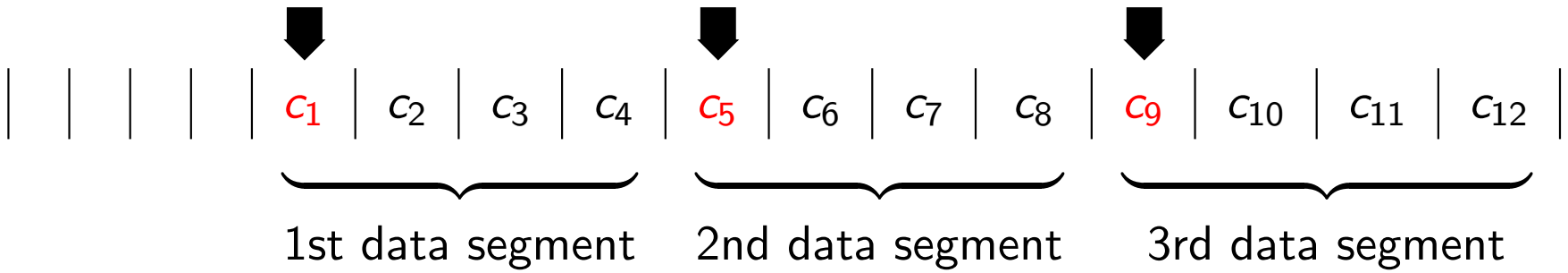
Codes Correcting Under- and Over-Shift Errors in Racetrack Memories

