2025/8/19 WAAC2025

LZ-Start-End: An LZ-style Compressor Supporting $O(\log n)$-time Random Access

Hiroki Shibata (Kyushu University),
Yuto Nakashima (Kyushu University),
Yutaro Yamaguchi (The University of Osaka),
Shunsuke Inenaga (Kyushu University)

Background

- **Query Capability**: Processing compressed data (e.g., extraction) without full decompression.
- **Compression performance** and **query capability** are in a trade-off.
  - LZ compression [Ziv and Lempel, 1977]:High compression ratio, but no query support.
  - Grammar Compression [Kieffer and Yang, 2000]: Supports various queries, but low compression ratio.
- **Goal**: High compression with query capability.

Our Contribution: LZ-Start-End (LZSE) Compression

- **Compression Performance**: Achieves compression performance between that of grammar and LZ.
- **Query Capability**: Supports compressed data access as fast as grammar compression.

	Compression	Querying
LZ comp.	Very High	Not Supported
Grammar comp.	Moderate	Excellent
LZSE comp.	High	Good

Our Results & Focus of This Talk

1. **Relationship between LZSE and grammar**
   - Converting LZSE and Grammar
   - A theoretical gap in compression power

Focus of This Talk

2. $O(\log n)$**-time random access index for LZSE**
3. **Linear-time computation of greedy LZSE**
4. **A lower bound between greedy/optimal LZSE**

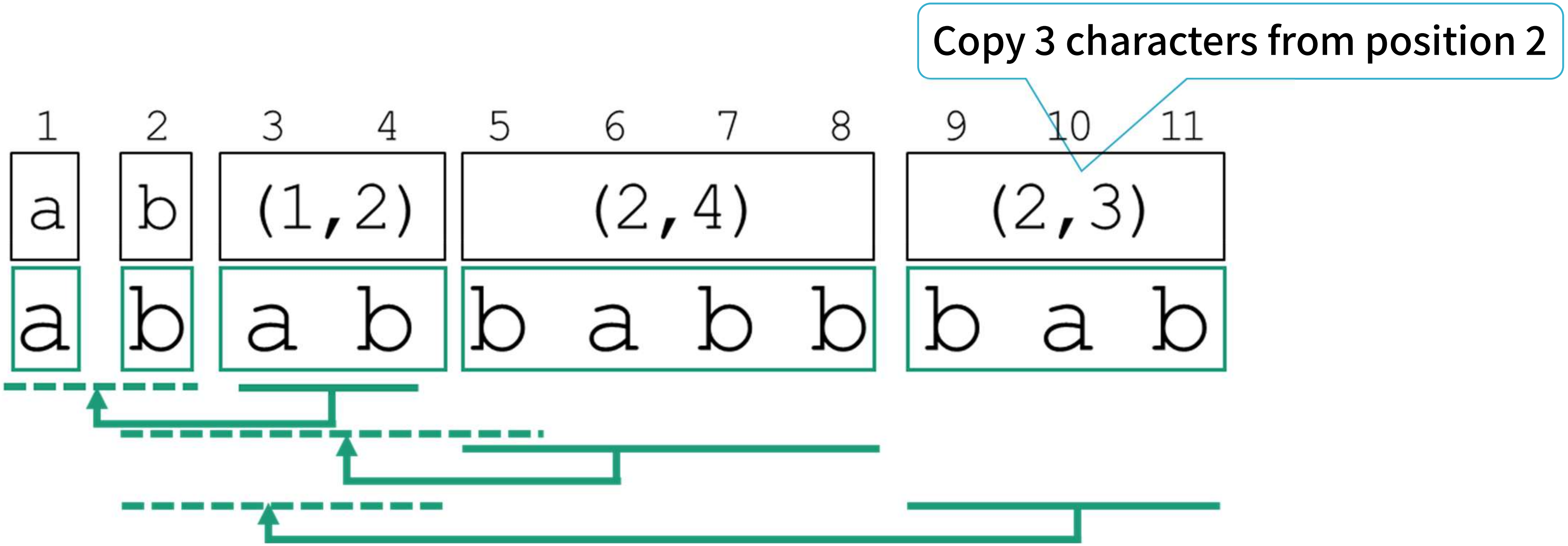
etc.

