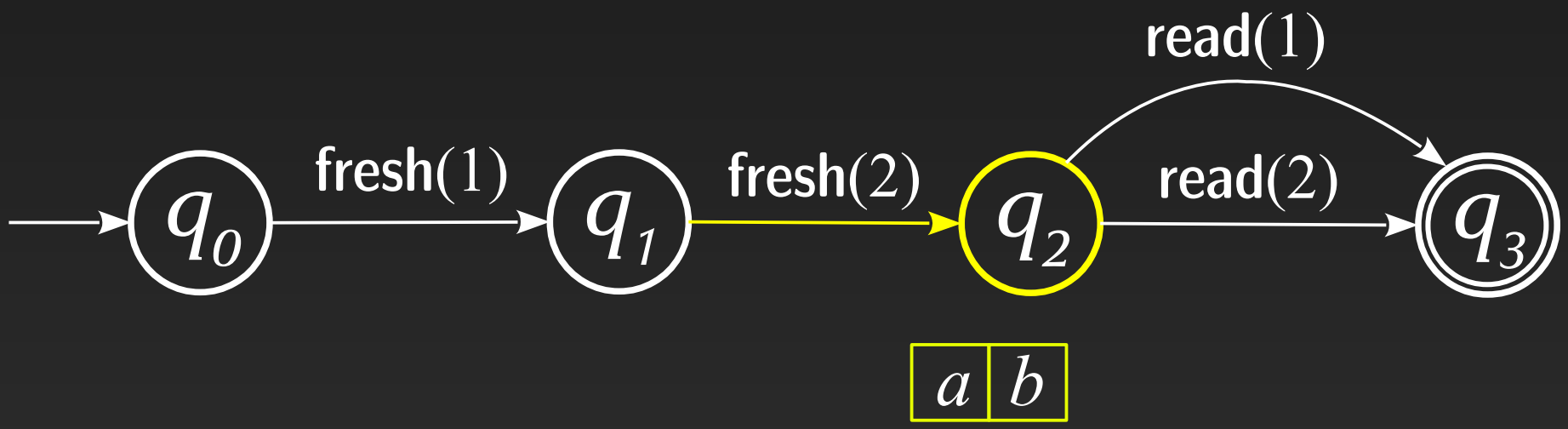
$$L_0 = \{\, abc \in \Sigma^3 \mid a \neq b, c \in \{a,b\} \,\}$$



a

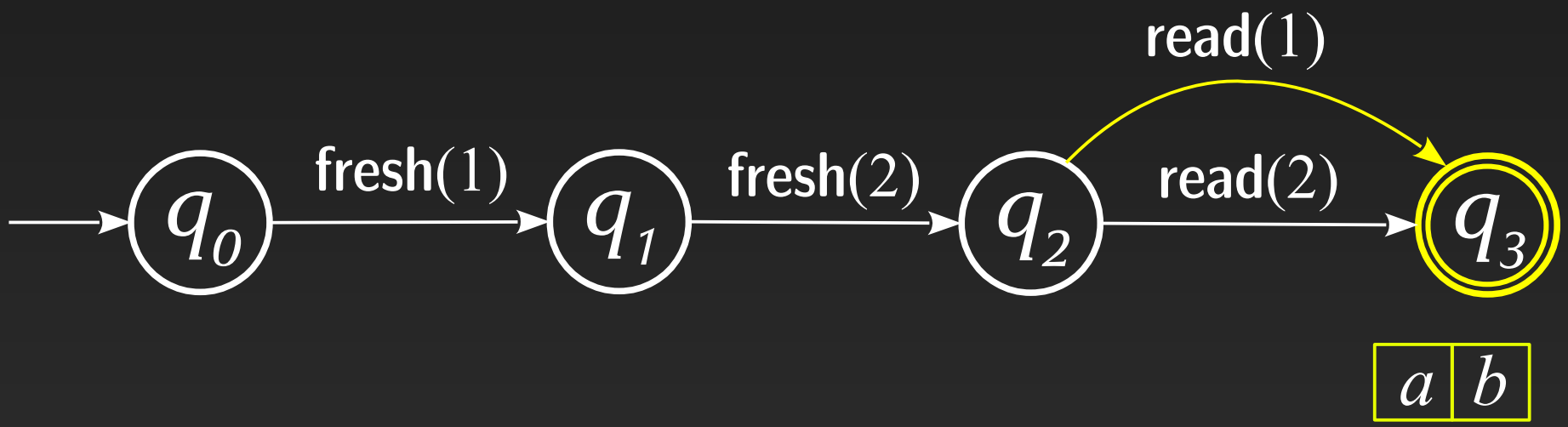
Example

$$L_0 = \{\, abc \in \Sigma^3 \mid a \neq b, c \in \{a,b\} \,\}$$



Example

$$L_0 = \{\, abc \in \Sigma^3 \mid a \neq b, c \in \{a,b\} \,\}$$



$a\,b\,a$

Example

$$L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$$

(all strings where each name is distinct from its predecessor)

Example

$$L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$$

(all strings where each name is distinct from its predecessor)

a

Example

$$L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$$

(all strings where each name is distinct from its predecessor)

$a\,b$

Example

$$L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$$

(all strings where each name is distinct from its predecessor)

$a\,b\,c$

Example

$$L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$$

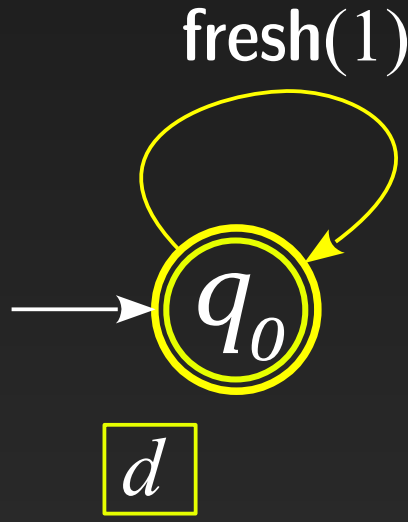
(all strings where each name is distinct from its predecessor)

$a\,b\,c\,a$

Example

$$L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$$

(all strings where each name is distinct from its predecessor)



$a\,b\,c\,a\,d$

Example

$$L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$$

(all strings where each name is distinct from its predecessor)

$a\,b\,c\,a\,d\,e$

Example

$$L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$$

(all strings where each name is distinct from its predecessor)

$a\,b\,c\,a\,d\,e\,b$

Example

$$L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$$

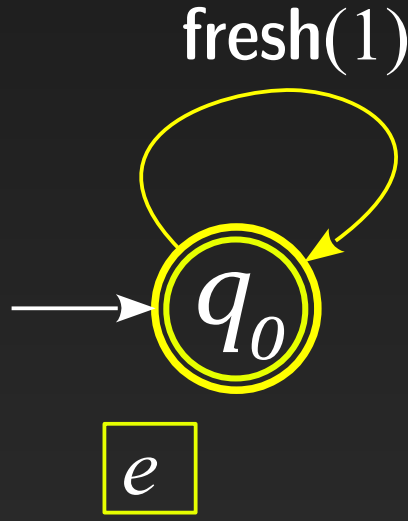
(all strings where each name is distinct from its predecessor)

$$a\,b\,c\,a\,d\,e\,b\,a\,g\,c\,a\,b$$

Example

$$L_1 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \,\}$$

(all strings where each name is distinct from its predecessor)

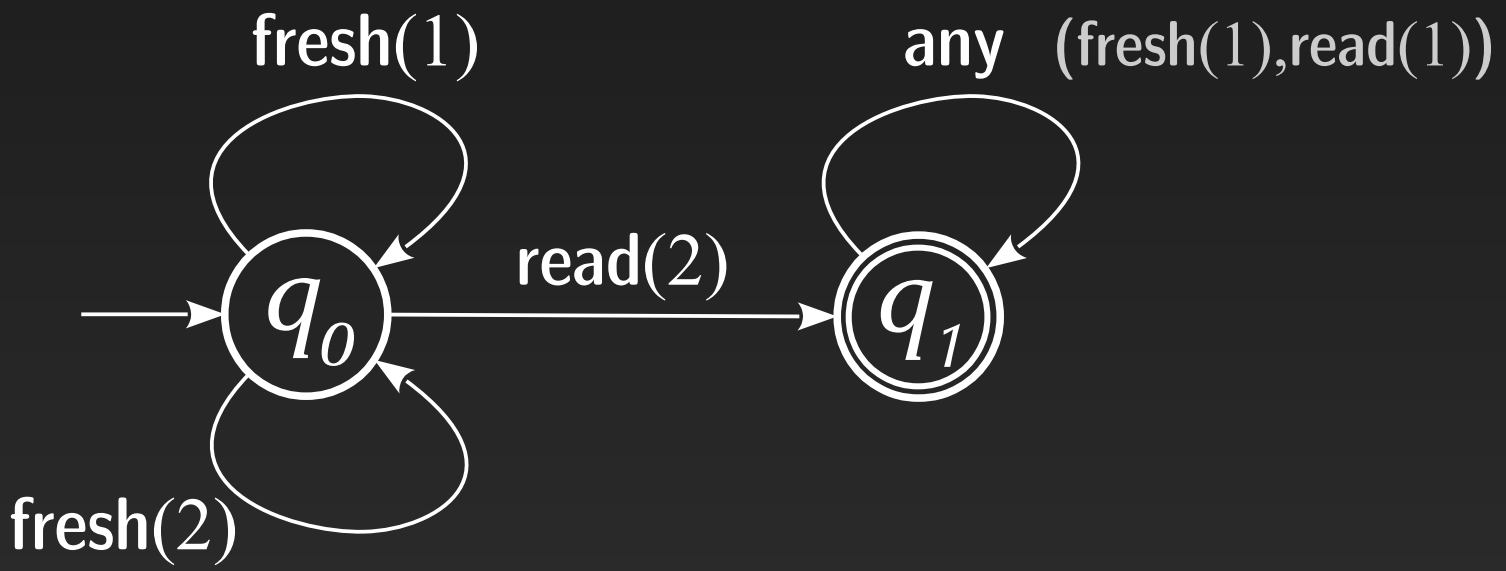


$$a\,b\,c\,a\,d\,e\,b\,a\,g\,c\,a\,b\ a\,n\,d\ w\,e\ l\,o\,v\,e\ c\,a\,k\,e$$

More examples

$$L_2 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \,\}$$

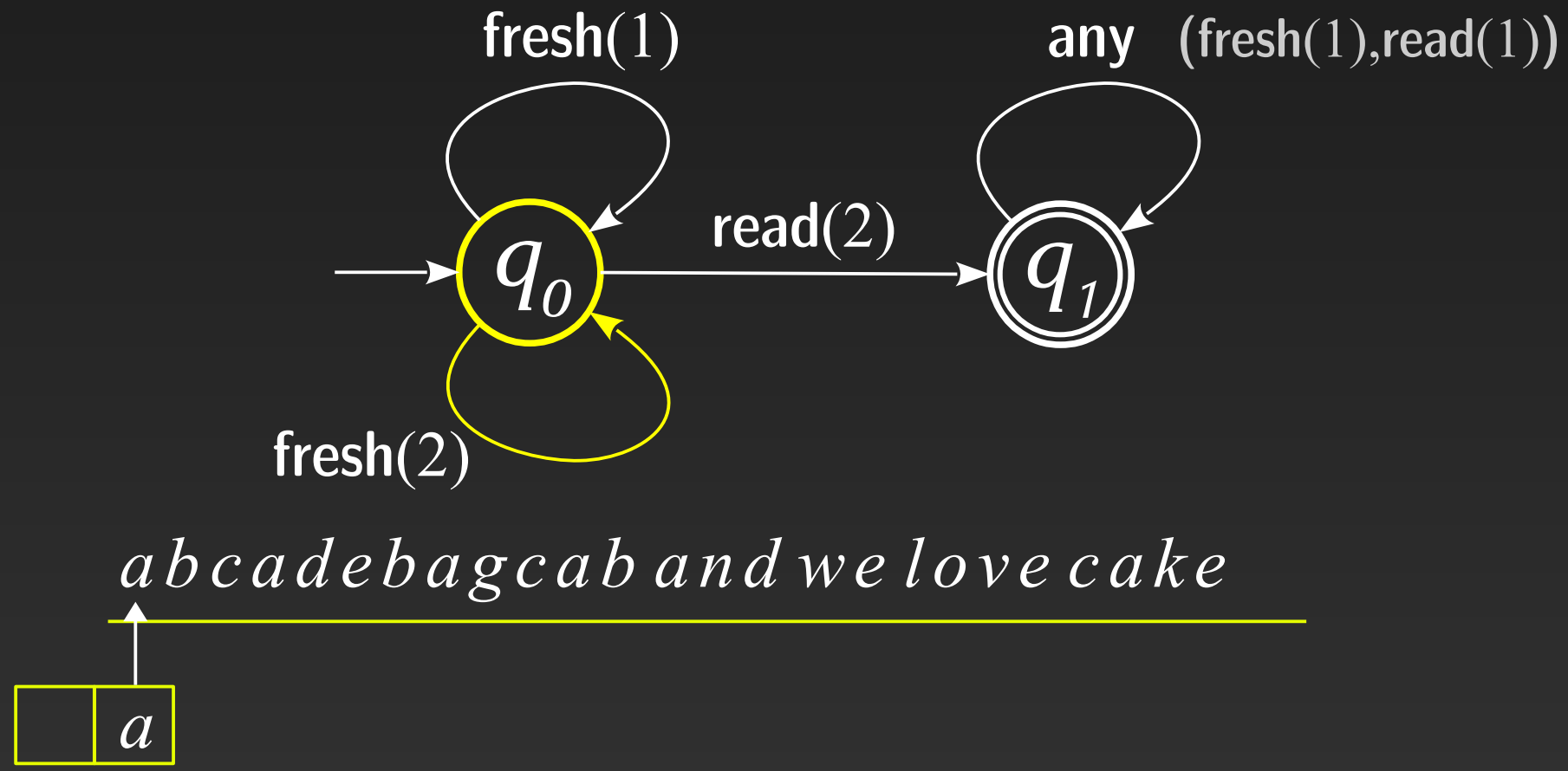
(all strings where some name is repeated)



More examples

$$L_2 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \,\}$$

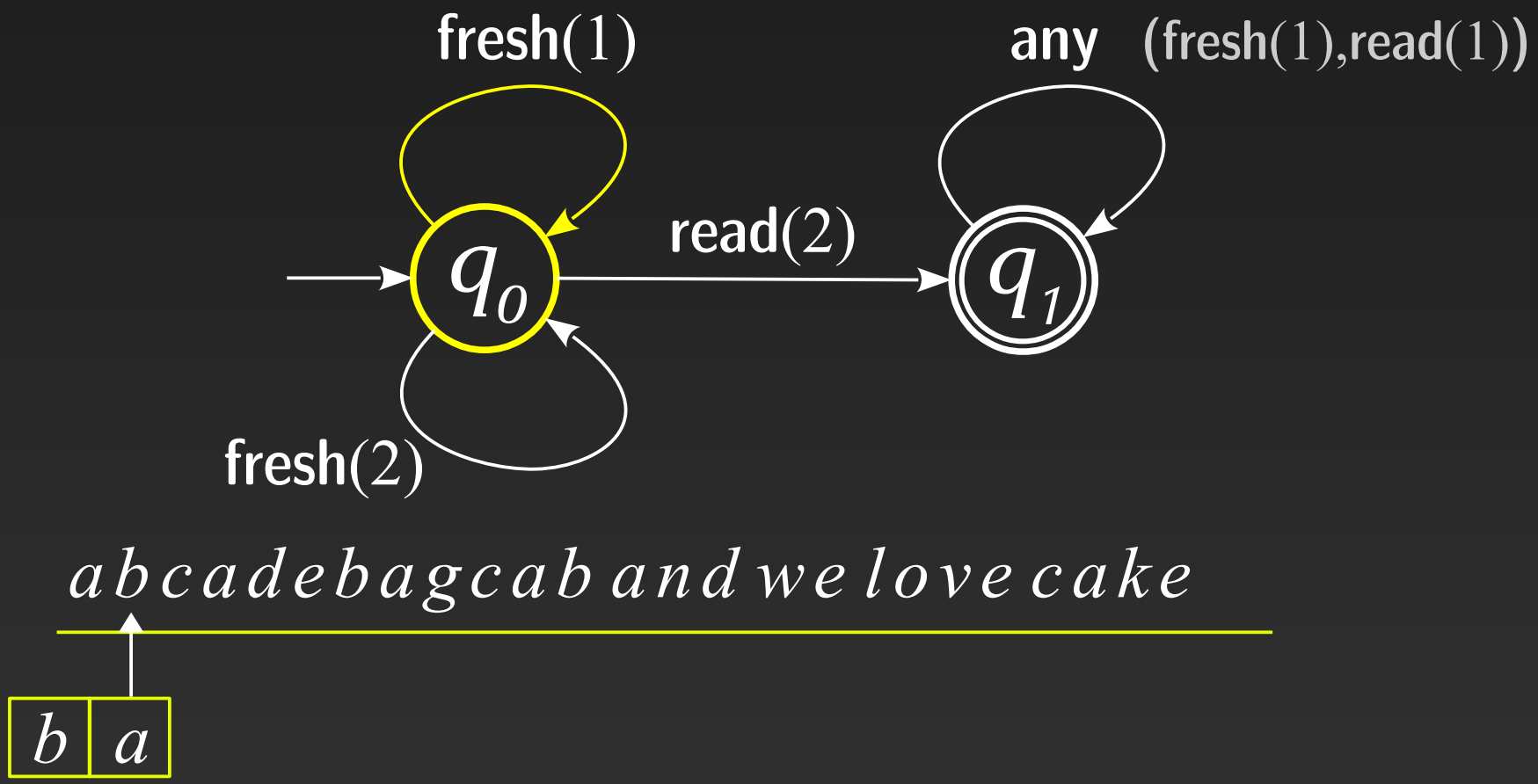
(all strings where some name is repeated)



