Optimally Computing Compressed Indexing Arrays Based on the Compact Directed Acyclic Word Graph

Hiroki Arimura[1]
Shunsuke Inenaga[2]
Yasuaki Kobayashi[1]
Yuto Nakashima[2]
Mizuki Sue[1]

1) Graduate School of IST, Hokkaido University, Japan
2) Department of Informatics, Kyushu University, Japan

For details and proofs, visits the arXIv site of this talk:
- Manuscript at https://arxiv.org/abs/2308.02269 ;
- Slide pdf at "Code section" or Github https://ikndeva.github.io)

This work is partly supported by MEXT Grant-in-Aid for Basic Research A, 2000-2004, Japan

Backgrounds

- Increasing amount and types of repetitive texts
  - Markup texts (Wikipedia), Genome sequences
- Development of compressed index structures for these repetitive texts attracts much attention.
- Such indices can compress highly-repetitive texts beyond the entropy bounds up to "compression parameters" – the sizes of indices
- We focus on the relationship between compressed index structures

Three compressed index structures

index	0 1 2 3 4 5
Text T	c a c a o $

- RL-BWT is obtained from SA by taking the preceding letter and run-length encoded

RL-BWT — Size r

o^1 c^2 $^{1} a^2

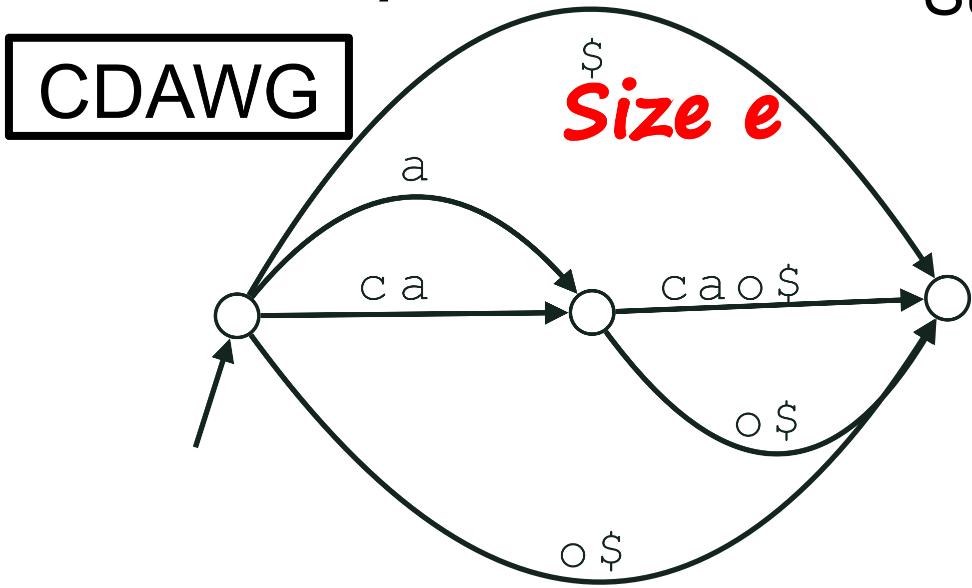
- The LZ-parse is obtained by partitoning T into the longest previous factors (PLFs)

LZ-parse — Size z

c | a | c a | o

- The CDAWG (Compact Directed Word Graphs) for a text T is an automata-based index in a DAG form

- It is obtained from the Suffix Tree of T by merging isomorphic subtrees



Suffixes of T

$

a c a o $

a o $

c a c a o $

c a o $

o $

Three compressed index structures

index	0 1 2 3 4 5
Text T	c a c a o $

- RL-BWT is obtained from SA by taking th
run-leng

RL-BW

- The CDAWG (Compact Directed Word Graphs) for a text T is an automata-based index in a DAG form