n : input string length

Background 1: Lempel-Ziv (LZ) Compression [Ziv and Lempel, 1977]

Represents data via a sequence of factors (LZ factorization):

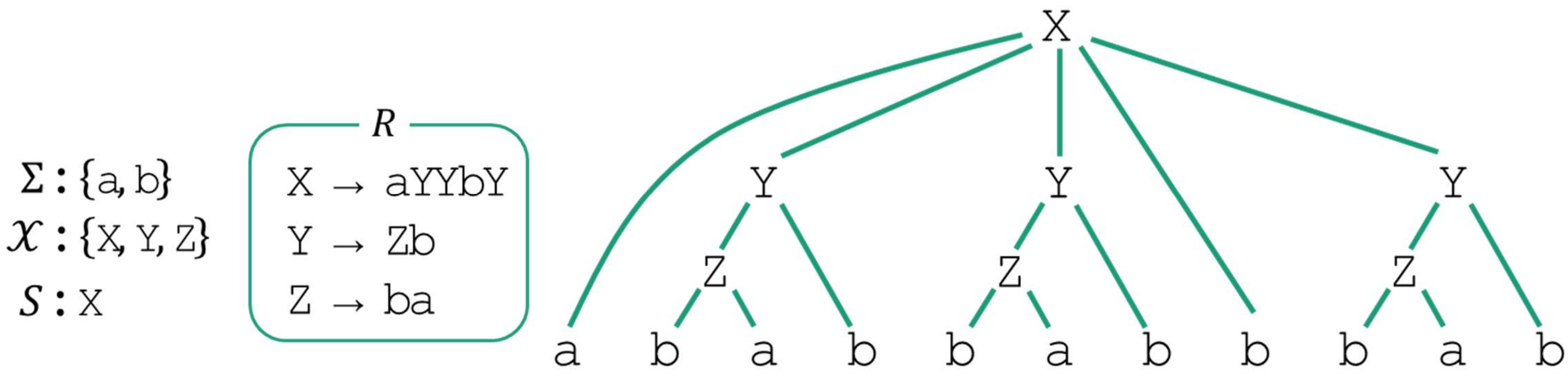
1. A single literal character
2. A copy of a previous substring



Background 2: Grammar Compression [Kieffer and Yang, 2000]

Represents data using a **Context-Free Grammar (CFG)** with:

1. A set of terminals Σ, A set of non-terminals $\mathcal{X}$
2. Production rules $R = \{X_i \rightarrow expr_i \mid X_i \in \mathcal{X}\}$
3. Start symbol $S \in \mathcal{X}$