More examples

$$L_2 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \,\}$$

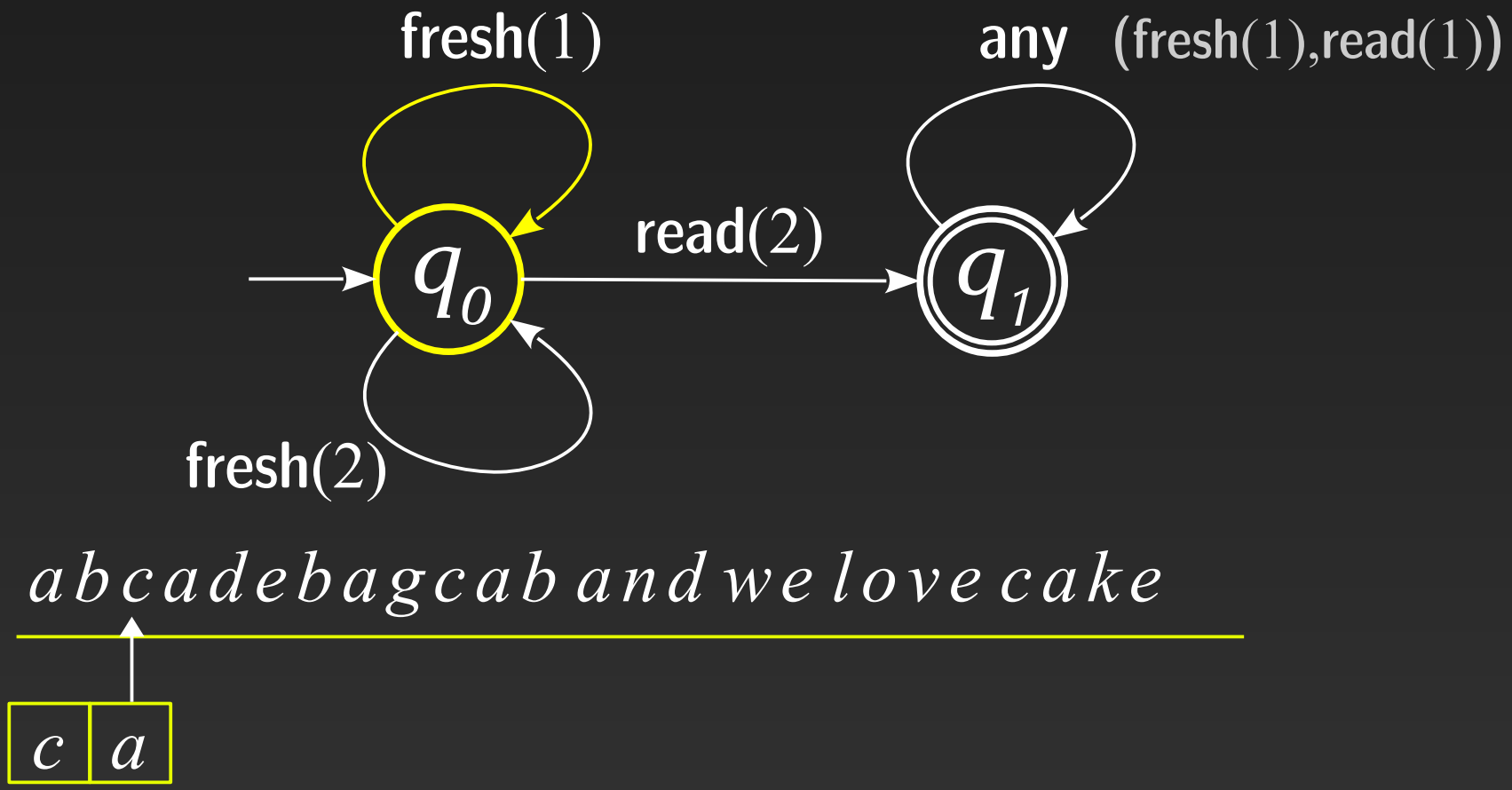
(all strings where some name is repeated)



More examples

$$L_2 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \,\}$$

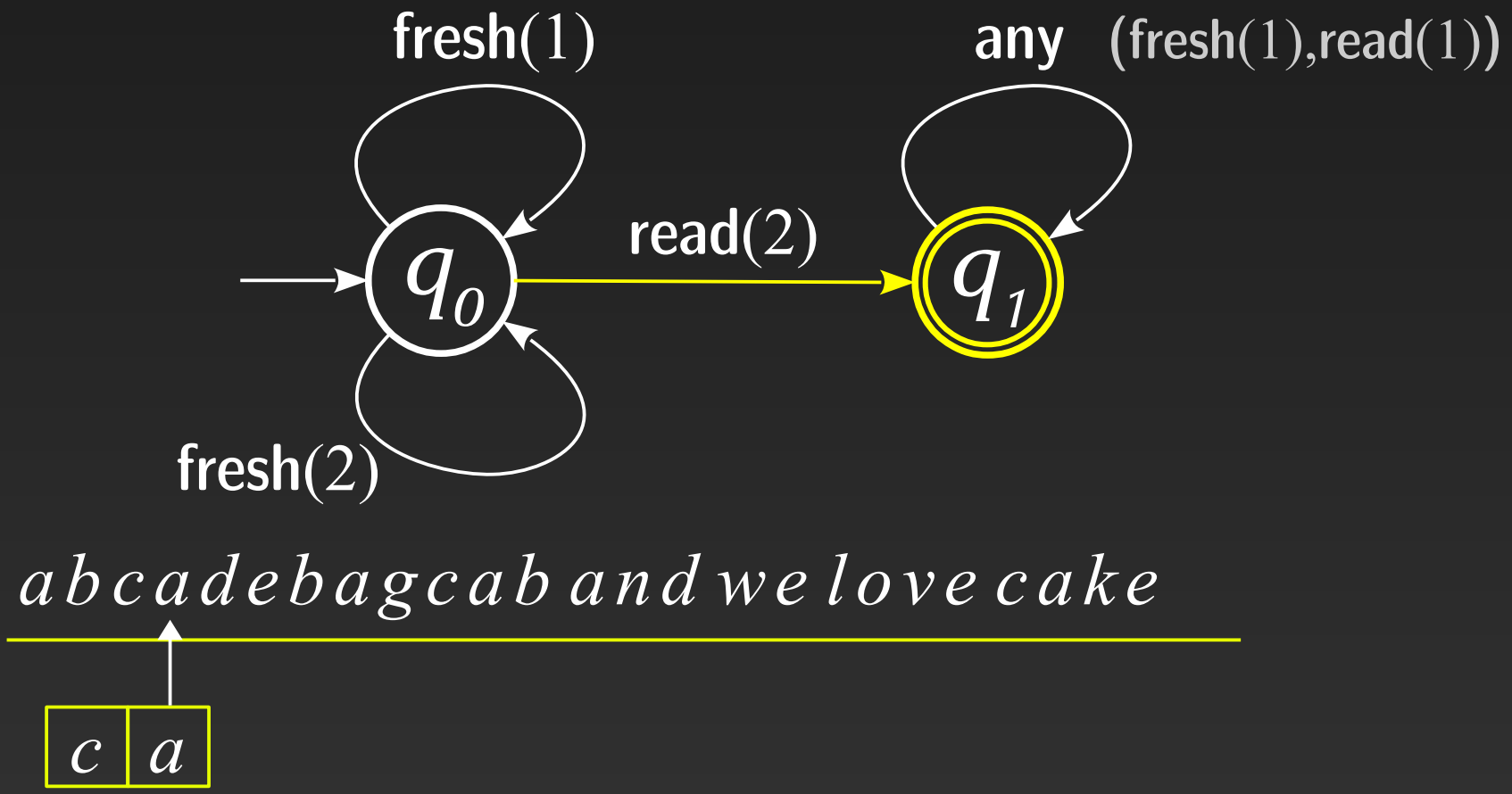
(all strings where some name is repeated)



More examples

$$L_2 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \,\}$$

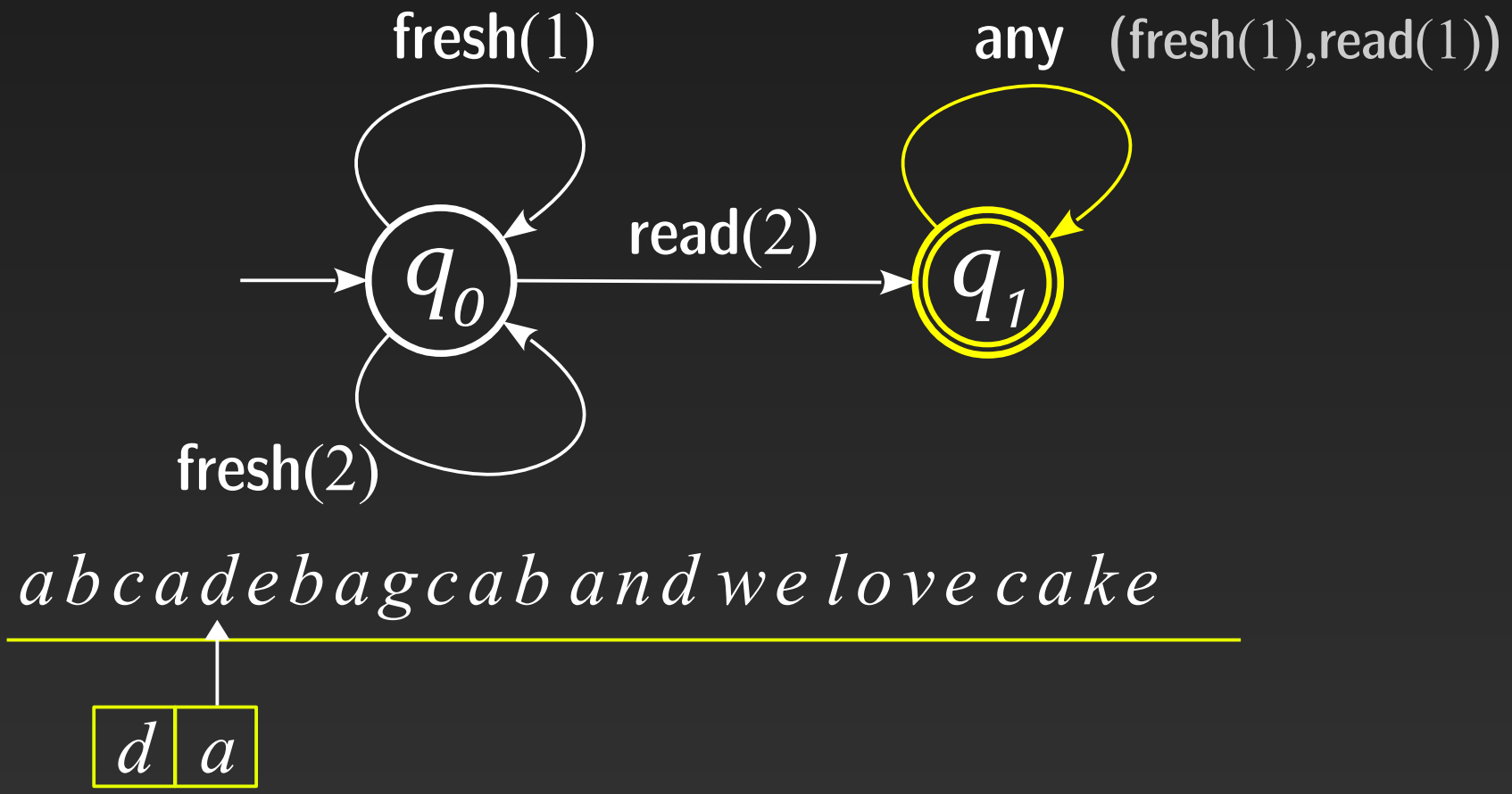
(all strings where some name is repeated)



More examples

$$L_2 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \,\}$$

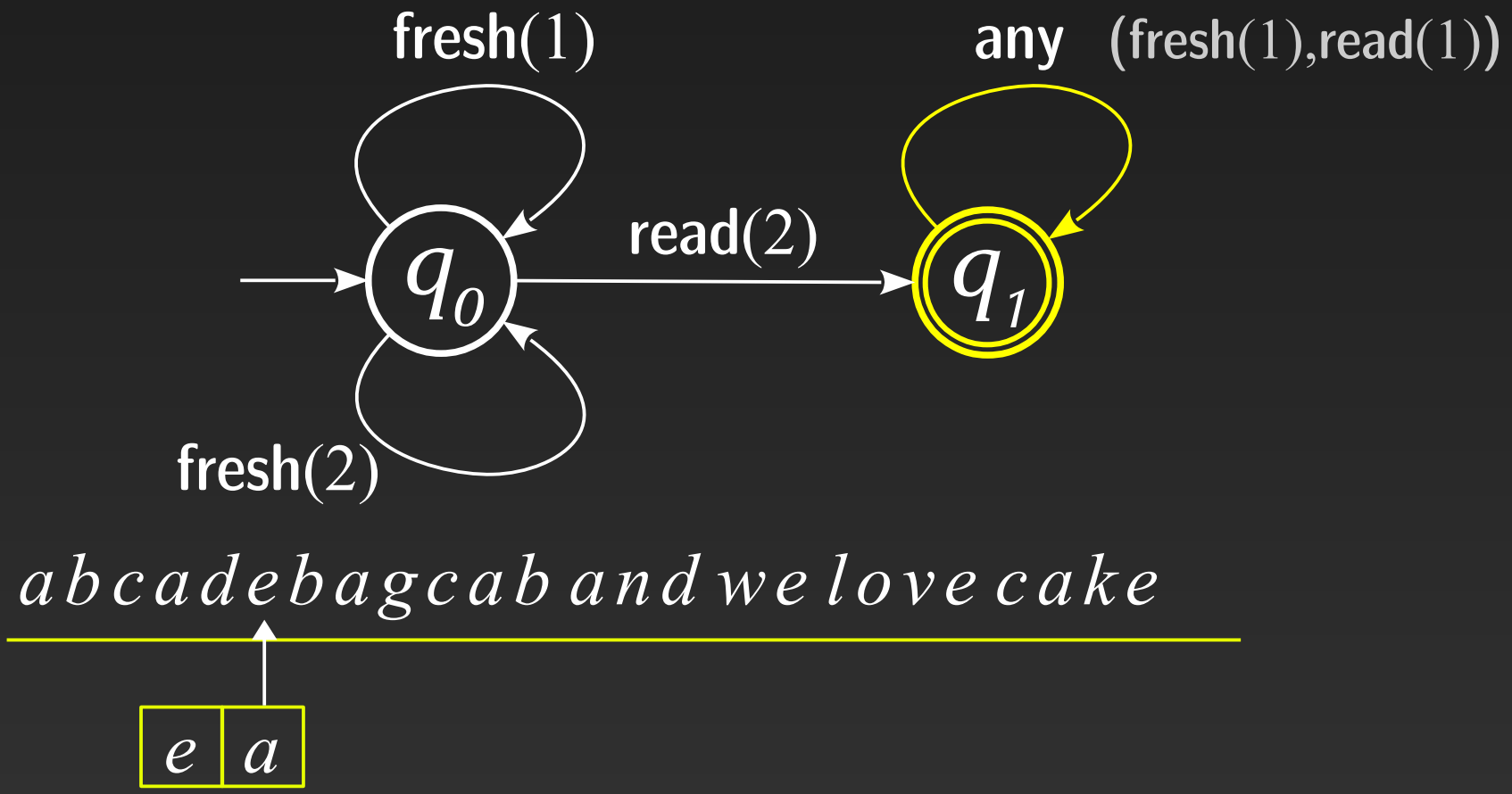
(all strings where some name is repeated)



More examples

$$L_2 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \,\}$$

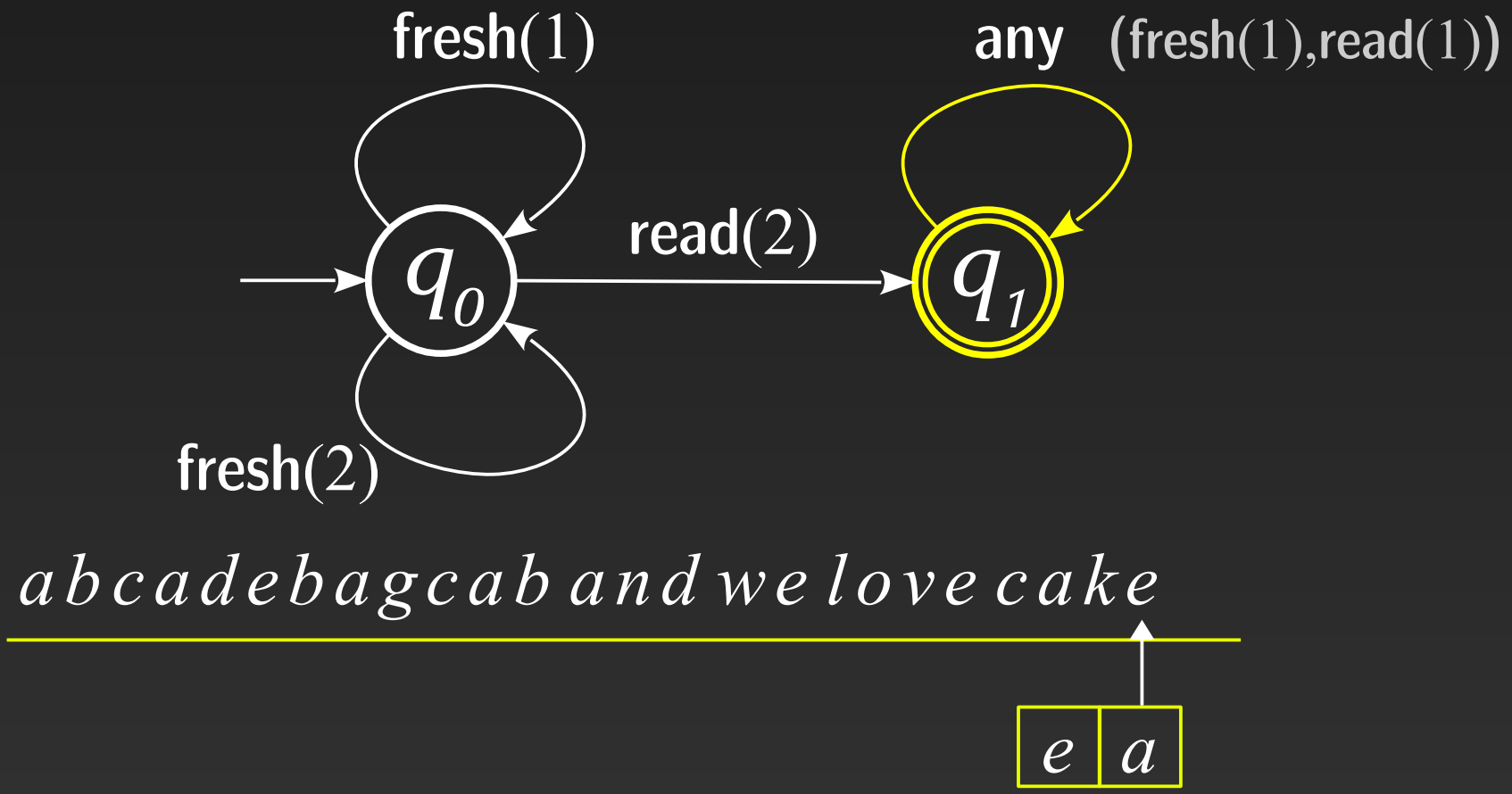
(all strings where some name is repeated)



More examples

$$L_2 = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \,\}$$

(all strings where some name is repeated)



More examples

$$L_2 = \{ a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \}$$

(all strings where some name is repeated)

complement

$$L_{\text{new}} = \{ a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \}$$

(all strings of pairwise distinct names)

More examples

$$L_2 = \{ a_1 a_2 ... a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \}$$

(all strings where some name is repeated)

complement

$$L_{\text{new}} = \{ a_1 a_2 ... a_n \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \}$$

(all strings of pairwise distinct names)

this is **not** RA recognisable, by “Pumping Lemma”:

If an RA A accepts some word of size n then A also accepts one of the same size with at most $R+1$ names

Register automata properties

- Capture regularity when Σ restricted to finite
  - Closed under $\cup, \cap, \bullet, *$.
  - not closed under complement & not determinisable

[Kaminski & Francez '94]

- Universality / equivalence undecidable

[Neven, Schwentick & Vianu '01]

- Decidable non-emptiness: NP-complete
  - complexity depends on register “mode” (NL → NP → PSPACE)

[Sakamoto & Ikeda '00; Demri & Lazić '09]

Example revisited

```
public void foo() {
  // Create new list
  List x = new ArrayList();

  x.add(1); x.add(2);
  Iterator i = x.iterator();
  Iterator j = x.iterator();
  i.next(); i.remove(); j.next();
}
```

here is a safety property φ:

if an iterator modifies its collection x then other iterators of x become invalid

e.g. the code on the left is bad.