the Suffix
ng
s

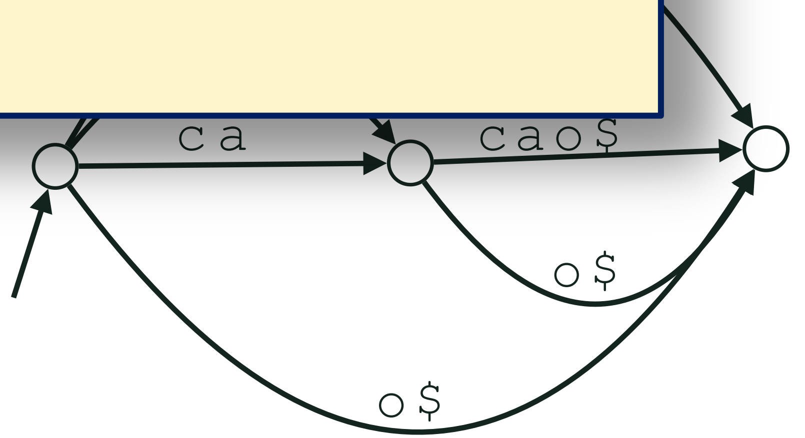
We are interested in the size-relation and computational complexities of conversion between them.

- The LZ-pa
partitonin
previous factors (PLFs)

LZ-parse

Size z

c | a | c a | o



Suffixes of T

$
a c a o $
a o $
c a c a o $
c a o $
o $

Backgrounds: Previous work

- Consider the relationship between their sizes
  - has been studied so far.