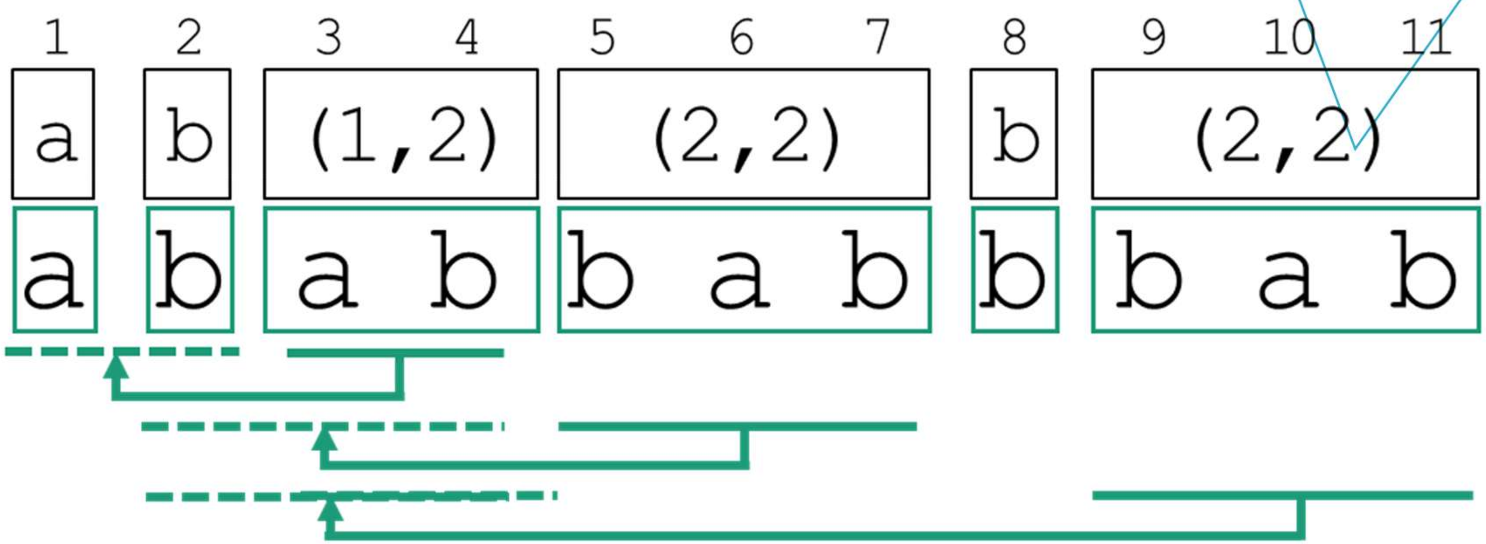


Proposed Method: LZ-Start-End (LZSE) Compression

Represents data via a sequence of factors (LZSE factorization):

1. A single literal character
2. A copy of <u>consecutive previous factors</u>

Copy 2 factors starting from the 2nd- factor

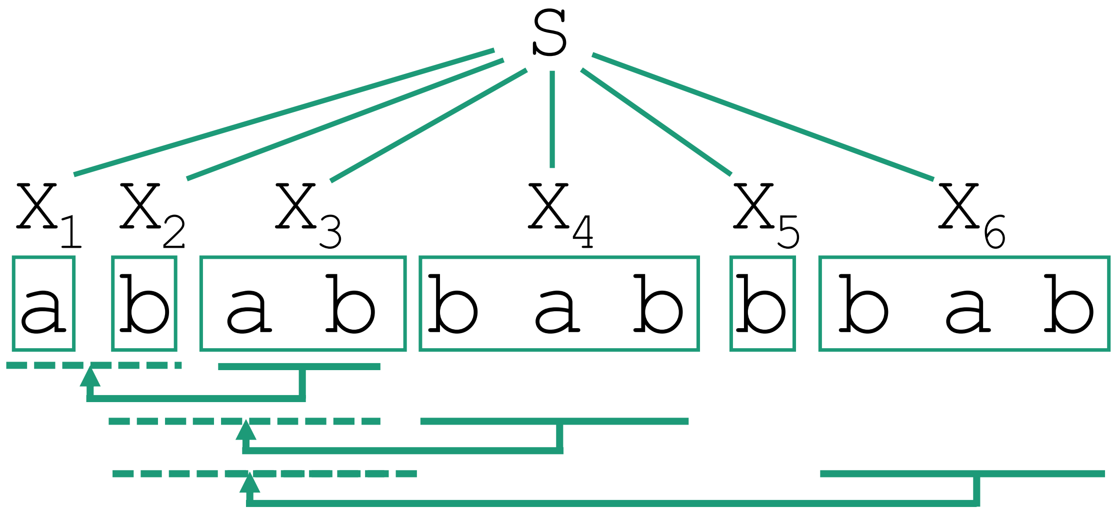


LZSE → Grammar

- **Start symbol:** $S \rightarrow X_1 X_2 \cdots X_m$ (each non-terminal corresponds to a factor)
- **Factor rules:**
  - $X_i \rightarrow c$ (for a literal factor representing a character c)
  - $X_i \rightarrow X_j \cdots X_k$ (for a copy factor referring to $X_j \cdots X_k$)
- The size of this grammar is $O(m^2)$ (m : the number of LZSE factors)