We can express such “chaining” properties using RAs

- and dynamically verify them

[Grigore, Distefano, Petersen & T. '13]

Example revisited

```
public void foo() {
  // Create new list
  List x = new ArrayList();

  x.add(1); x.add(2);
  Iterator i = x.iterator();
  Iterator j = x.iterator();
  i.next(); i.remove(); j.next();
}
```

here is a safety property φ:

if an iterator modifies its collection x then other iterators of x become invalid

e.g. the code on the left is bad.

We can express such “chaining” properties using RAs

- and dynamically verify them

[Grigore, Distefano, Petersen & T. '13]

However, RAs do not capture `new` and hence cannot give abstract models of such programs (e.g. for static analysis)

[T. '11]

Fresh-Register Automata (FRA)

Let $\Sigma = \{a_1, a_2, \ldots, a_n, \ldots\}$ be an infinite alphabet of names

finitely many (say R) registers

registers store names

Label λ of the form:

- **read**(i), $i \in \{1,\ldots,R\}$
- **fresh**(i), $i \in \{1,\ldots,R\}$
- **gl-fresh**(i), $i \in \{1,\ldots,R\}$

global freshness oracle

[T. '11]

Fresh-Register Automata (FRA)

Let $\Sigma = \{a_1, a_2, \ldots, a_n, \ldots\}$ be an infinite alphabet of names

finitely many (say R) registers

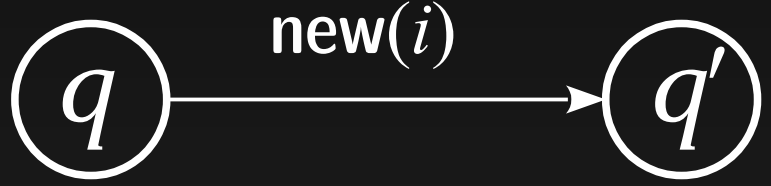
registers store names

Label λ of the form:

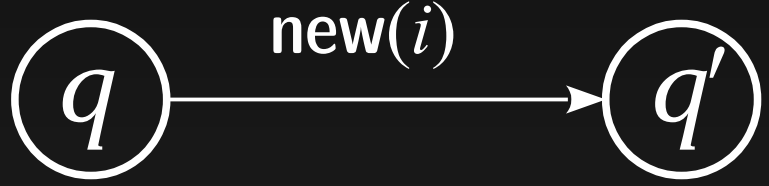
- **read**(i), $i \in \{1,\ldots,R\}$
- **fresh**(i), $i \in \{1,\ldots,R\}$
- **new**(i), $i \in \{1,\ldots,R\}$

global freshness oracle

Transitions:



Transitions:

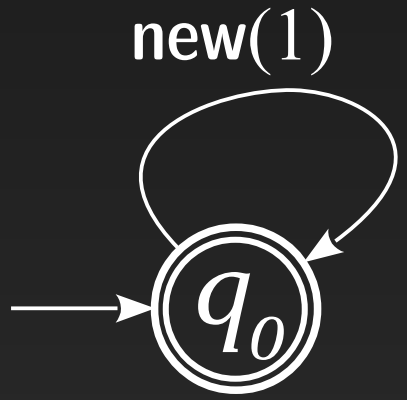


globally fresh

Examples

$$L_{\text{new}} = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

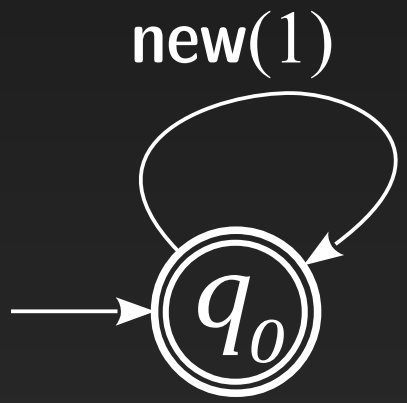
(all strings of pairwise distinct names)

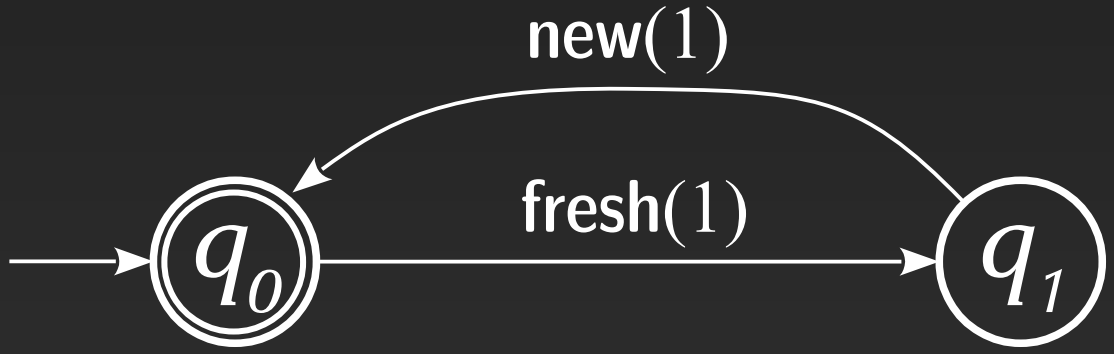


Examples

$$L_{\text{new}} = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

(all strings of pairwise distinct names)





$$L_3 = \{\, a_1 a_2 \ldots a_{2n} \in \Sigma^* \mid \forall i < 2n.\ a_i \neq a_{i+1} \quad \forall i \leq n,\, j < 2i.\ a_j \neq a_{2i} \,\}$$

FRA properties

- Closed under $\cup$, $\cap$.
- *Not* closed under $\bullet$, *.

FRA properties

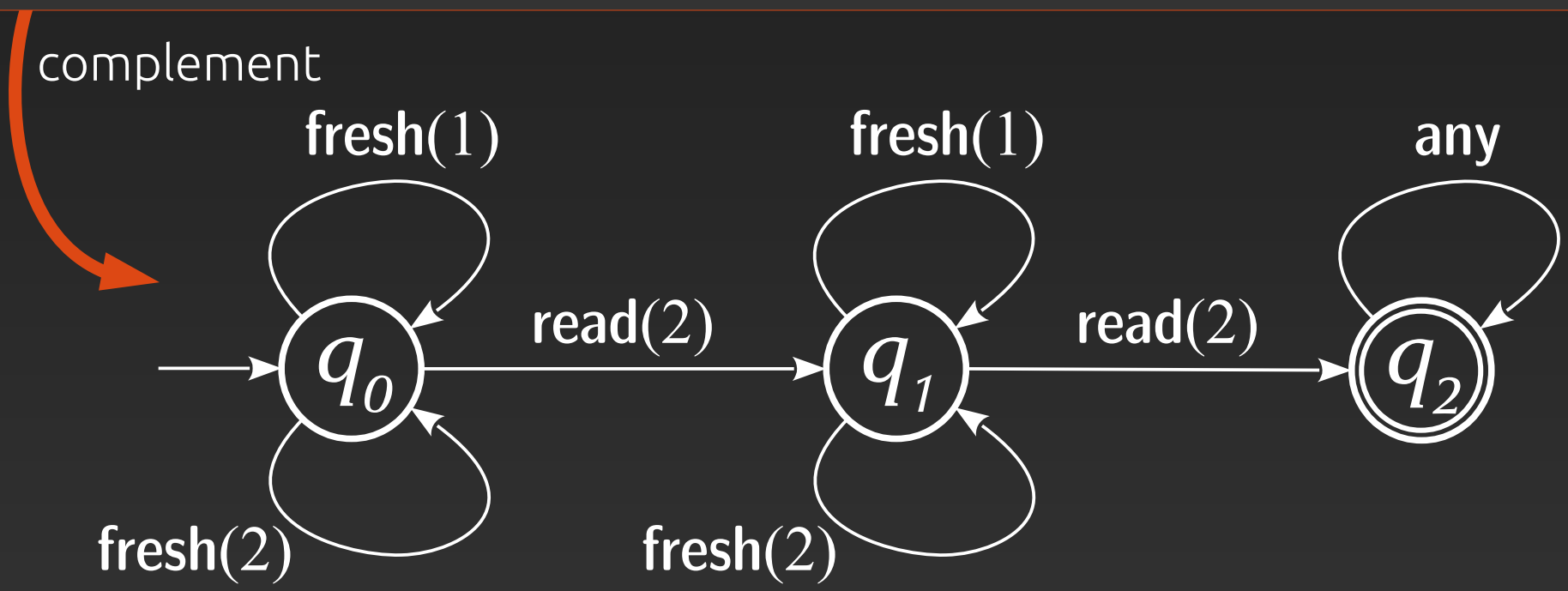
- Closed under $\cup$, $\cap$.
- *Not* closed under $\bullet$, *.

$$L_{\text{new}} \bullet L_{\text{new}} = \{\, a_1 \ldots a_n\, b_1 \ldots b_m \in \Sigma^* \mid \forall i \neq j, i' \neq j'.\ a_i \neq a_j, b_{i'} \neq b_{j'} \,\}$$

FRA properties

- Closed under $\cup$, $\cap$.
- *Not* closed under $\bullet$, *.

$$L_{\text{new}} \bullet L_{\text{new}} = \{\, a_1 \ldots a_n\, b_1 \ldots b_m \in \Sigma^* \mid \forall i \neq j, i' \neq j'.\ a_i \neq a_j, b_{i'} \neq b_{j'} \,\}$$



FRA properties

- Closed under $\cup$, $\cap$.
- *Not* closed under $\bullet$, *.
- Not closed under complement & not determinisable

[T. '11]

- Universality / equivalence undecidable

[Neven, Schwentick & Vianu '01]

- Decidable non-emptiness: NP-complete
  - complexity depends on register “mode” (NL → NP → PSPACE)

[Sakamoto & Ikeda '00; Demri & Lazić '09]

Application: program verification

The modelling power of FRAs can be used to model resourceful programs via game semantics

Program → game model

model is *fully abstract*:

two programs are equivalent ⇔ their game models are the same

Application: program verification

The modelling power of FRAs can be used to model resourceful programs via game semantics

Program → game model → FRA

effectively:

two programs are equivalent ⇔ their FRAs are language equivalent / bisimilar

Application: program verification

The modelling power of FRAs can be used to model resourceful programs via game semantics

Program → game model → FRA

effectively:

two programs are equivalent ⇔ their FRAs are language equivalent / bisimilar

Nominal Algorithmic Game Semantics:

- decision procedures for ML fragments [Murawski & T. '11, '12]
- same for Interface Middleweight Java [Murawski, Ramsay & T.]

Investigations in FRAs

Investigations in FRAs

Global freshness: from oracles to histories

- History Register Automata [Grigore & T. '13]

Context-freeness: Pushdown FRA [Cheng & Kaminski '98; Segoufin '06] [Murawski & T. '12]

- Reachability EXPTIME-complete
- Global reachability via “saturation” [Murawski, Ramsay & T. '14]

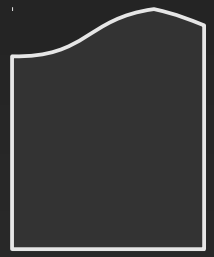
Bisimilarity for FRA (complexity)

- PSPACE-c (depending on modes: NP? → PSPACE → EXPTIME)
- approach uses permutation group theory [Murawski, Ramsay & T.' 15]

[Grigore & T. '13]

(unary) History-Register Automata (HRA)

Label λ of the form:



finitely many (say R) registers

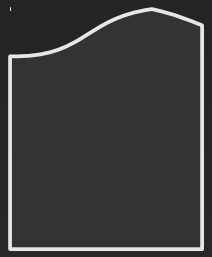
unbounded history set

registers & history store names

[Grigore & T. '13]

(unary) History-Register Automata (HRA)

finitely many (say R) registers



unbounded history set

registers & history store names

Label λ of the form:

- **read**(i,j,X,Y)

  $i,j \in \{0,1,\ldots,R\}$

  $X,Y \in \{$ in, out $\}$

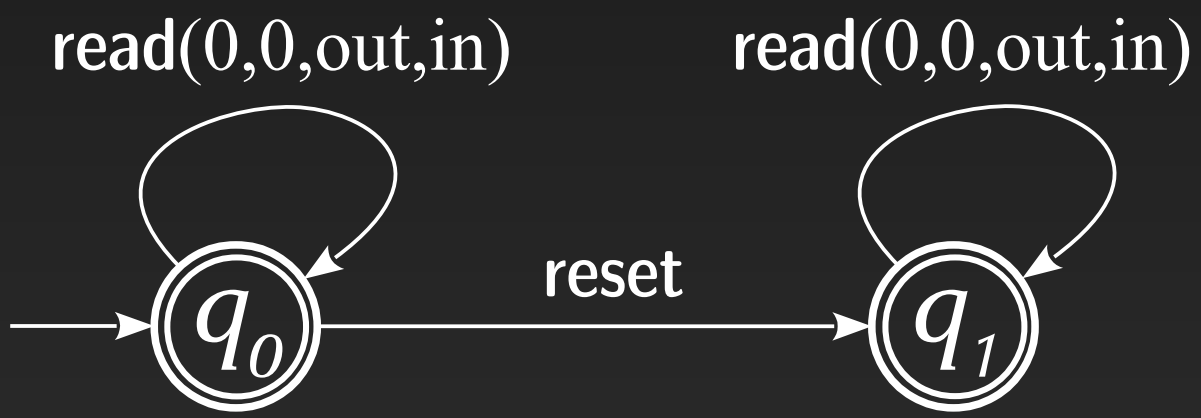
- **reset**

before: name in register i, and in/out history (X)

after: name in register j, and in/out history (Y)

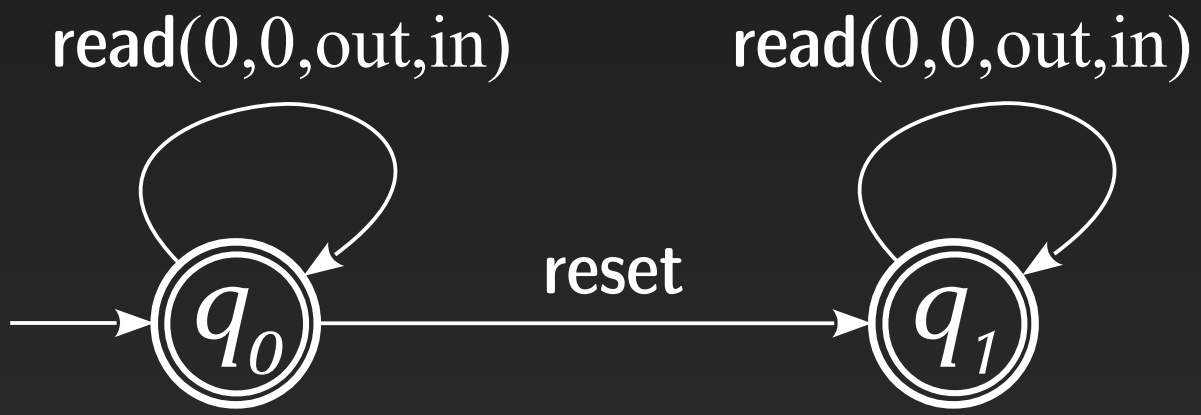
Example

$$L_{\text{new}} \bullet L_{\text{new}} = \{\, a_1 \ldots a_n\, b_1 \ldots b_m \in \Sigma^* \mid \forall i \neq j, i' \neq j'.\ a_i \neq a_j, b_{i'} \neq b_{j'} \,\}$$



Example

$$L_{\text{new}} \bullet L_{\text{new}} = \{\, a_1 \ldots a_n\, b_1 \ldots b_m \in \Sigma^* \mid \forall i \neq j, i' \neq j'.\ a_i \neq a_j, b_{i'} \neq b_{j'} \,\}$$



read(0,0,out,in)

before: name not in registers, and not in history

after: name not in registers, and in history

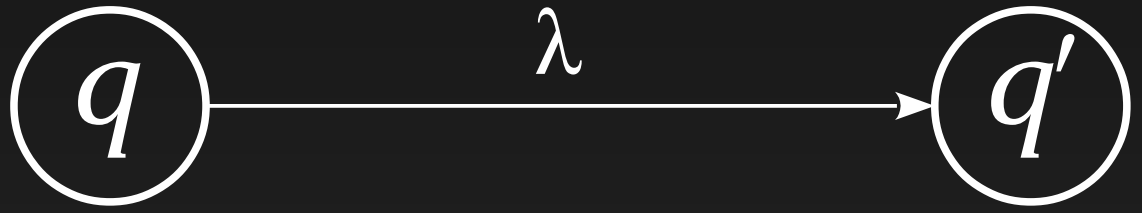
Unary HRA properties (1 history)

- Closed under $\cup$, $\bullet$, *.
- *Not* closed under $\cap$.
- Not closed under complement & not determinisable

- Universality / equivalence undecidable
- Decidable non-emptiness in cubic space

[Grigore & T. '13]

History-Register Automata (HRA)



finitely many (say R) registers

finitely many (say H) histories

registers & histories store names

Label λ of the form:

- (X,Y)
  $X,Y \subseteq \{1,\ldots,R\} \cup \{1,\ldots,H\}$
- **reset**

before: name in registers and histories X

after: name in registers and histories Y

Example: name consumption

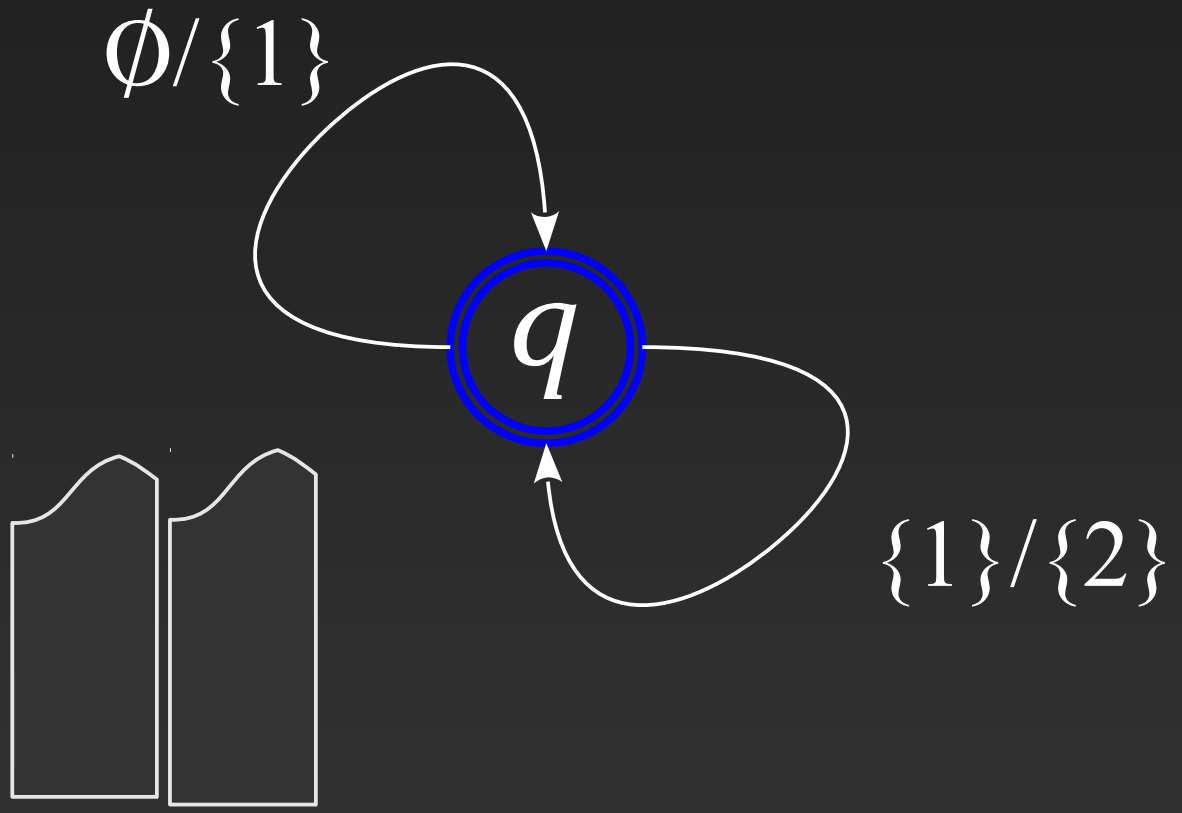
$$P \mid\mid C$$

***P** produces* fresh names, and ***C** consumes* them

$$L_{\text{new}} = \{\, a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

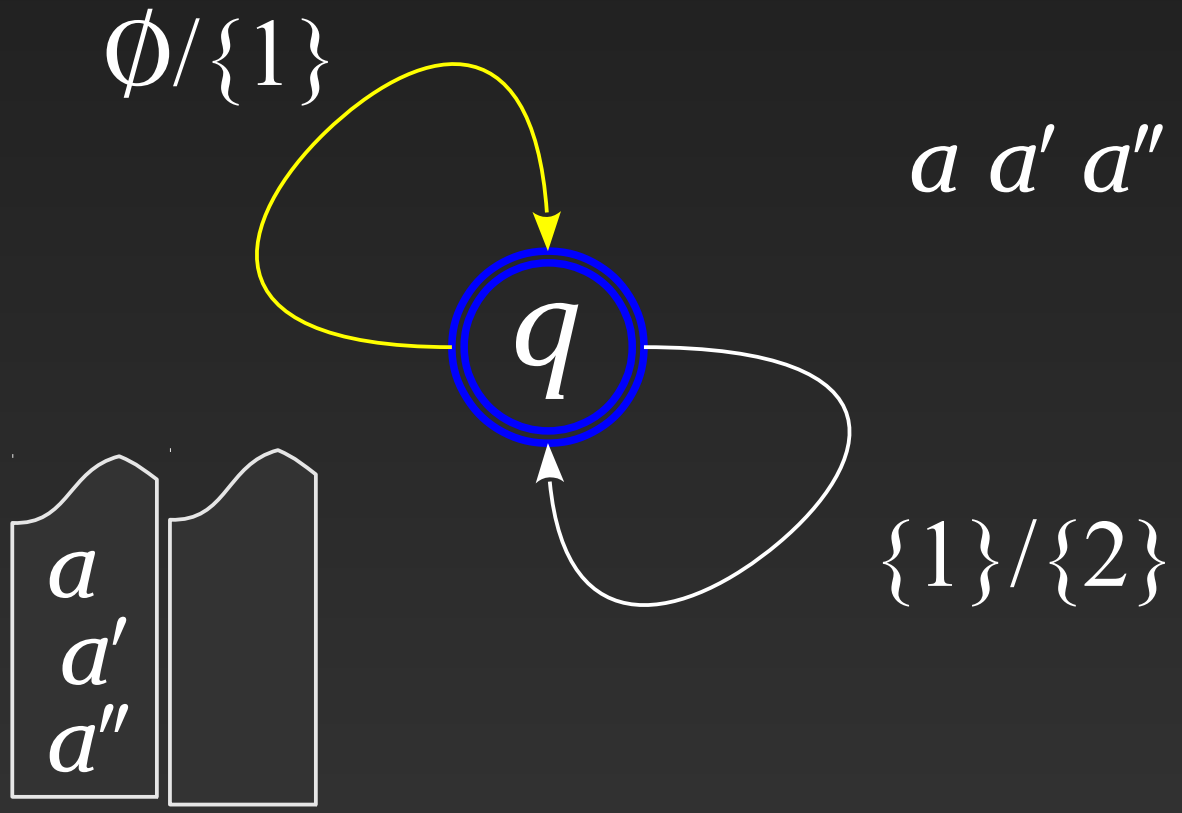
Example: name consumption

$$P \mid\mid C$$



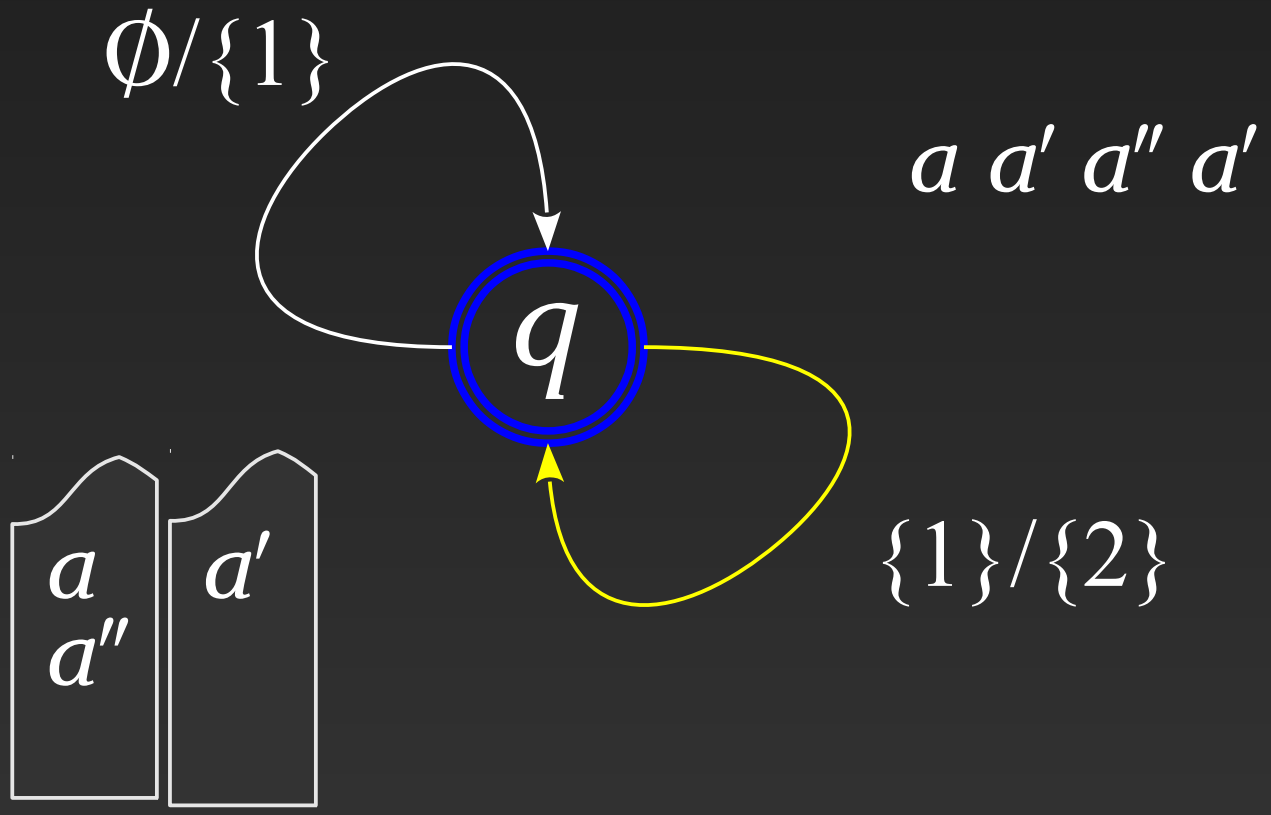
Example: name consumption

$$P \mid\mid C$$



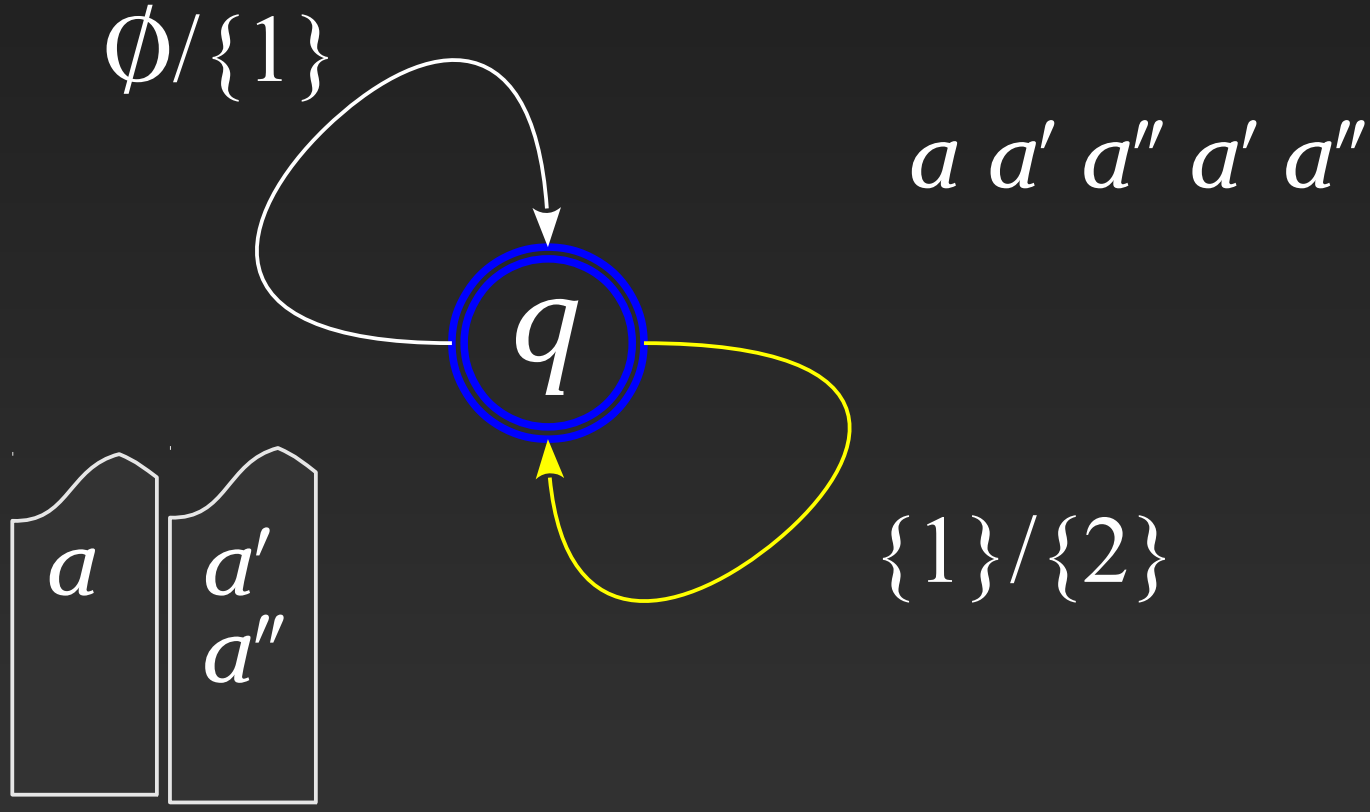
Example: name consumption

$$P \mid\mid C$$



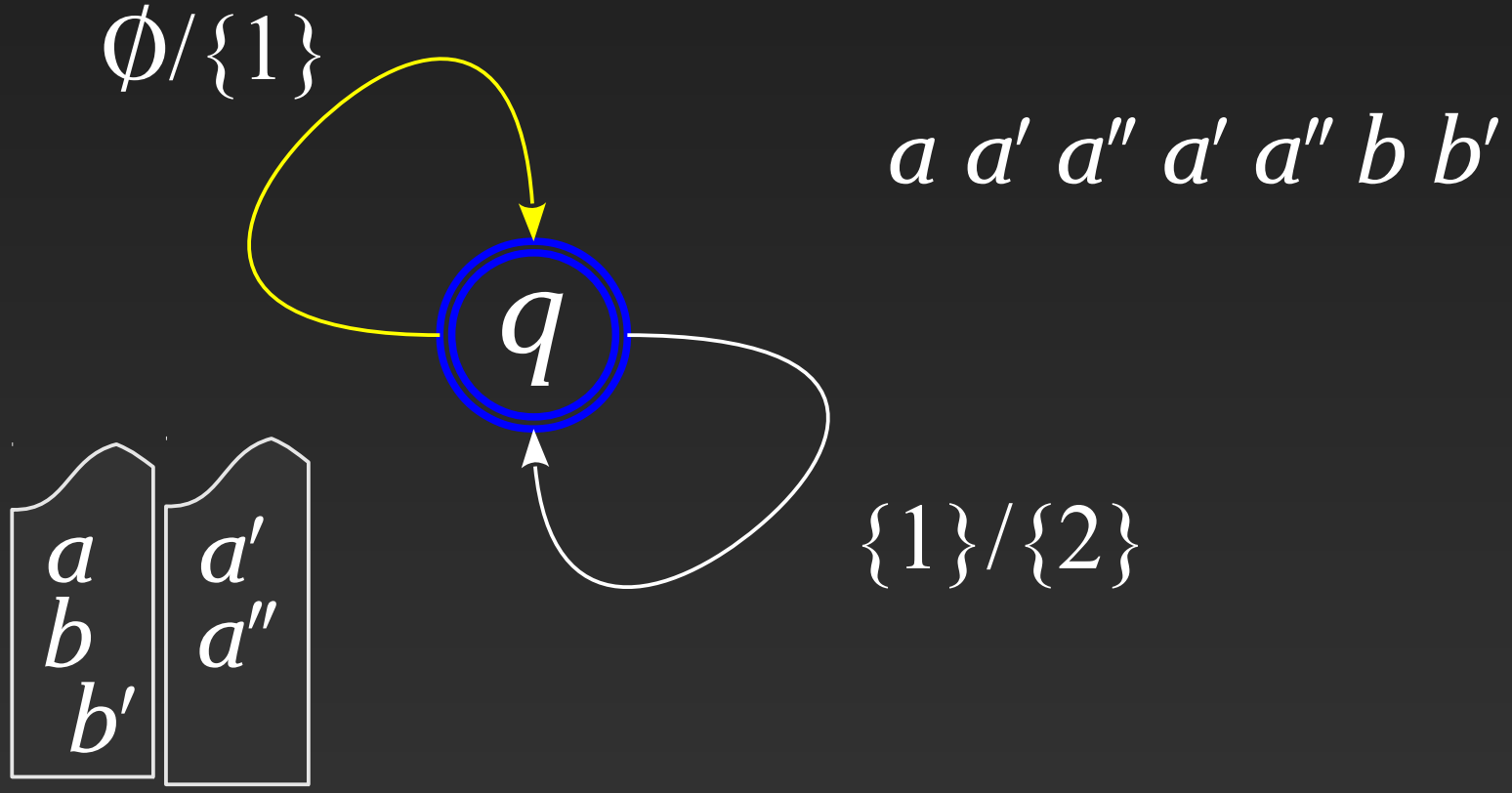
Example: name consumption

$$P \mid\mid C$$



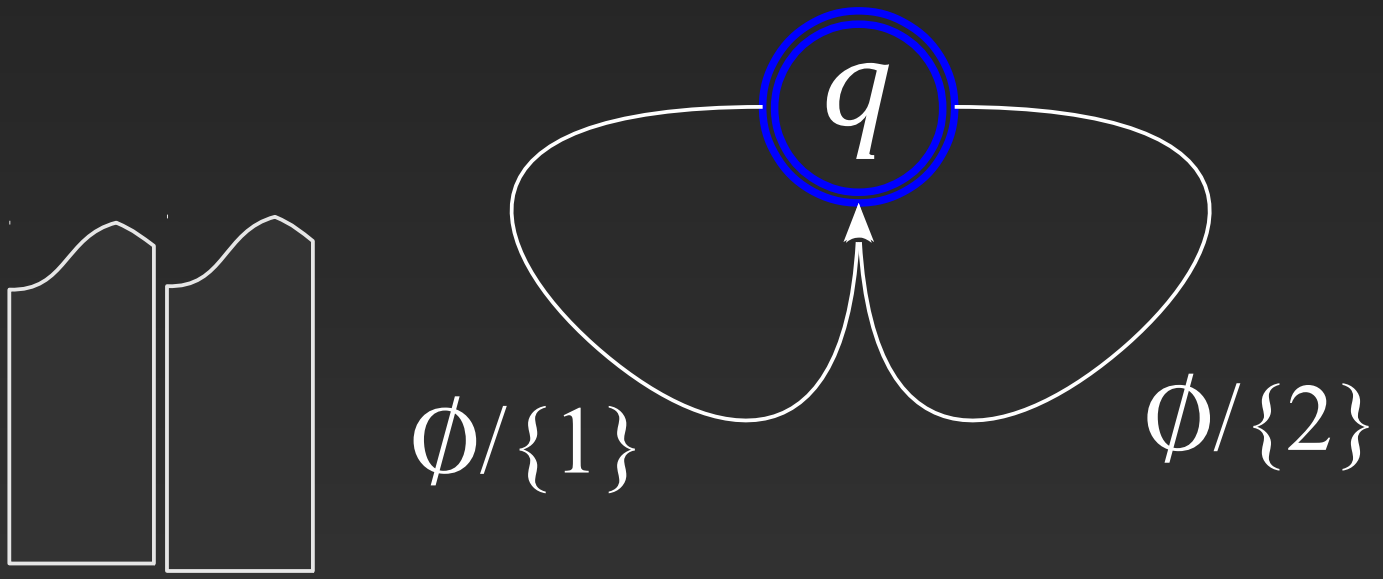
Example: name consumption

$$P \mid\mid C$$



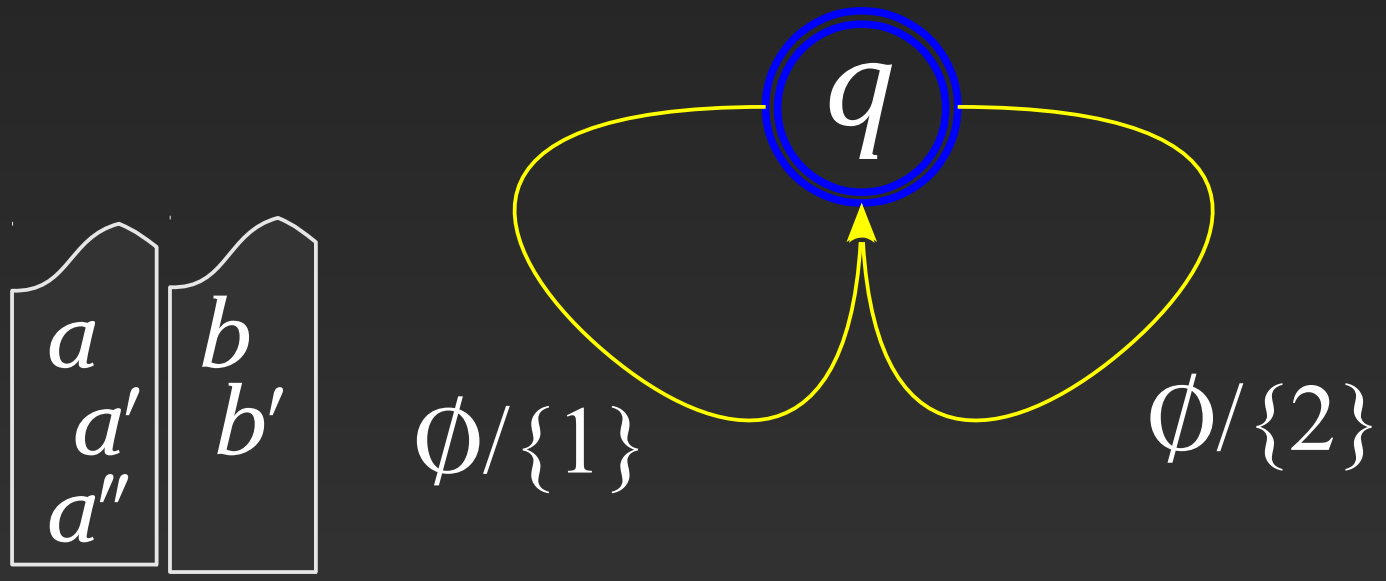
Example: interleaving

$$P \;|*|\; P$$



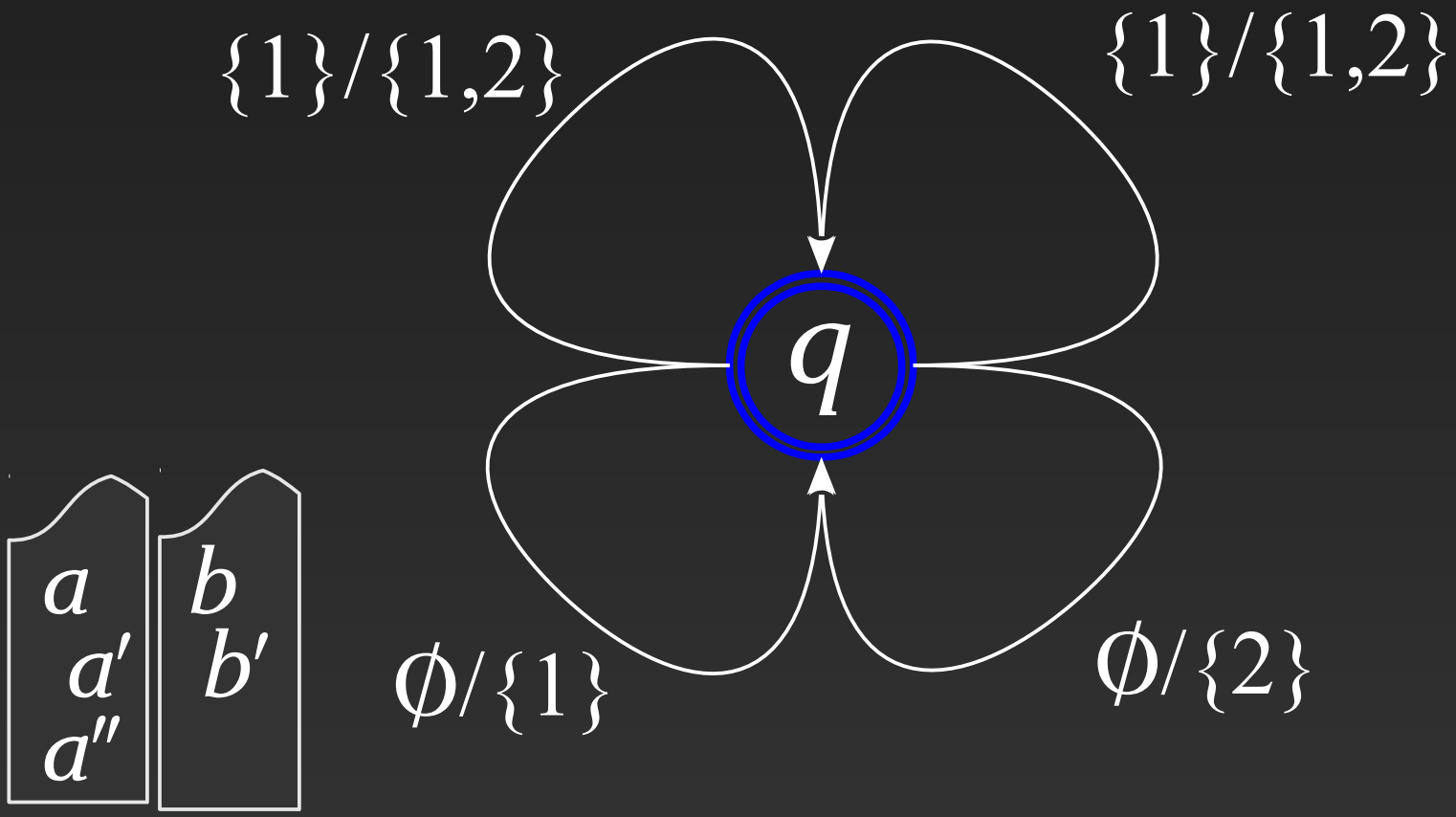
Example: interleaving

$$P \;|*|\; P$$



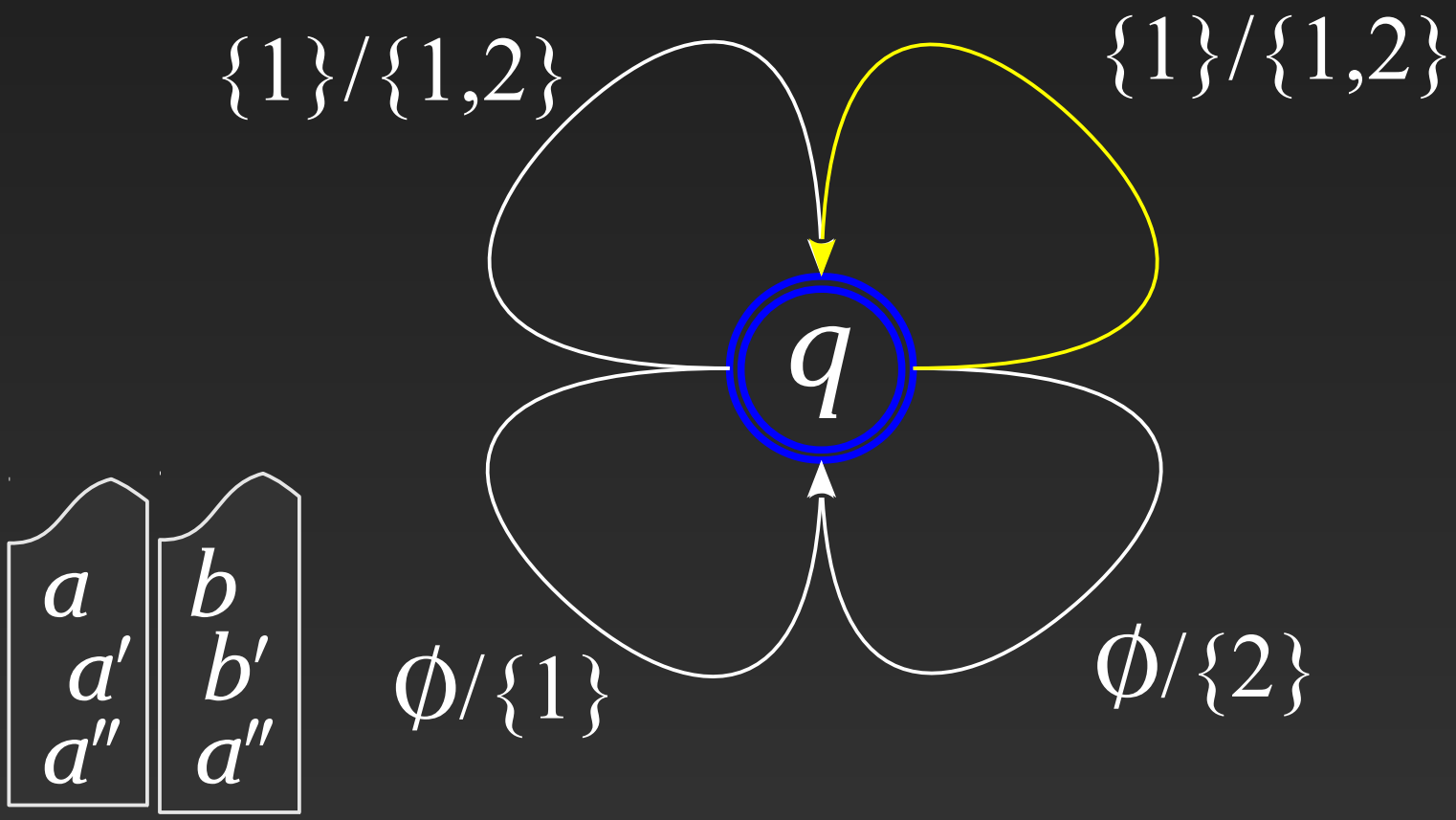
Example: interleaving

$$P \;|*|\; P$$



Example: interleaving

$$P \;|*|\; P$$



HRA properties

- Closed under $\cap$, $\cup$, $\bullet$, * .
- Closed under interleaving
- Not closed under complement & not determinisable

- Universality / equivalence undecidable
- Decidable non-emptiness – but huge complexity:
  - non-primitive recursive (~ Petri nets with transfers)
  - if we ban resets: EXPSPACE-complete

Investigations in FRAs

Global freshness: from oracles to histories

- History Register Automata

[Grigore & T. '13]

Context-freeness: Pushdown FRA

[Cheng & Kaminski '98; Segoufin '06]
[Murawski & T. '12]

- Reachability EXPTIME-complete
- Global reachability via “saturation”