Presented by: Van Khu Vu

Joint work with Yeow Meng Chee, Han Mao Kiah, Alexander Vardy and Eitan Yaakobi

11th March 2019

Racetrack Memory

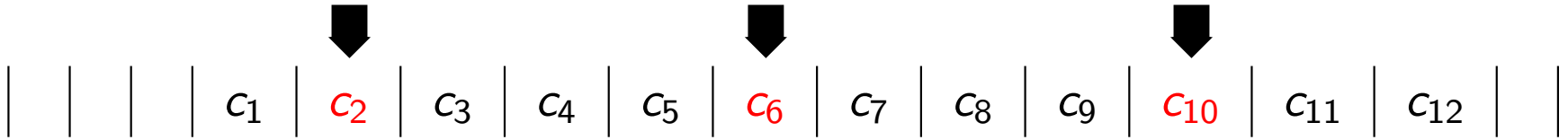


Head 1 : c_1

Head 2 : c_5

Head 3 : c_9

Racetrack Memory

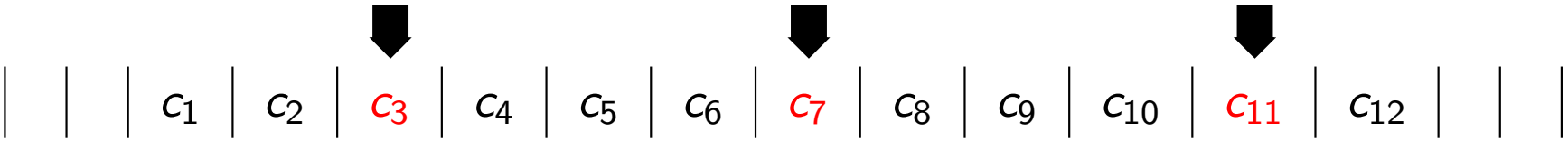


Head 1 : c_1, c_2

Head 2 : c_5, c_6

Head 3 : c_9, c_{10}

Racetrack Memory

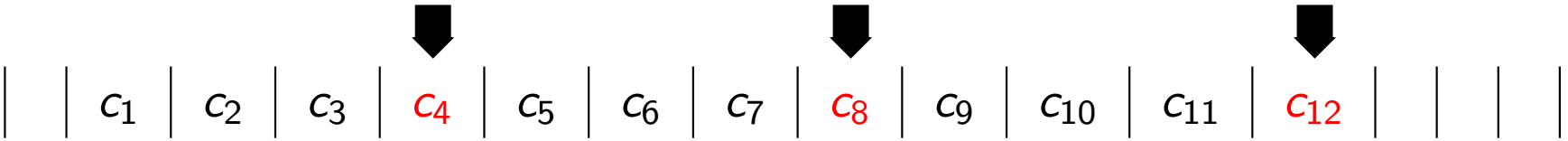


Head 1 : c_1, c_2, c_3

Head 2 : c_5, c_6, c_7

Head 3 : c_9, c_{10}, c_{11}

Racetrack Memory



Head 1 : c_1, c_2, c_3, c_4

Head 2 : c_5, c_6, c_7, c_8

Head 3 : $c_9, c_{10}, c_{11}, c_{12}$

Racetrack Memory

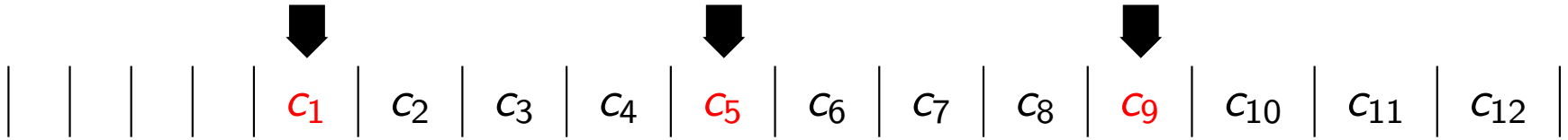
Racetrack Memory: n domains, m heads where $n = m \cdot \ell$.
Stored word: $\boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \mathbb{F}_2^n$.
Output from m heads:

$$\begin{pmatrix} \boldsymbol{c}^1 \\ \boldsymbol{c}^2 \\ \vdots \\ \boldsymbol{c}^m \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,\ell} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,\ell} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,\ell} \end{pmatrix}$$

where $c_{i,j} = c_{(i-1)\cdot\ell+j}$.
Output as q-ary word: $\boldsymbol{u} = (u_1, u_2, \ldots, u_\ell) \in \mathbb{F}_q^\ell$ where $q = 2^m$ and $u_i = (c_{1,i}, c_{2,i}, \ldots, c_{m,i})$.