R

$S \rightarrow X_1 \ldots X_6$
$X_1 \rightarrow a$
$X_2 \rightarrow b$
$X_3 \rightarrow X_1 X_2$
$X_4 \rightarrow X_2 X_3$
$X_5 \rightarrow b$
$X_6 \rightarrow X_2 X_3$

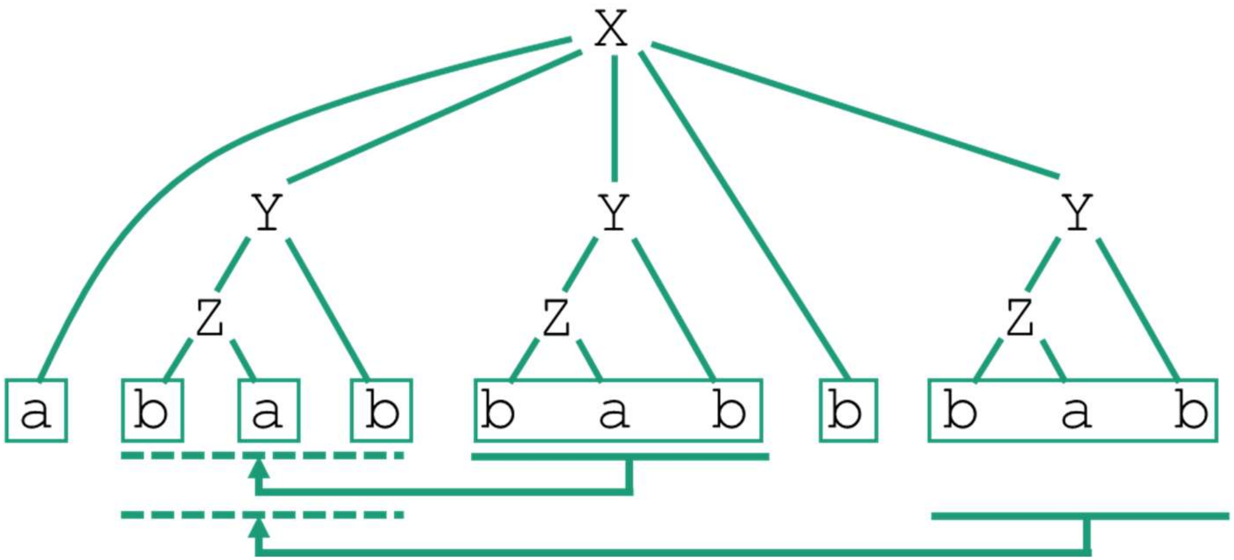


Grammar → LZSE

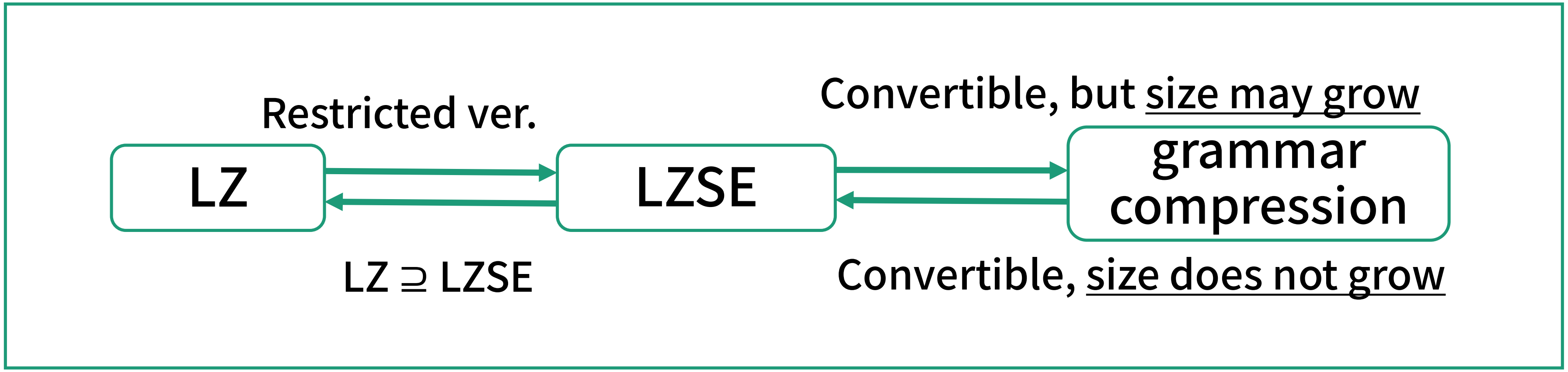
Method: **Grammar Decomposition** [Rytter, 2003]

Replace subsequent occurrences of a non-terminal with a pointer to the first one.

- **Resulting LZ size ≤ original grammar size.**
- **The resulting LZ factorization is actually an LZSE factorization.**



Relationships and a Question



Question: Do LZSE and Grammar have the same compression power?

(i.e., are their minimal sizes within a constant factor?)

- **Known gap between LZ & LZSE:** $\Theta(\log n)$ **in the worst-case**
- **Can we find a similar gap between LZSE & the smallest grammar?**

A separation between LZSE and grammar

Theorem

For an arbitrarily large m, there exists a string T of length $\Theta(m^2)$ where:

- The size of greedy LZSE of T : $\Theta(m)$
- The size of the smallest grammar of T : $\Omega\big(m\alpha(m)\big)$

This theorem implies that LZSE is asymptotically more powerful than grammar compression.