[Murawski, Ramsay & T. '14]

Bisimilarity for FRA (complexity)

- PSPACE-c (depending on modes: NP? → PSPACE → EXPTIME)
- approach uses permutation group theory

[Murawski, Ramsay & T.' 15]

Beyond "regular" languages

Let $\Sigma = \{a_1, a_2, \ldots, a_n, \ldots\}$ be an infinite alphabet of names

can only be compared for equality

- $L_0 = \{ abc \in \Sigma^3 \mid a \neq b, c \in \{a, b\} \}$
- $L_1 = \{ a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i < n.\ a_i \neq a_{i+1} \}$
- $L_2 = \{ a_1 a_2 \ldots a_n \in \Sigma^* \mid \exists i \neq j.\ a_i = a_j \}$
- $L_{\text{new}} = \{ a_1 a_2 \ldots a_n \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \}$
- $L_4 = \{ a_1 \ldots a_n b_1 \ldots b_m \in \Sigma^* \mid \forall i \neq j, i' \neq j'.\ a_i \neq a_j, b_{i'} \neq b_{j'} \}$
- $L_5 = \{ a_1 a_2 \ldots a_n a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \}$

[Cheng & Kaminski '98, Segoufin '06, Murawski & T. '12]

Fresh Pushdown Register Automata (FPDRA)

[Cheng & Kaminski '98, Segoufin '06, Murawski & T. '12]

Fresh Pushdown Register Automata (FPDRA)

Let $\Sigma = \{a_1, a_2, \ldots, a_n, \ldots\}$ be an infinite alphabet of names

finitely many (say R) registers

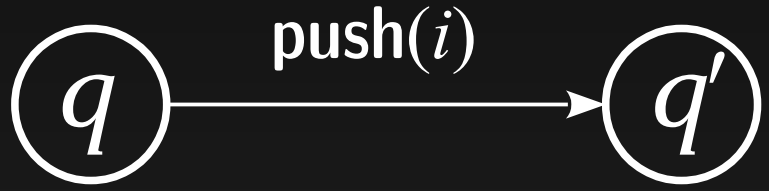
pushdown stack

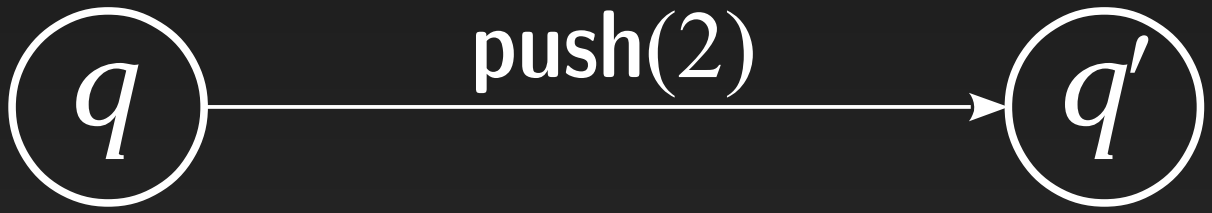
registers & stack store names

Label λ of the form:

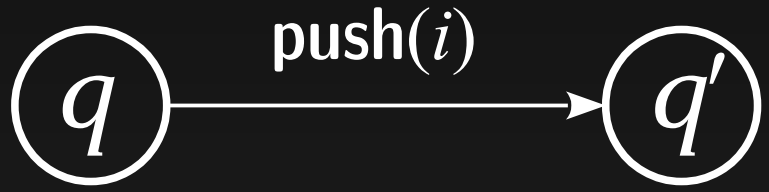
- **read**(i), **fresh**(i), **new**(i), $i \in \{1,\ldots,R\}$
- **push**(i), $i \in \{1,\ldots,R\}$
- **pop**(i), $i \in \{1,\ldots,R\}$
- **pop-fresh**

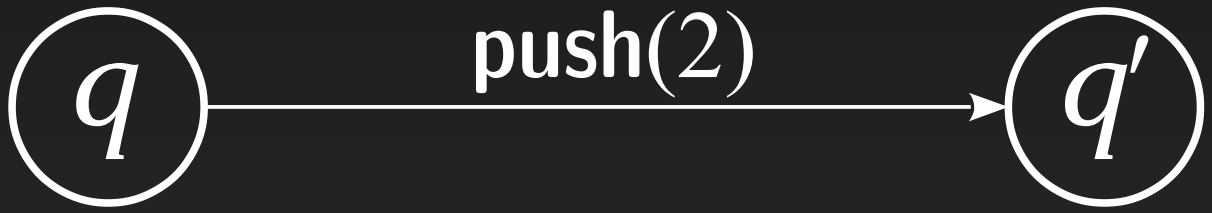
Transitions:

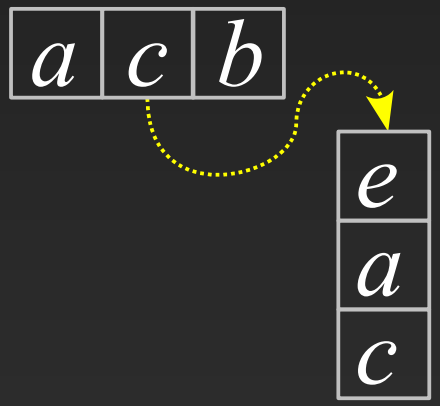




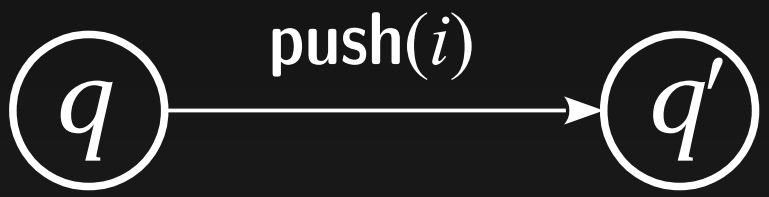
Transitions:

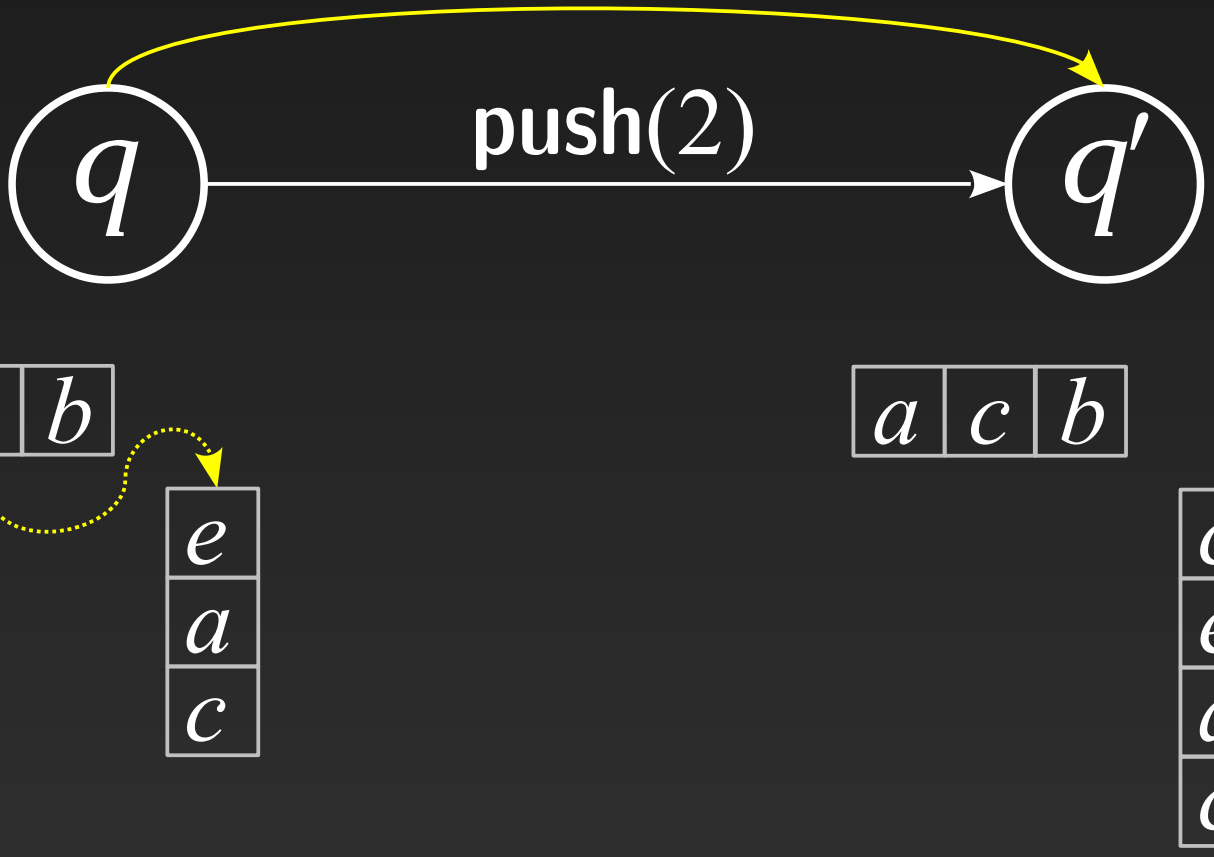




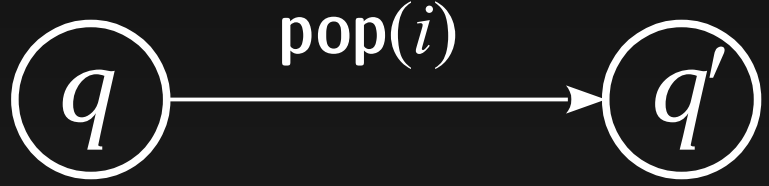


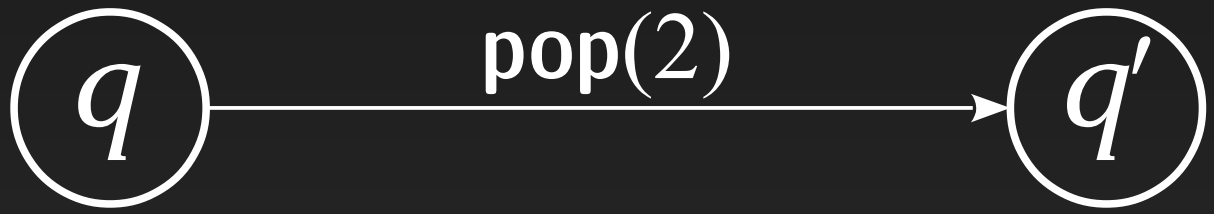
Transitions:



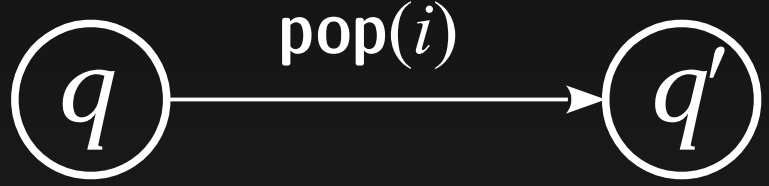


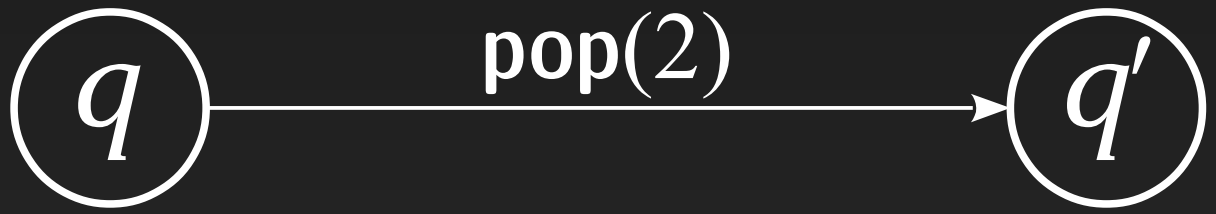
Transitions:

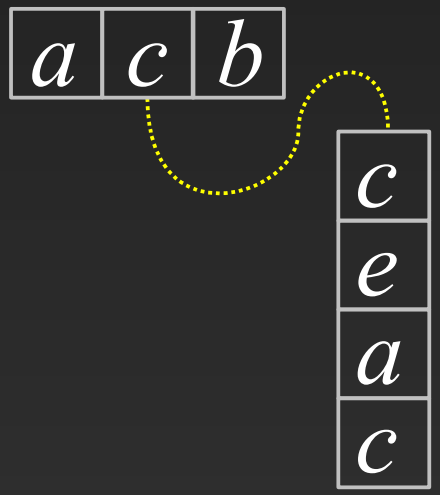




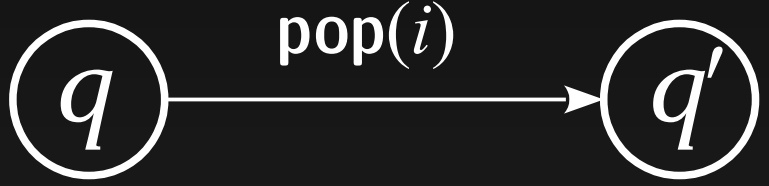
Transitions:

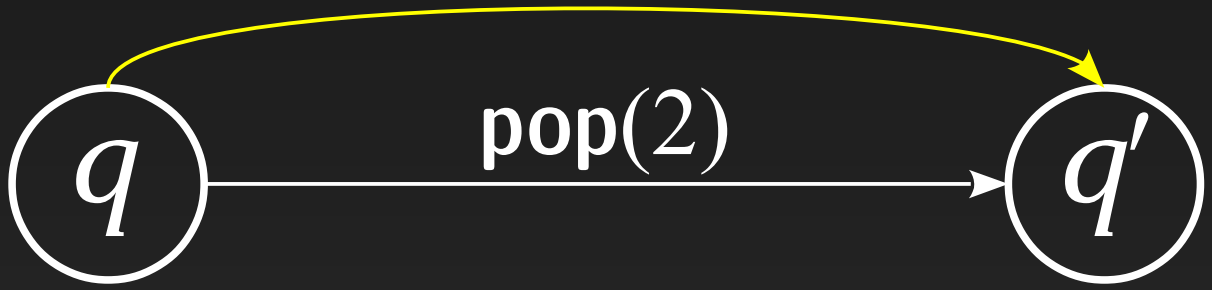


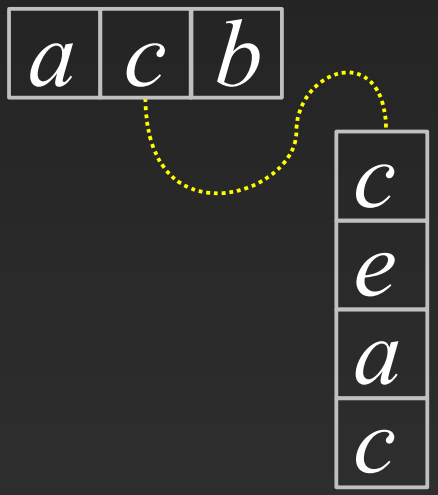


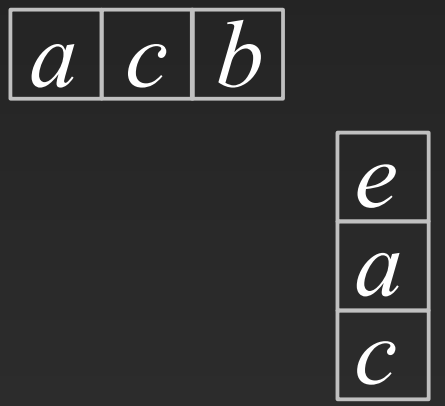


Transitions:

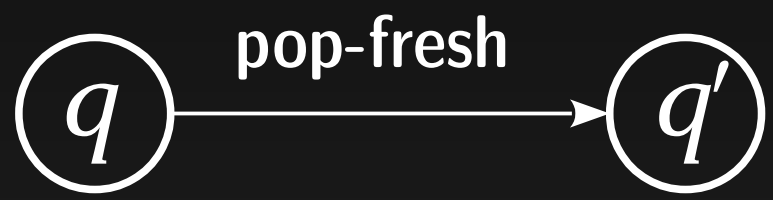


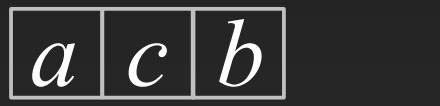




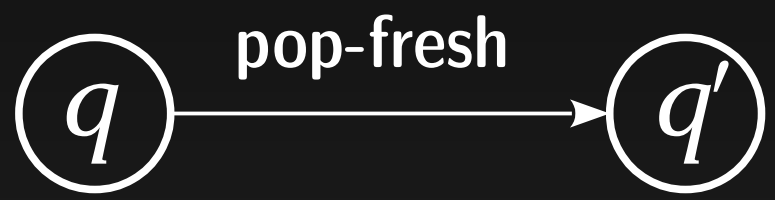


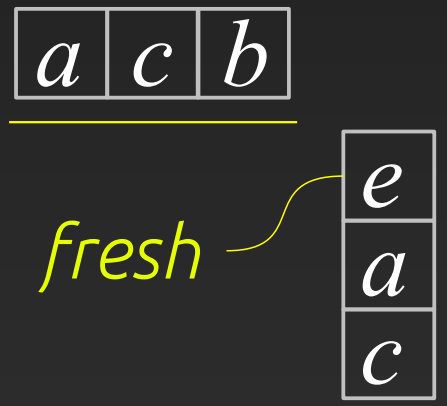
Transitions:



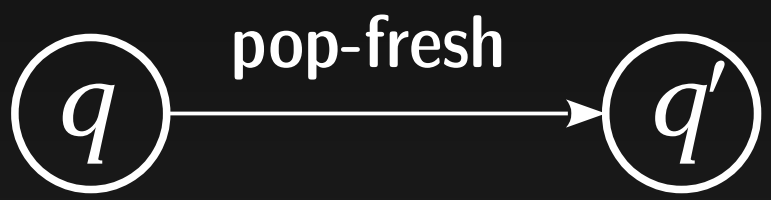


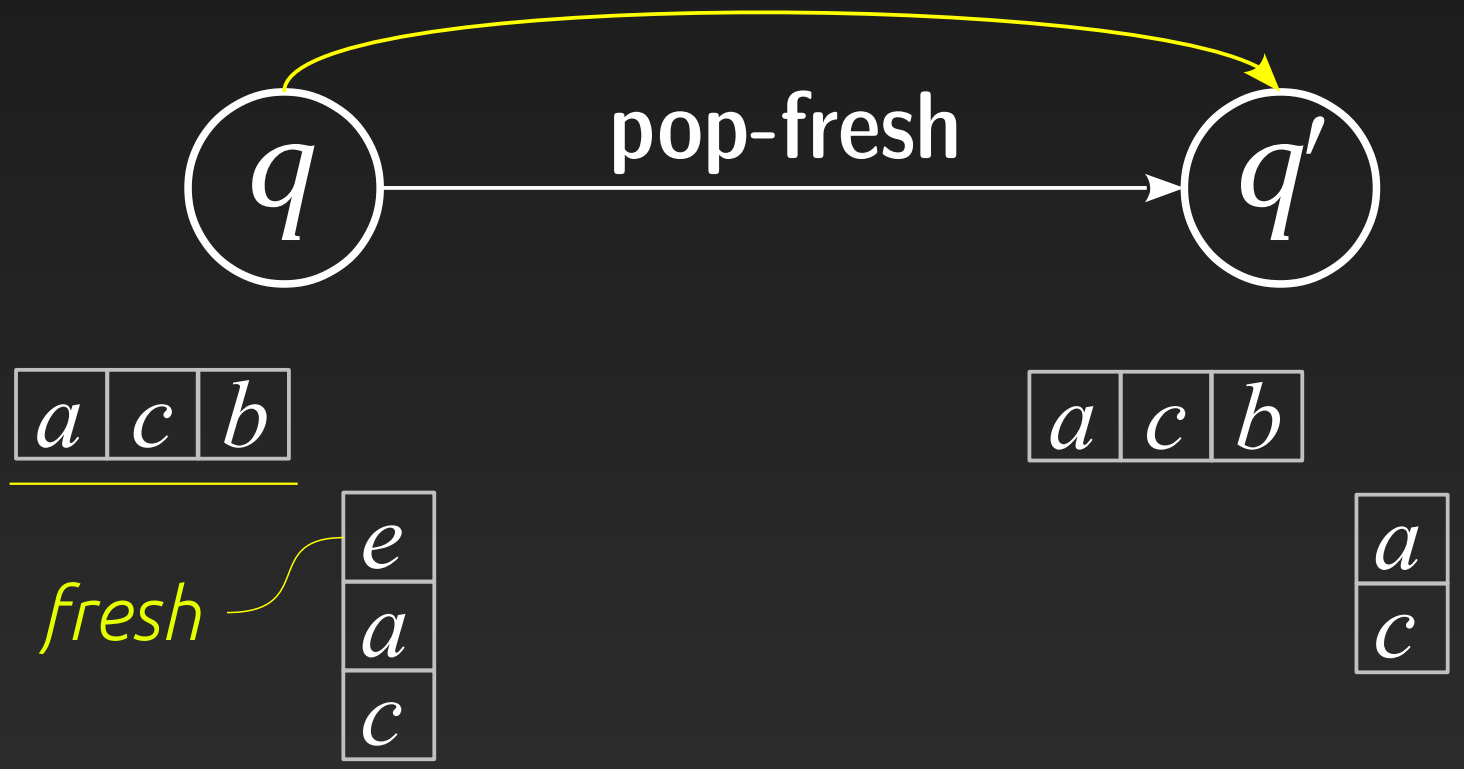
Transitions:





Transitions:

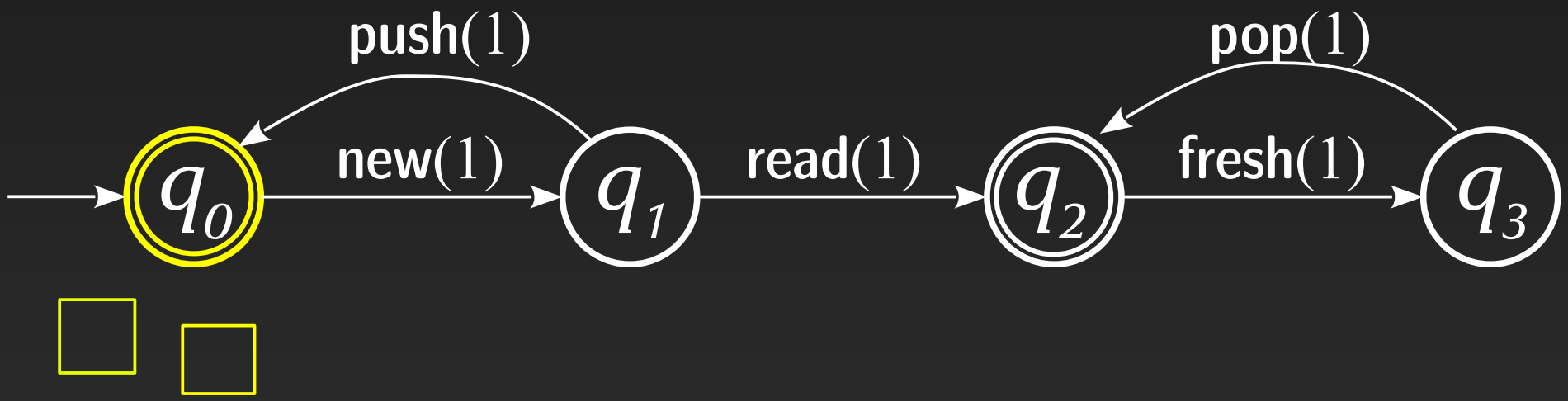




Example

$$L_5 = \{\, a_1 a_2 \ldots a_n \, a_n a_{n-1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

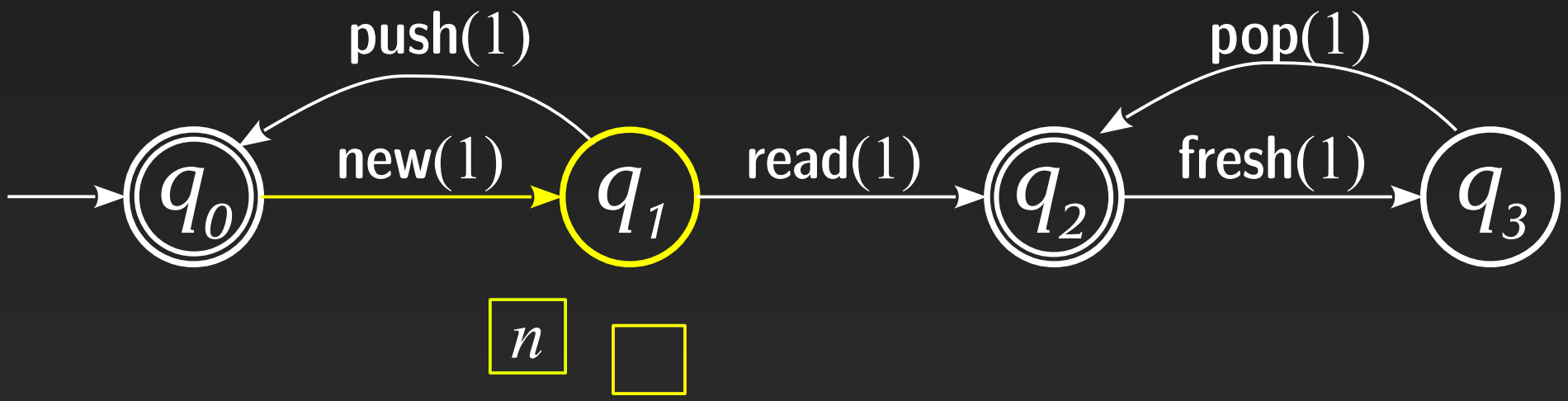
(all even-length palindromes of pairwise distinct names)