NANYANG TECHNOLOGICAL UNIVERSITY

Under-shift Error

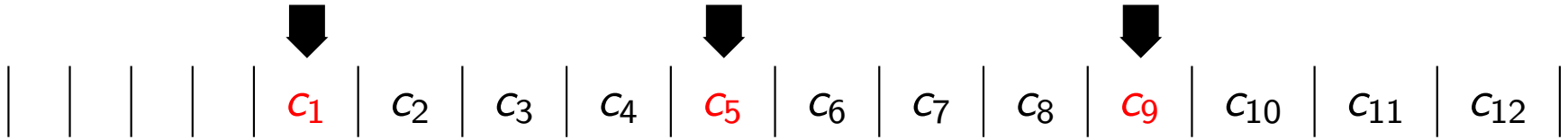


Head 1 : c_1

Head 2 : c_5

Head 3 : c_9

Under-shift Error

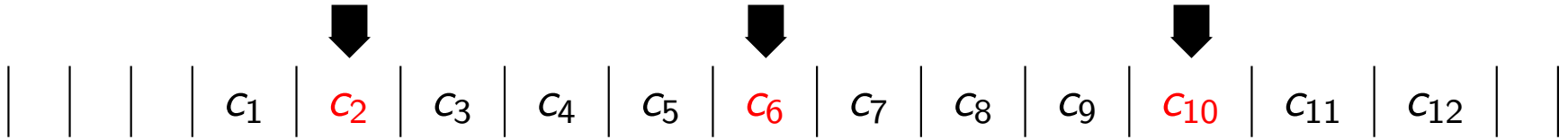


Head 1 : c_1, c_1

Head 2 : c_5, c_5

Head 3 : c_9, c_9

Under-shift Error

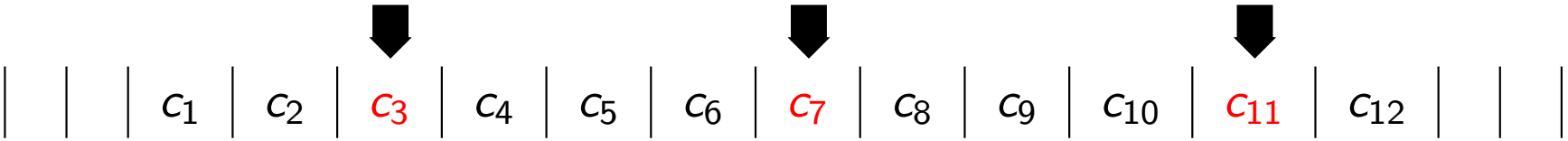


Head 1 : c_1, c_1, c_2

Head 2 : c_5, c_5, c_6

Head 3 : c_9, c_9, c_{10}

Under-shift Error

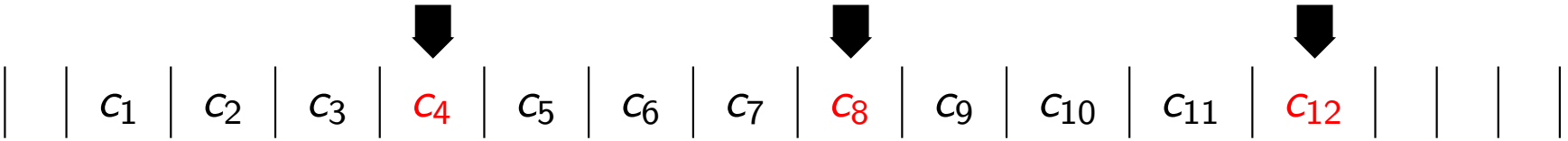


Head 1 : c_1, c_1, c_2, c_3

Head 2 : c_5, c_5, c_6, c_7

Head 3 : c_9, c_9, c_{10}, c_{11}

Under-shift Error



Head 1 : c_1, c_1, c_2, c_3, c_4

Head 2 : c_5, c_5, c_6, c_7, c_8

Head 3 : $c_9, c_9, c_{10}, c_{11}, c_{12}$

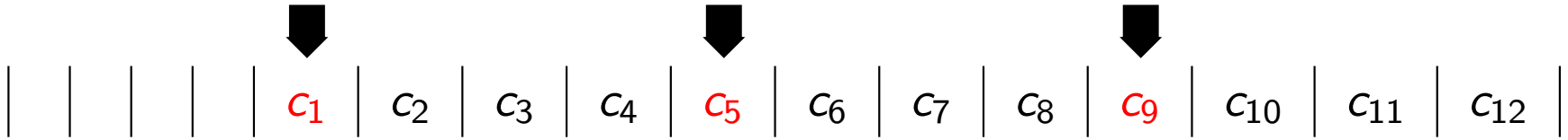
Sticky-Insertion

Output:

$$\begin{pmatrix} c_1 & c_1 & c_2 & c_3 & c_4 \\ c_5 & c_5 & c_6 & c_7 & c_8 \\ c_9 & c_9 & c_{10} & c_{11} & c_{12} \end{pmatrix} = (u_1, u_1, u_2, u_3, u_4)$$

Model 1: An under-shift error can be modeled as a sticky-insertion.
Question 1: How to construct a q-ary code correcting multiple sticky-insertions?
Answer 1: L. Dolecek and V. Anantharam (2010); H. Mahdavifar and A. Vardy (2017).

Over-shift Error

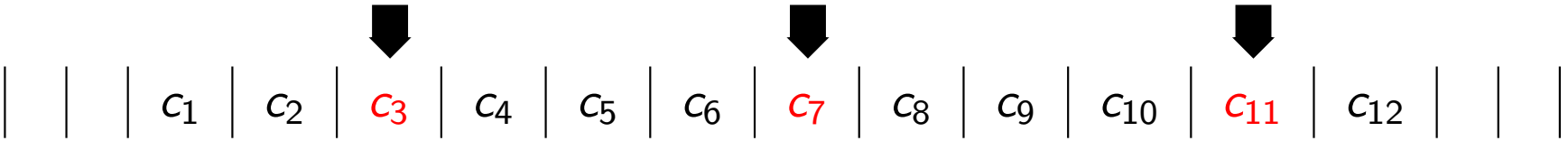


Head 1 : c_1

Head 2 : c_5

Head 3 : c_9

Over-shift Error

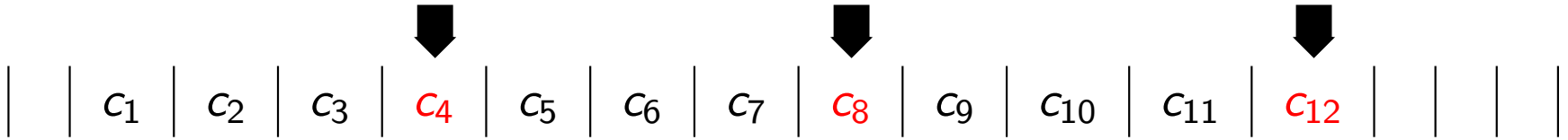


Head 1 : c_1, c_3

Head 2 : c_5, c_7

Head 3 : c_9, c_{11}

Over-shift Error



Head 1 : c_1, c_3, c_4

Head 2 : c_5, c_7, c_8

Head 3 : c_9, c_{11}, c_{12}

Limited-Burst-Deletion

Output:

$$\begin{pmatrix} c_1 & c_3 & c_4 \\ c_5 & c_7 & c_8 \\ c_9 & c_{11} & c_{12} \end{pmatrix} = (u_1, u_3, u_4)$$

Model 2: An over-shift error can be modeled as a burst of consecutive deletions of limited length.

Question 2: How to construct a q-ary code correcting a burst of consecutive deletions of limited length?

Answer 2: C. Schoeny, A. Wachter-Zeh, R. Gabrys and E. Yaakobi (2017).