- For any string: |the smallest LZSE| $\leq$ |the smallest grammar|
- For a specific string: |the greedy LZSE| $\ll$ |the smallest grammar|

$\alpha(x)$: the inverse Ackermann function

Lower Bound on Grammar Size (1)

Off-line Range Product Problem

- **Input**: a sequence $x_1, \dots, x_m \in X^m$, ranges $[l_1, r_1], \dots, [l_m, r_m]$
- **Output**: $q_i = \bigotimes_{j=l_i}^{r_i} x_j \ (1 \le i \le m)$

($\otimes$: a semigroup operation on X)

Known lower bound: **Requires $\Omega(m\alpha(m))$ semigroup operations for some inputs.** [Chazelle and Rosenberg, 1991]

Lower Bound on Grammar Size (2)

Construct a string T from the ORPP instance:

$$T = x_1 \cdots x_m \$_1 Q_1 \$_2 Q_2 \$_3 \cdots \$_m Q_m \$_{m+1}$$

$$Q_i = x_{l_i} \cdots x_{r_i} \; (1 \le i \le m)$$

$\$_i$ **: delimiter**

Example: Sequence: $(1,2,3,4)$ **, Query:** $[2,3], [1,3], [2,4], [3,4]$

$$T = 1234 \;\; \$_1 \; 23 \;\; \$_2 \; 123 \;\; \$_3 \; 234 \;\; \$_4 \; 34 \;\; \$_5$$

Lower Bound on Grammar Size (3)

Construct a string T from the ORPP instance:

$$T = x_1 \cdots x_m \$_1 Q_1 \$_2 Q_2 \$_3 \cdots \$_m Q_m \$_{m+1}$$

$$Q_i = x_{l_i} \cdots x_{r_i} \ (1 \leq i \leq m)$$

$\$_i$: delimiter

The proving strategy:

- **Show that we can solve the ORPP instance in $O(g)$ operations if there exists a grammar of T with size g.**
- **Due to the hardness of ORPP, the grammar size must be $\Omega\big(m\alpha(m)\big)$.**

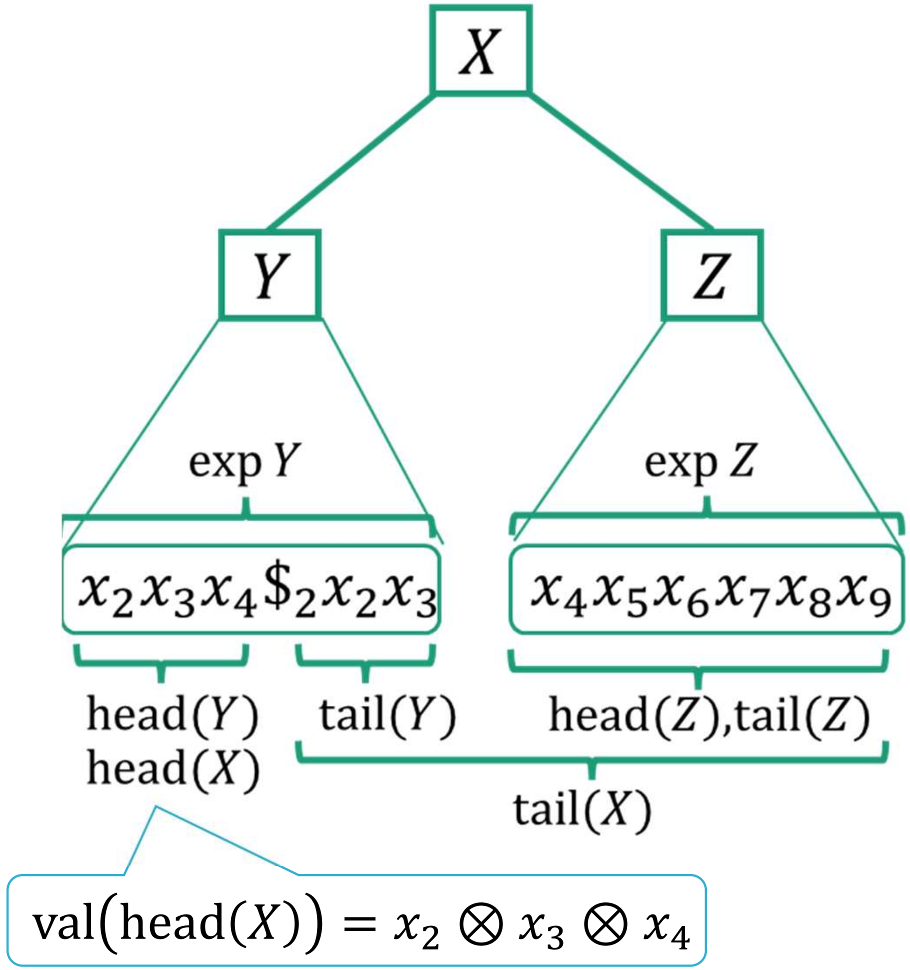
Lower Bound on Grammar Size (4)

For each nonterminal X in the grammar:

- $\exp X$: **the string derived from** X
- $\mathrm{head}(X)$: **the prefix of** $\exp X$ **before the first delimiter**
- $\mathrm{tail}(X)$: **the suffix of** $\exp X$ **after the last delimiter**

For a string $S = s_1 \cdots s_{|S|}$ **on** X**,**

let the value of S **as** $\mathrm{val}(S) = \bigotimes_{i=1}^{|S|} s_i$.