Example

$$L_5 = \{\, a_1 a_2 \ldots a_n \, a_n a_{n-1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

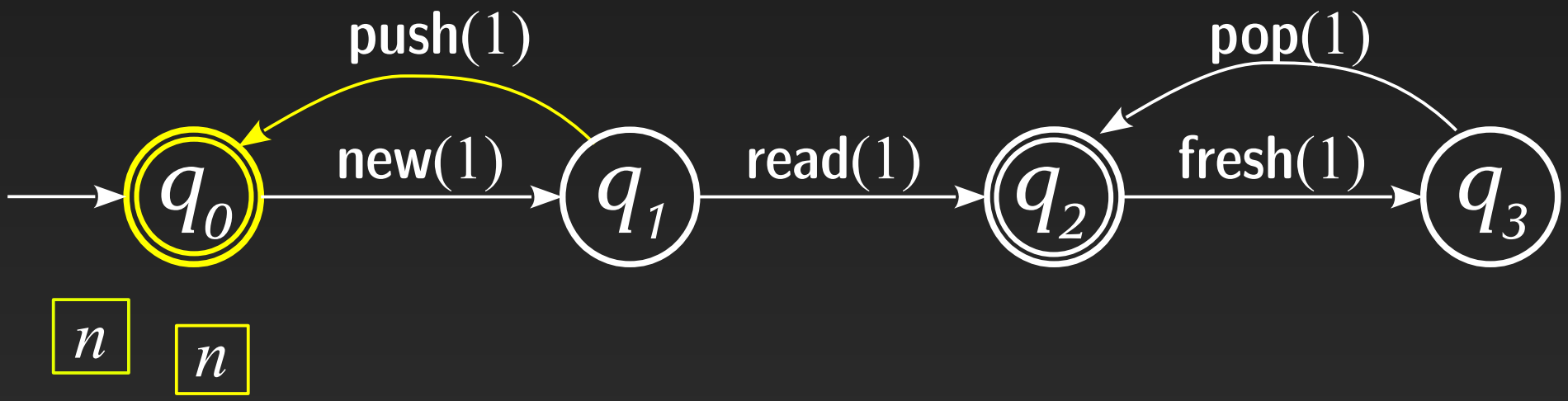


n

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n \, a_n a_{n-1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

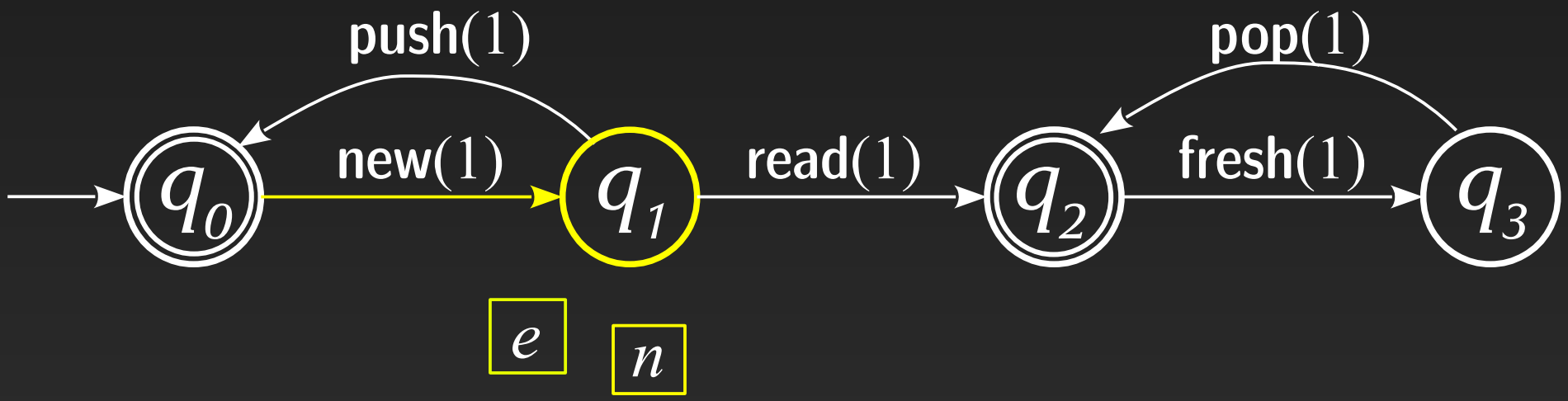


n

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n\, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \;\mid\; \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

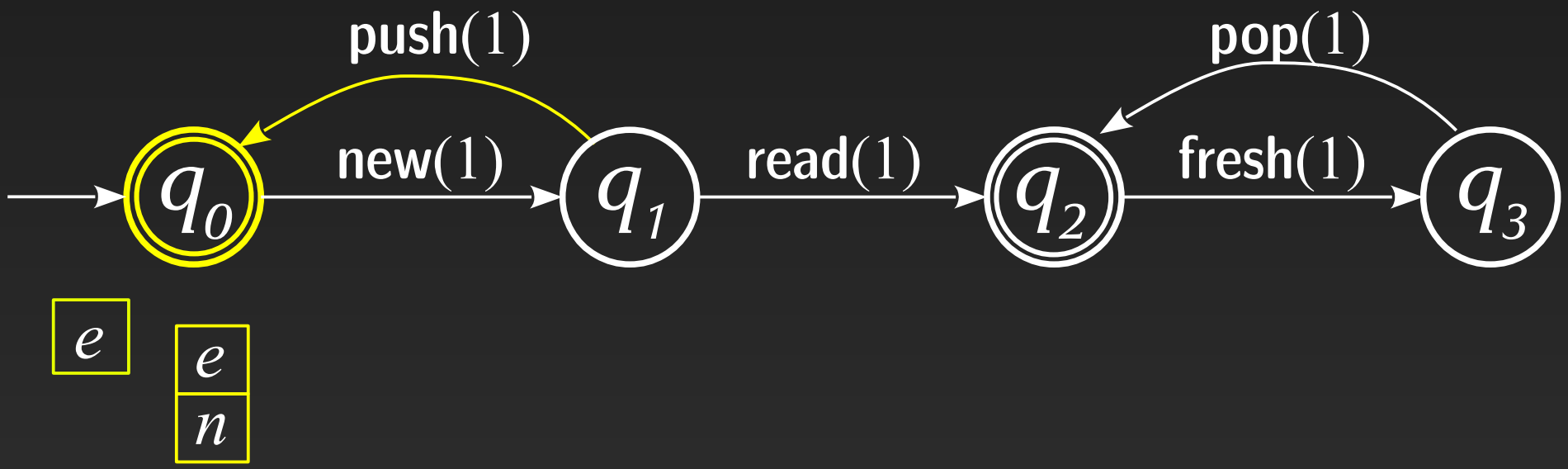


$n\,e$

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n\, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

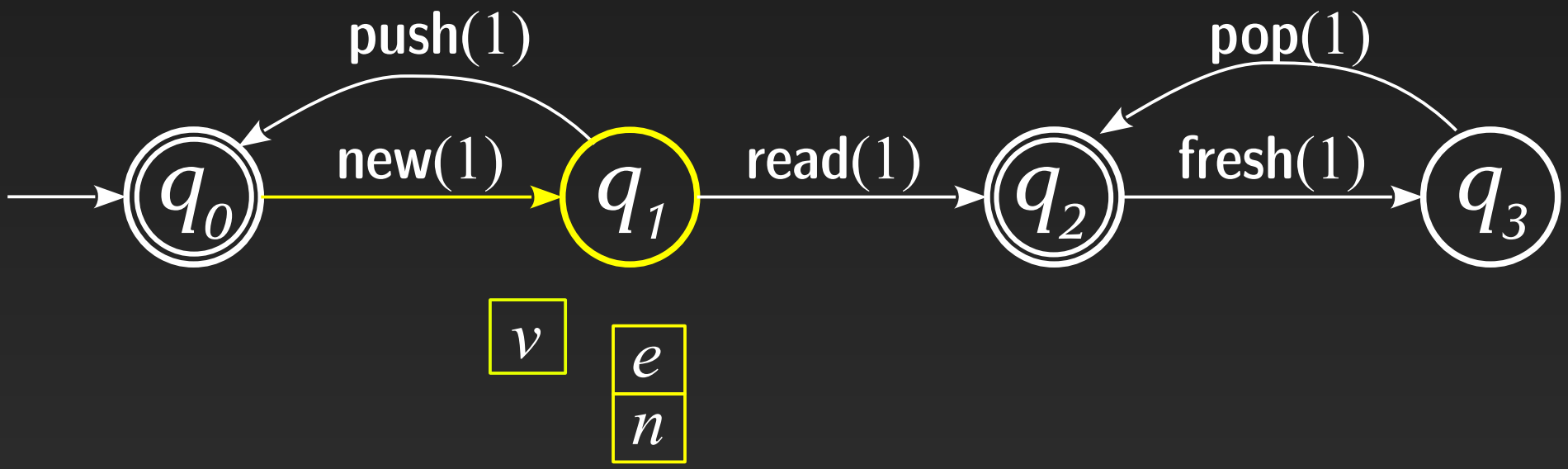


$n\,e$

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n\, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \;\mid\; \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

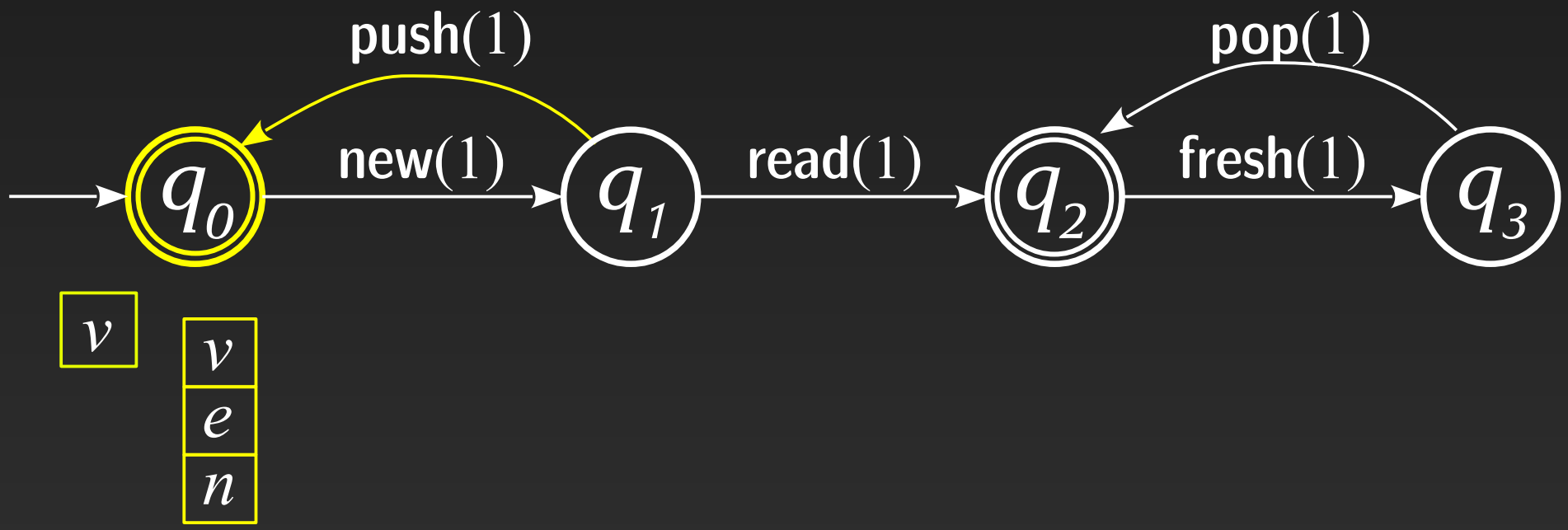


n e v

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n\, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \;\mid\; \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

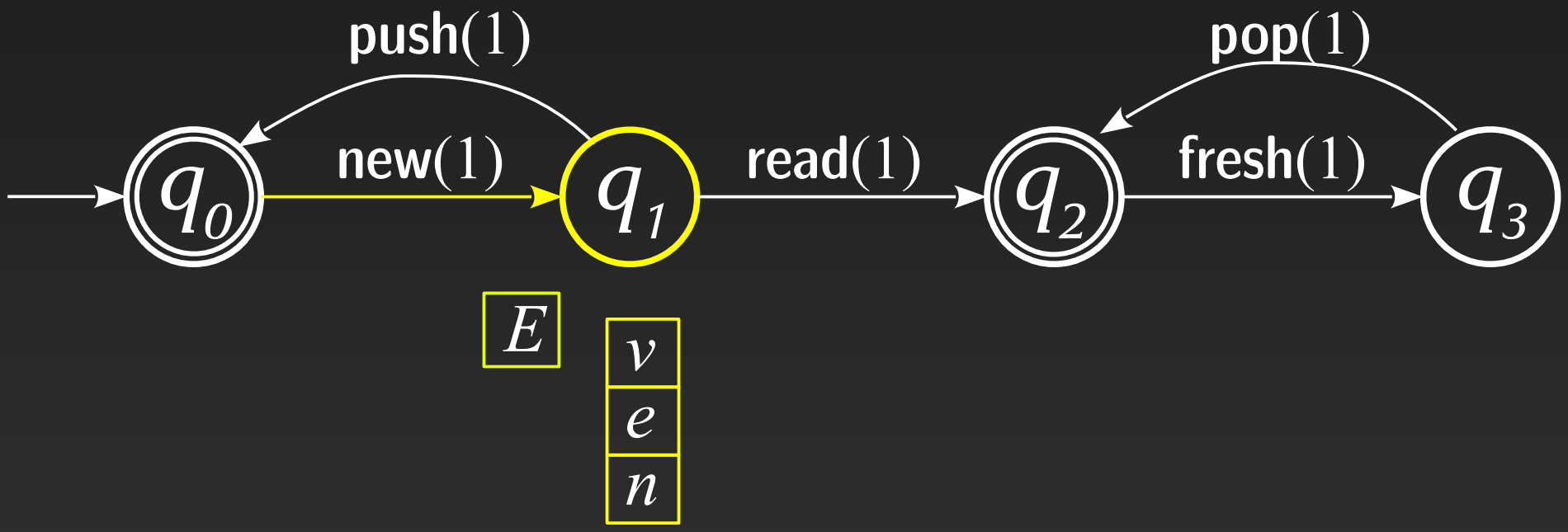


n e v

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n\, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \;\mid\; \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

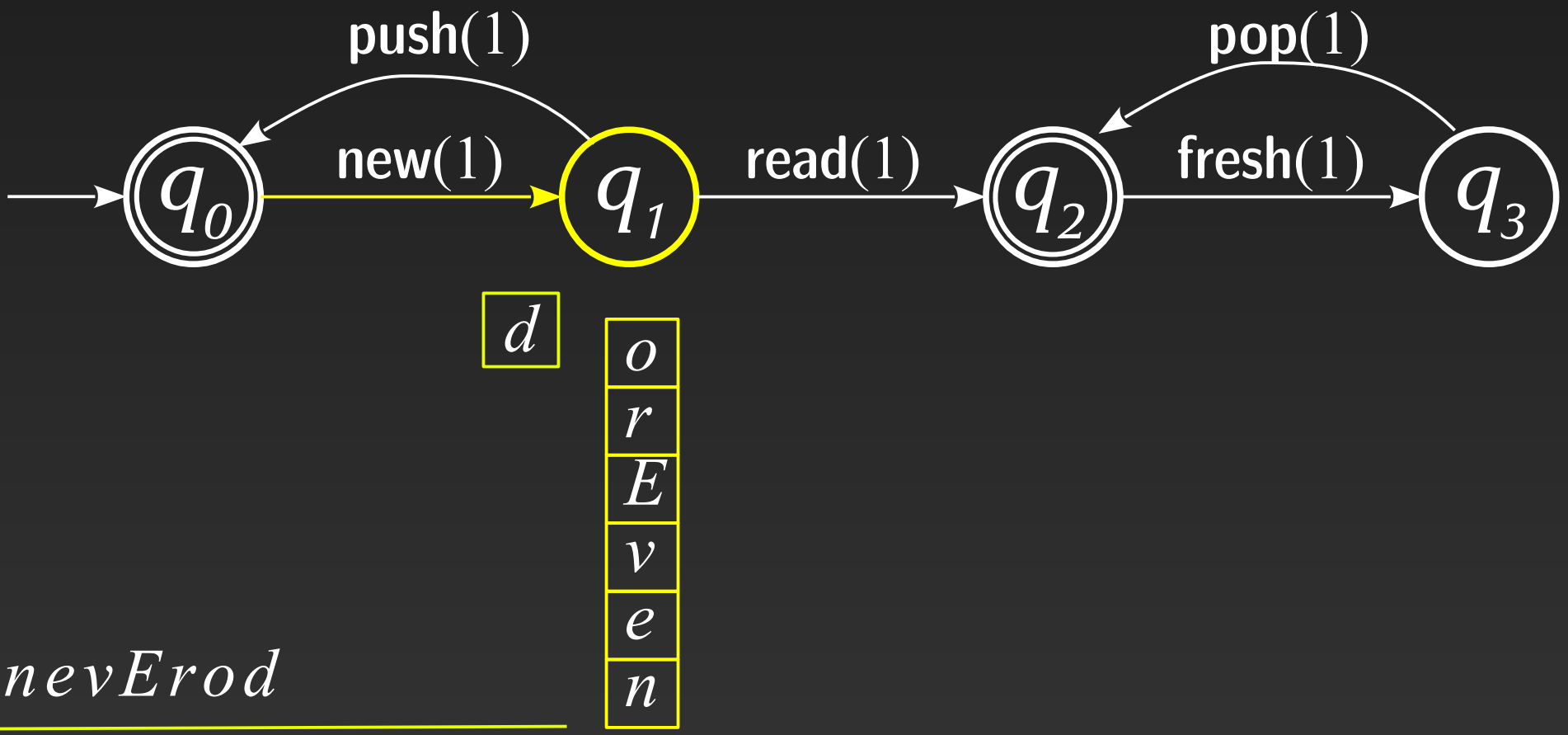


$nevE$

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n \, a_n a_{n-1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

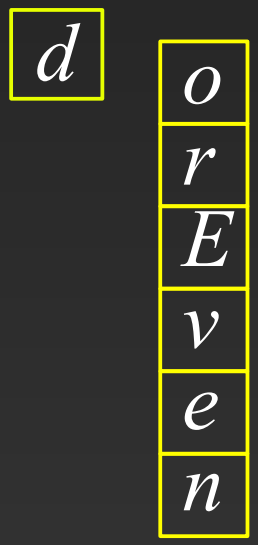
(all even-length palindromes of pairwise distinct names)