Multiple Limited-Burst-Deletions

Model 3: Multiple over-shift errors can be modeled as multiple bursts of consecutive deletions of limited length.
Question 3: How to construct a q-ary code correcting multiple bursts of consecutive deletions of limited length?
Answer 3: ?

NANYANG TECHNOLOGICAL UNIVERSITY

Limited-Shift-Error

Model 4: In racetrack memory, both under-shift and over-shift can occur. Hence, there are two kinds of errors: sticky-insertions and burst of deletions.

Question 4: How to construct a q-ary code correcting a combination of multiple sticky-insertions and multiple bursts of deletions of limited length?

Answer 4: ?

NANYANG TECHNOLOGICAL UNIVERSITY

Our Main Results

Theorem 1

Given $0 < \delta, \epsilon < 1$, there exists a q-ary b-limited t_1-burst-deletion-correcting code $\mathcal{C}_1$ of length ℓ such that its rate satisfies

$$1 - \delta \geq R_1 = \frac{\log |\mathcal{C}_1|}{\ell} \geq (1 - \log_q(b+1)) \cdot (1 - \delta - \epsilon)$$

where $t_1 \cdot b = \delta \cdot \ell$.

The code is asymptotic optimal when q tends to infinity.

NANYANG TECHNOLOGICAL UNIVERSITY

Our Main Results

Theorem 2

Given $0 < \delta, \epsilon < 1$, there exists a q-ary b-limited t_1-burst-deletion t_2-sticky-insertion-correcting code $\mathcal{C}_2$ of length ℓ for any arbitrarily large t_2 and $t_1 \cdot b = \delta \cdot \ell$ such that its rate

$$1 - \delta \geq R_2 = \frac{\log |\mathcal{C}_2|}{\ell} \geq (1 - \log_q(b+2)) \cdot (1 - \delta - \epsilon).$$

The code is asymptotic optimal when q tends to infinity.

Construction

Definition

- A cyclic sequence $\sigma = (\sigma_1, \ldots, \sigma_\ell)$ is called a *de Bruijn sequence* of strength h over an alphabet of size q if all ℓ possible substrings of length h are distinct. It is known that $\ell \leq q^h$.
- A q-ary sequence $\pi = (\pi_1, \ldots, \pi_\ell)$ is called a *b-bounded de Bruijn sequence* of strength h if all length-h subvectors $\pi[i, i+h-1]$ in b consecutive positions are distinct. That is, we can always determine the position i of sub-vector $\pi[i, i+h-1]$ provided the estimation of that position in a segment of length b.

Construction

Construction

Let $\pi = (\pi_1, \pi_2, \ldots, \pi_\ell)$ be a b-bounded de Bruijn sequence of strength one over an alphabet of size q_1. Let $\mathcal{C}_{q_2}(\ell, t)$ be a q_2-ary t-erasure-correcting code of length ℓ. Let $q = q_1 \cdot q_2$. For each word $\boldsymbol{c} = (c_1, c_2, \ldots, c_\ell) \in \mathcal{C}_{q_2}(\ell, t)$, we define $\boldsymbol{f}(\boldsymbol{c}, \pi) = (f_1, f_2, \ldots, f_\ell)$ such that $f_i = (\pi_i, c_i)$ for all $1 \leq i \leq \ell$. We construct the following q-ary code of length ℓ, $\mathcal{C}_q(b, \ell, t) = \{\boldsymbol{f}(\boldsymbol{c}, \pi) : \boldsymbol{c} \in \mathcal{C}_{q_2}(\ell, t)\}$.

NANYANG TECHNOLOGICAL UNIVERSITY

Construction

Theorem 3
The code $\mathcal{C}_q(b+1, \ell, t)$ from the above construction is a q-ary b-limited t_1-burst-deletion-correcting code where $t = t_1 \cdot b$.

Theorem 4
The code $\mathcal{C}_q(b+2, \ell, t)$ from Construction 1 is a q-ary b-limited t_1-burst-deletion t_2-sticky-insertion-correcting code for any integer t_2 and $t = t_1 \cdot b$.

NANYANG TECHNOLOGICAL UNIVERSITY

Conclusion and Discussion

- Non-binary codes correcting multiple bursts of deletions of limited length and multiple sticky-insertions are constructed.
- These codes can be decoded efficiently without knowing the number of deletions and insertions.
- These codes can be applied to correct limited-shift errors in racetrack memories and to correct block deletions in DNA-based storage.
- Some more results can be found in our paper.

NANYANG TECHNOLOGICAL UNIVERSITY

Furtherwork

- Construct a good q-ary code correcting t deletions and sticky-insertions with small q.
- Finding some coding schemes to combat under- and over-shift errors in racetrack memories.

NANYANG TECHNOLOGICAL UNIVERSITY

THANK YOU

THANK YOU FOR YOUR ATTENTION.