Lower Bound on Grammar Size (5)

Property 1

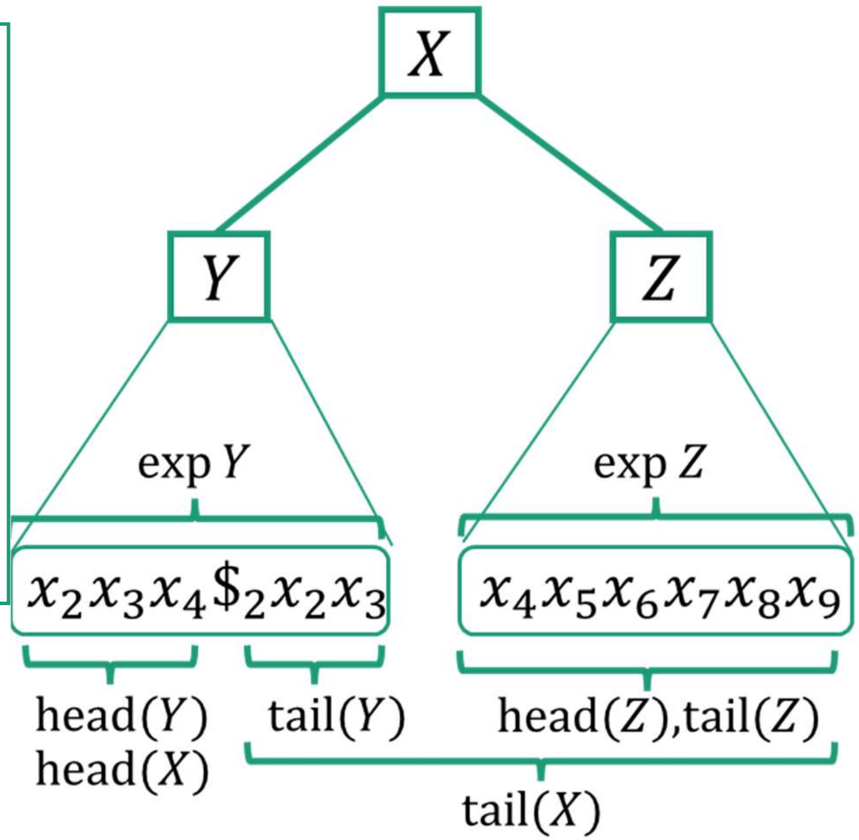
For a rule $X \to YZ$, we can compute

$$\mathrm{val}\big(\mathrm{head}(X)\big), \mathrm{val}\big(\mathrm{tail}(X)\big)$$

from the values of

$$\mathrm{head}(Y), \mathrm{tail}(Y), \mathrm{head}(Z), \mathrm{tail}(Z)$$

using $O(1)$ semigroup operations.



Example from Figure:

- $\mathrm{val}\big(\mathrm{head}(X)\big) = \mathrm{val}\big(\mathrm{head}(Y)\big)$
- $\mathrm{val}\big(\mathrm{tail}(X)\big) = \mathrm{val}\big(\mathrm{tail}(Y)\big) \otimes \mathrm{val}\big(\mathrm{tail}(Z)\big)$

Lower Bound on Grammar Size (6)