Example

$$L_5 = \{\, a_1 a_2 \ldots a_n\, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \;\mid\; \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

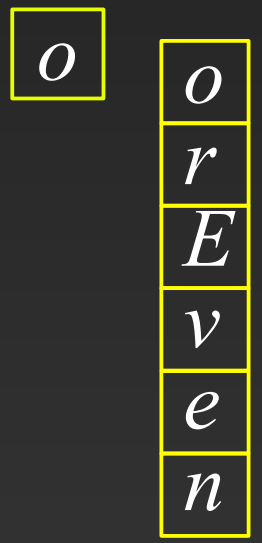


nevErodd

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n\, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

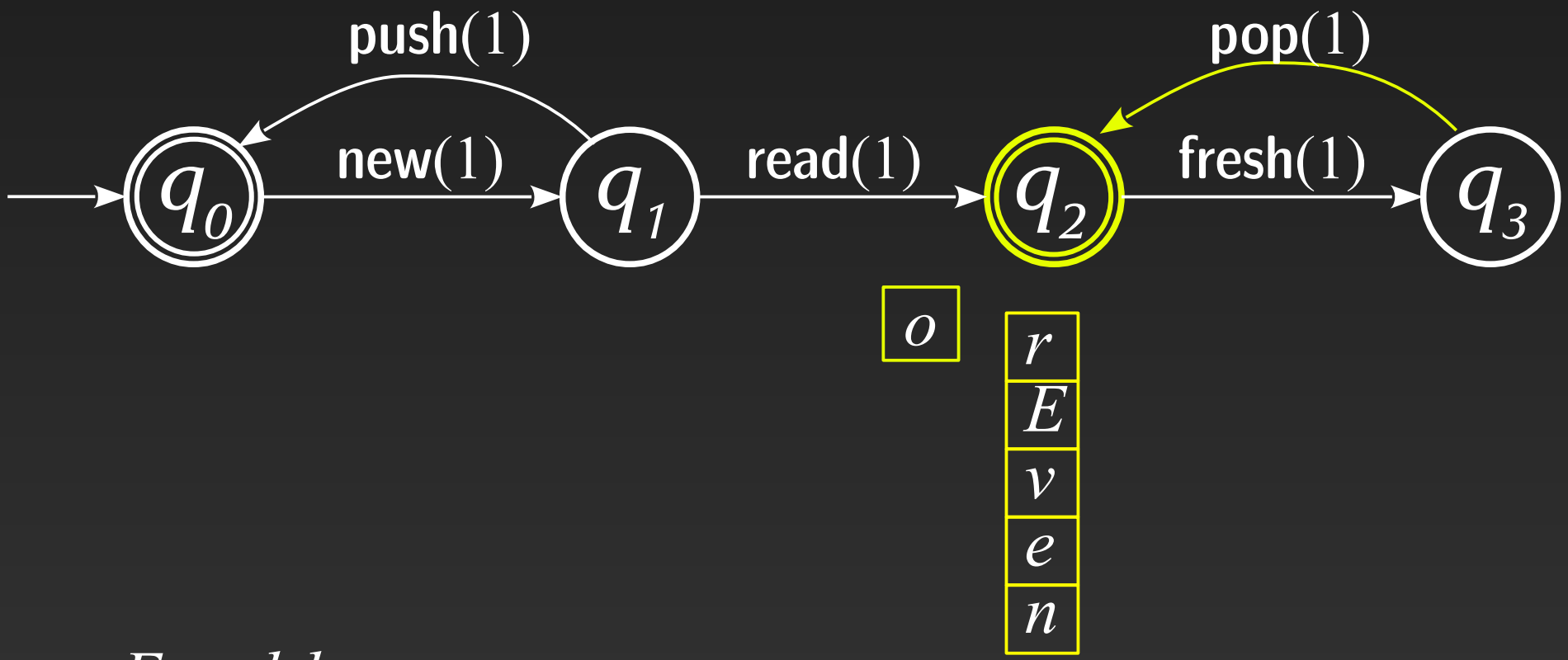


nevEroddo

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n \, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

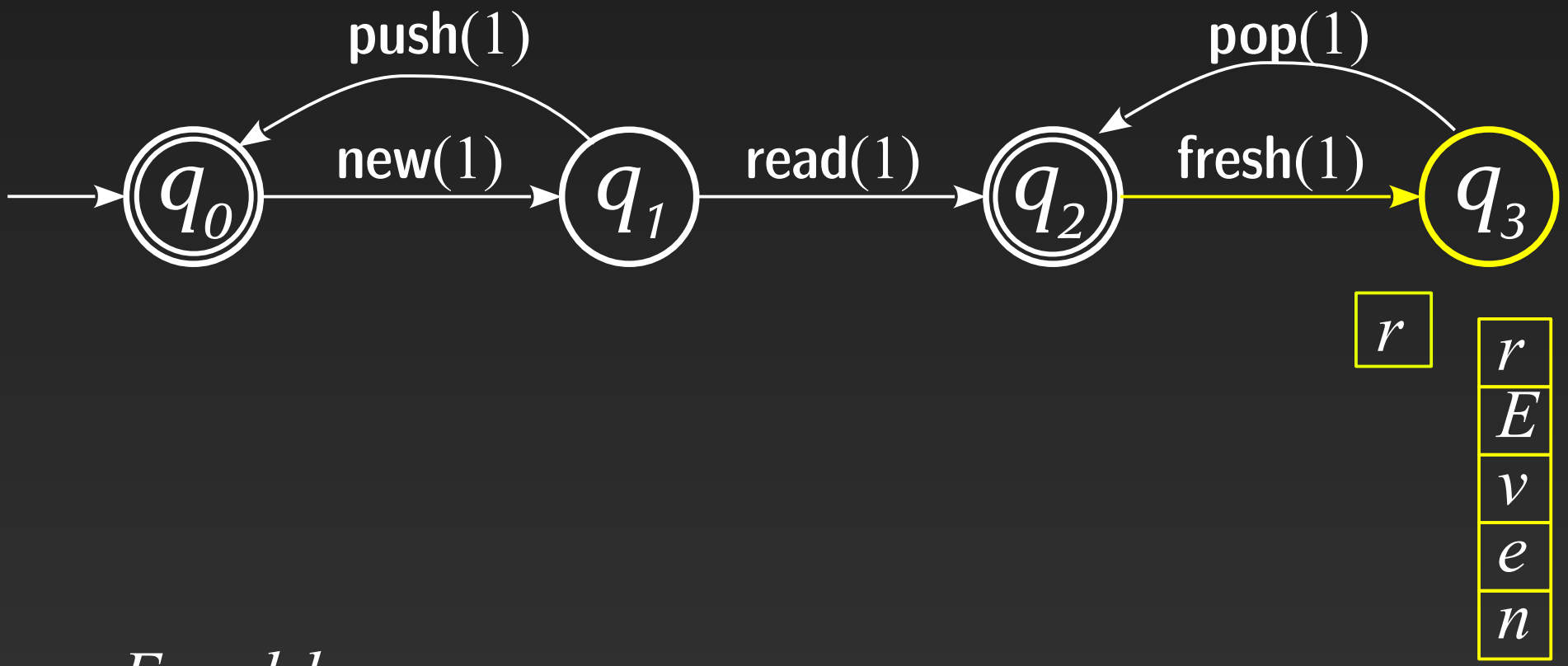


nevEroddo

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n\, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

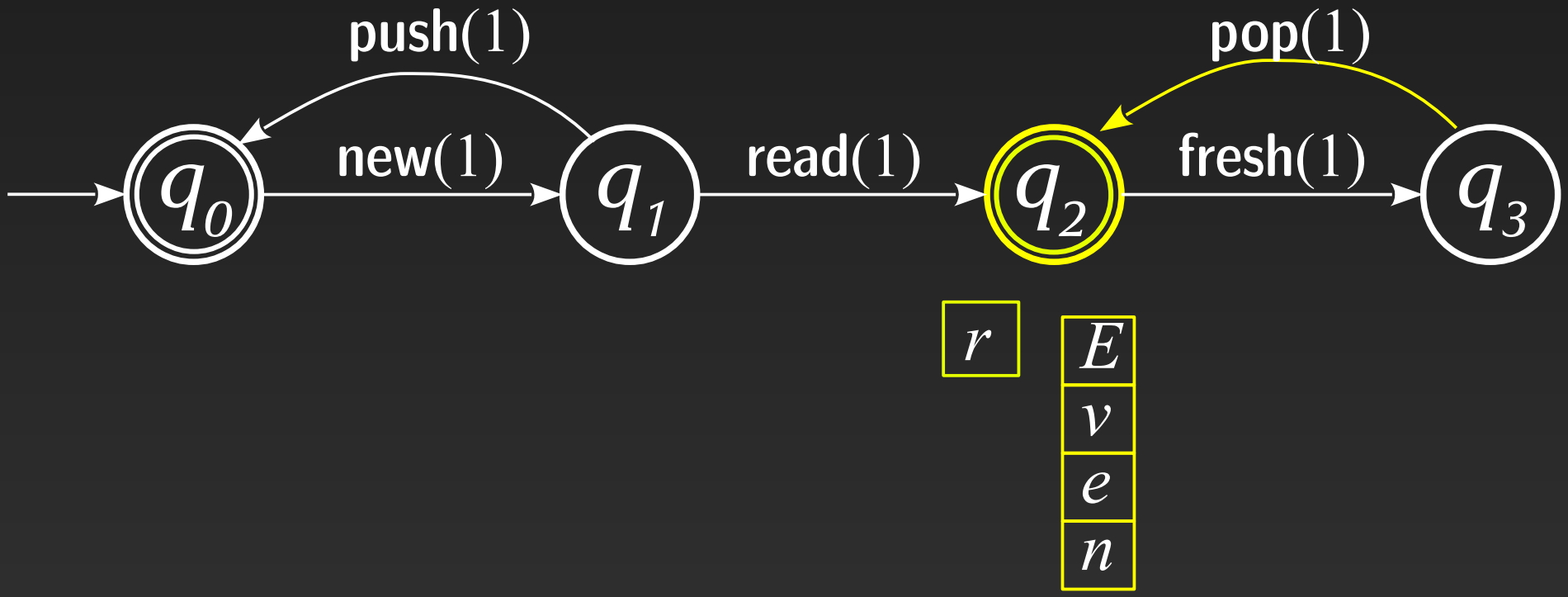


nevEroddor

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n\, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)

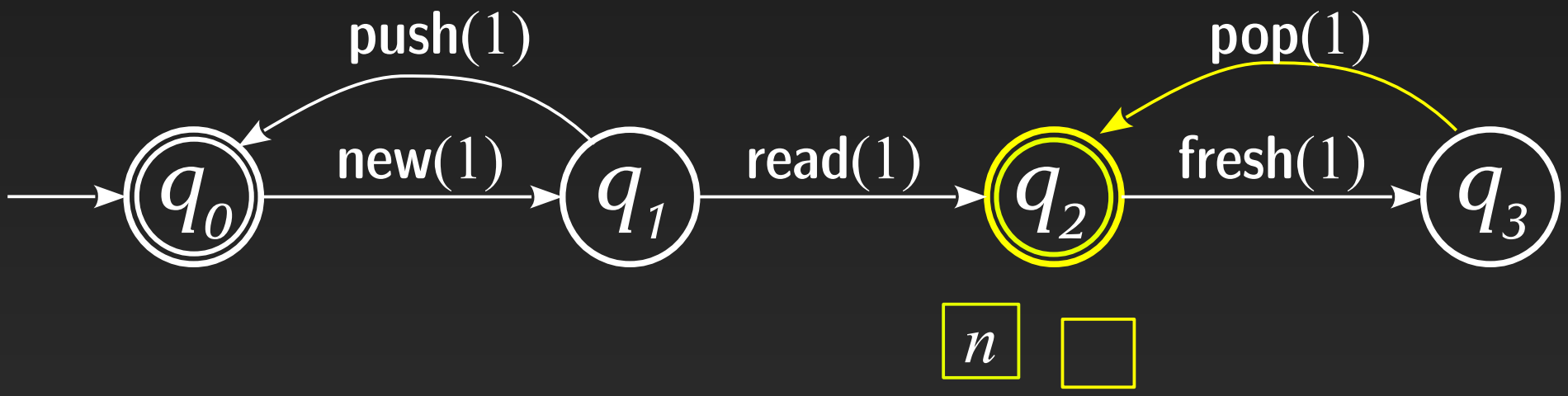


nevEroddor

Example

$$L_5 = \{\, a_1 a_2 \ldots a_n\, a_n a_{n\text{-}1} \ldots a_1 \in \Sigma^* \mid \forall i \neq j.\ a_i \neq a_j \,\}$$

(all even-length palindromes of pairwise distinct names)



nevEroddorEven

[Murawski, Ramsay & T. '14]

Non-emptiness for PDRAs

Lemma: If some word of size n is accepted by A then one of the same size with at most $3R$ names is also accepted by A

Theorem: PRDA Non-emptiness is EXPTIME-complete

EXPTIME solvability:

- By previous Lemma, $3R$ names suffice: $\Sigma' = \{a_1, \ldots, a_{3R}\}$
- so, registers can be encoded inside states: $Q' = Q \times 3R^R$
- and we reduce to PDA reachability (PTIME)

EXPTIME hardness, even with “full” registers:

- Reduction from TMs with stack – more difficult

[Murawski & T. '12, Murawski, Ramsay & T. '14]

Non-emptiness for FPDRAs

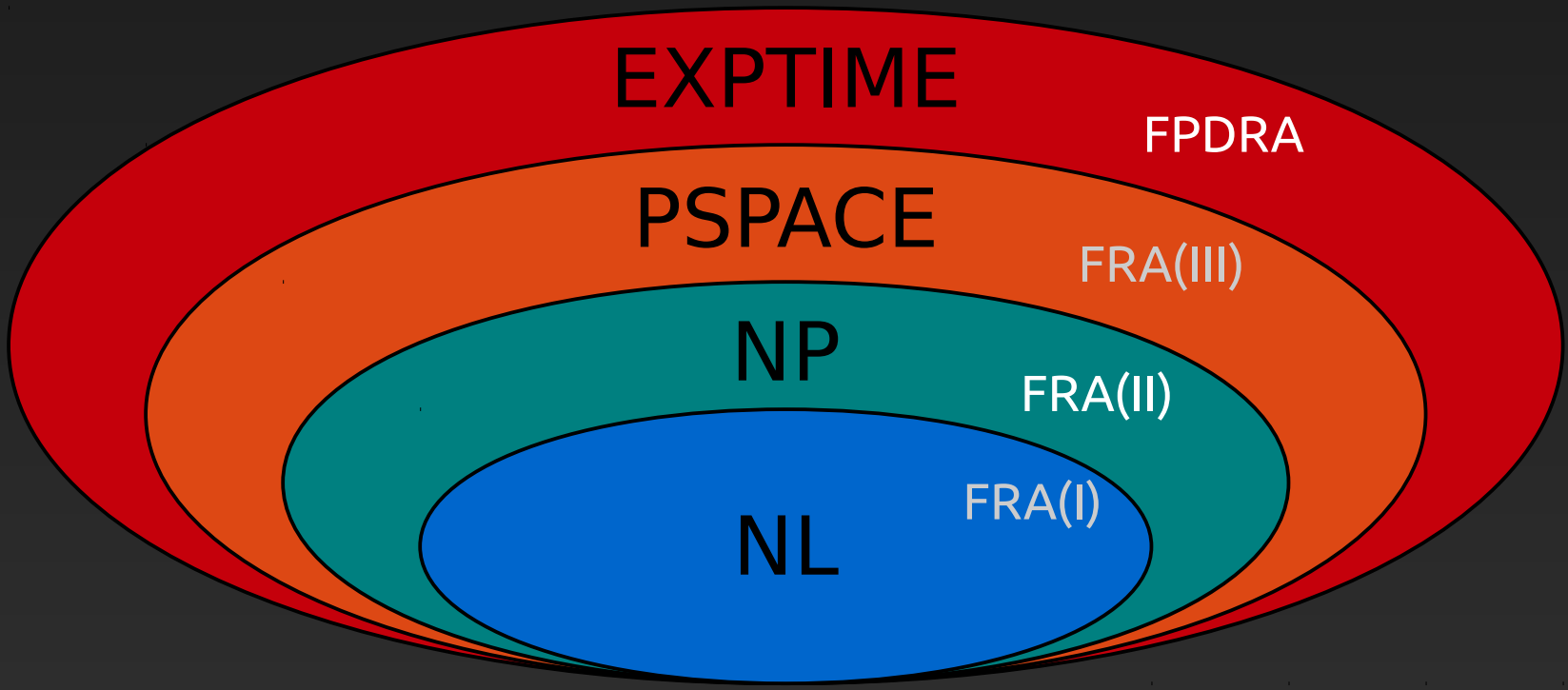
Lemma: Given an FPDRA A with set of states Q we can construct a PDRA A' with states $Q' = Q \times 2^R$ such that $L(A) \neq \emptyset$ iff $L(A') \neq \emptyset$

This is done by *simulating* global fresh:

- by local fresh + tags on registers and stack
- the tags ensure distinctness in every comparison
- the simulation is exponential: $Q' = Q \times 2^R$
  - summing up: $Q'' = Q \times 2^R \times 3R^R$

Theorem: FPRDA Non-emptiness is EXPTIME-complete

Non-emptiness for FRAs



Investigations in FRAs

Global freshness: from oracles to histories

- History Register Automata

[Grigore & T. '13]

Context-freeness: Pushdown FRA

[Cheng & Kaminski '98; Segoufin '06]
[Murawski & T. '12]

- Reachability EXPTIME-complete
- Global reachability via “saturation”

[Murawski, Ramsay & T. '14]

Bisimilarity for FRA (complexity)

- PSPACE-c (depending on modes: NP? → PSPACE → EXPTIME)
- approach uses permutation group theory

[Murawski, Ramsay & T.' 15]

Related work (there is a lot!)

Some examples:

- A wealth of results on RAs (including powerful extensions), stemming from XML model checking

[Segoufin '06 (survey), Bojanczyk, David, Muscholl, Schwentick & Segoufin '11, Björklund & Schwentick '10]

- A recent research programme on *abstract machines with atoms*

[Bojanczyk, Klin, Lasota & Torunczyk'13, Bojanczyk, Klin & Lasota '14]

Concluding

thanks

Fresh-Register Automata:

- Class of automata over infinite alphabets
  “natural” for computation with names/resources
- new landscape of algorithms and results
- applications in verification

Things to do:

- algorithm implementations (an FRA toolkit!)
- logics and use in model checking
- theory: some open cases, automata learning