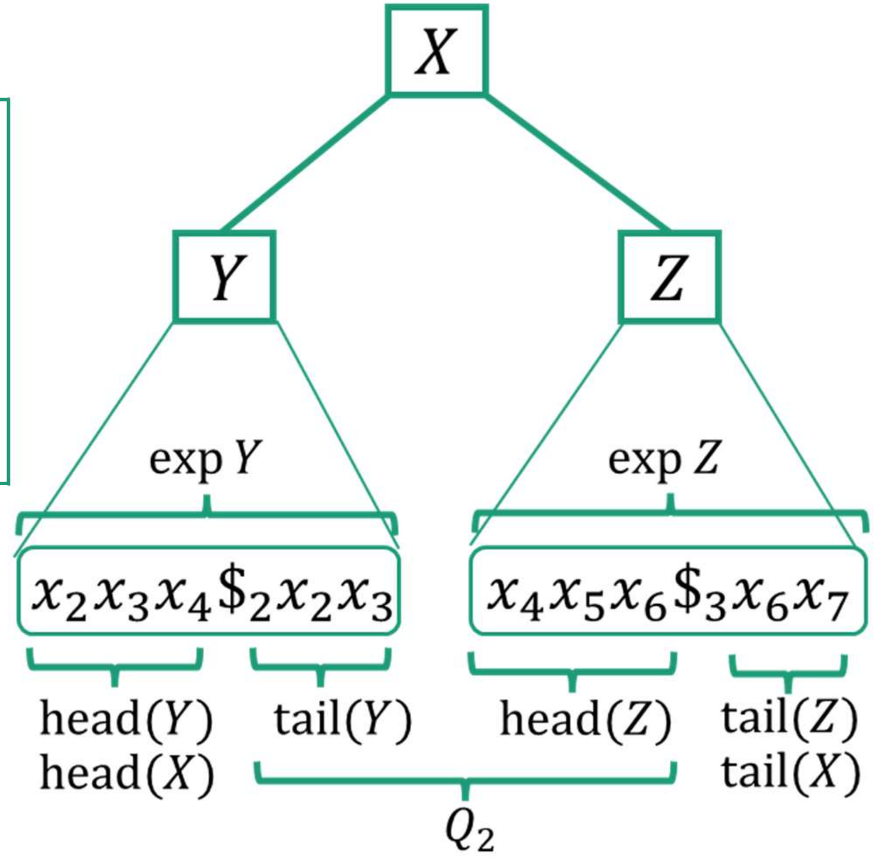
Property 2

For a rule $X \to YZ$, if $\exp Y$ contains $\$_i$ and $\exp Z$ contains $\$_{i+1}$, we can compute q_i as

$$q_i = \mathrm{val}(Q_i) = \mathrm{val}\big(\mathrm{tail}(Y)\big) \otimes \mathrm{val}\big(\mathrm{head}(Z)\big).$$

Example from Figure:

$$\begin{aligned}\mathrm{val}(Q_2) &= \mathrm{val}(x_2x_3x_4x_5x_6)\\ &= \mathrm{val}(x_2x_3) \otimes \mathrm{val}(x_4x_5x_6)\\ &= \mathrm{val}\big(\mathrm{tail}(Y)\big) \otimes \mathrm{val}\big(\mathrm{head}(Z)\big)\end{aligned}$$



Lower Bound on Grammar Size (7)

1. Convert a grammar to SLP of size $O(g)$.
2. Compute $\mathrm{val}\big(\mathrm{head}(X)\big)$ and $\mathrm{val}\big(\mathrm{tail}(X)\big)$ for all nonterminals X.
   - It takes $O(g)$ semigroup operations.
3. Compute all query answers $q_i = \mathrm{val}(Q_i)$.
   - It takes $O(m) \subseteq O(g)$ semigroup operations.

Total cost: $O(g)$ semigroup operations.

Final Comparison: Grammar vs. LZSE

- **Smallest Grammar Size:**
  - The problem can be solved in $O(g)$ semigroup operations
  - The lower bound of this problem is $\Omega(m\alpha(m))$

→ There exists an input for which the size of smallest grammar is $\Omega(m\alpha(m))$

- **Greedy LZSE size: $O(m)$ (see the following figure)**

$$T = \boxed{x_1} \cdots \boxed{x_m}\boxed{\$_1}\boxed{Q_1}\boxed{\$_2}\boxed{Q_2}\boxed{\$_3} \cdots \boxed{\$_m}\boxed{Q_m}\boxed{\$_{m+1}}$$

Result: An asymptotic gap exists between the LZSE and grammar comp.

Summary

LZSE: A restricted LZ factorization

- **Positioned between LZ and Grammar-based methods.**
  - An asymptotically strong compression power.
- **Supports efficient queries, same as grammar-based methods.**