RL-BWT
r

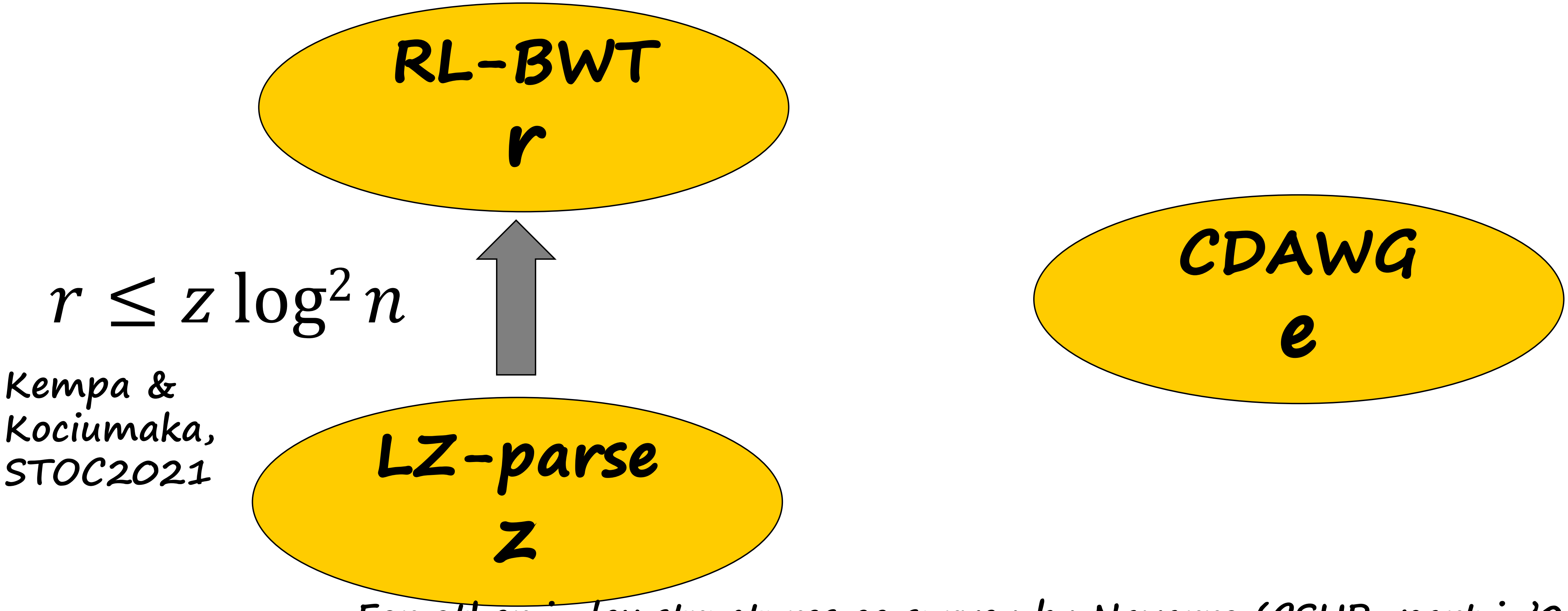
CDAWG
e

LZ-parse
z

For other index structures ee survey by Navarro (CSUR, part i, '21)

Backgrounds: Previous work

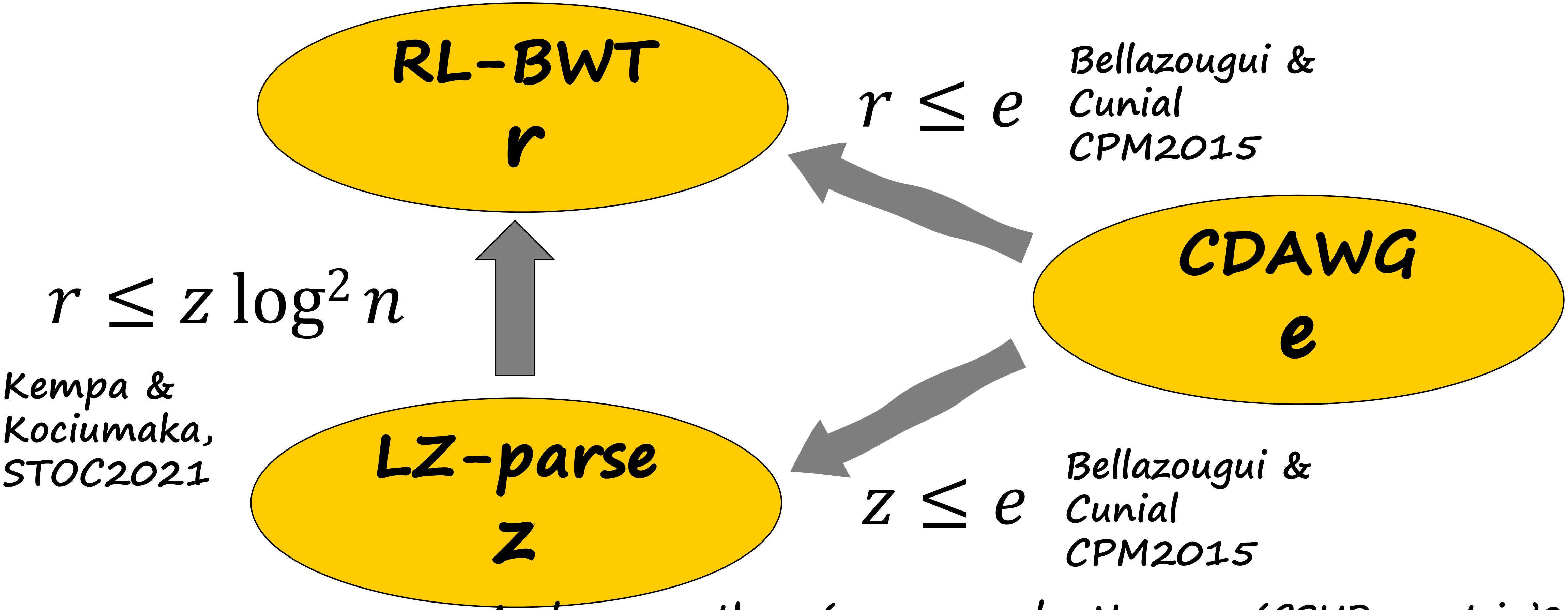
- Consider the relationship between their sizes
  - has been studied so far.



For other index structures ee survey by Navarro (CSUR, part i, '21)

Backgrounds: Previous work

- **Consider the relationship between their sizes**
  - has been studied so far.



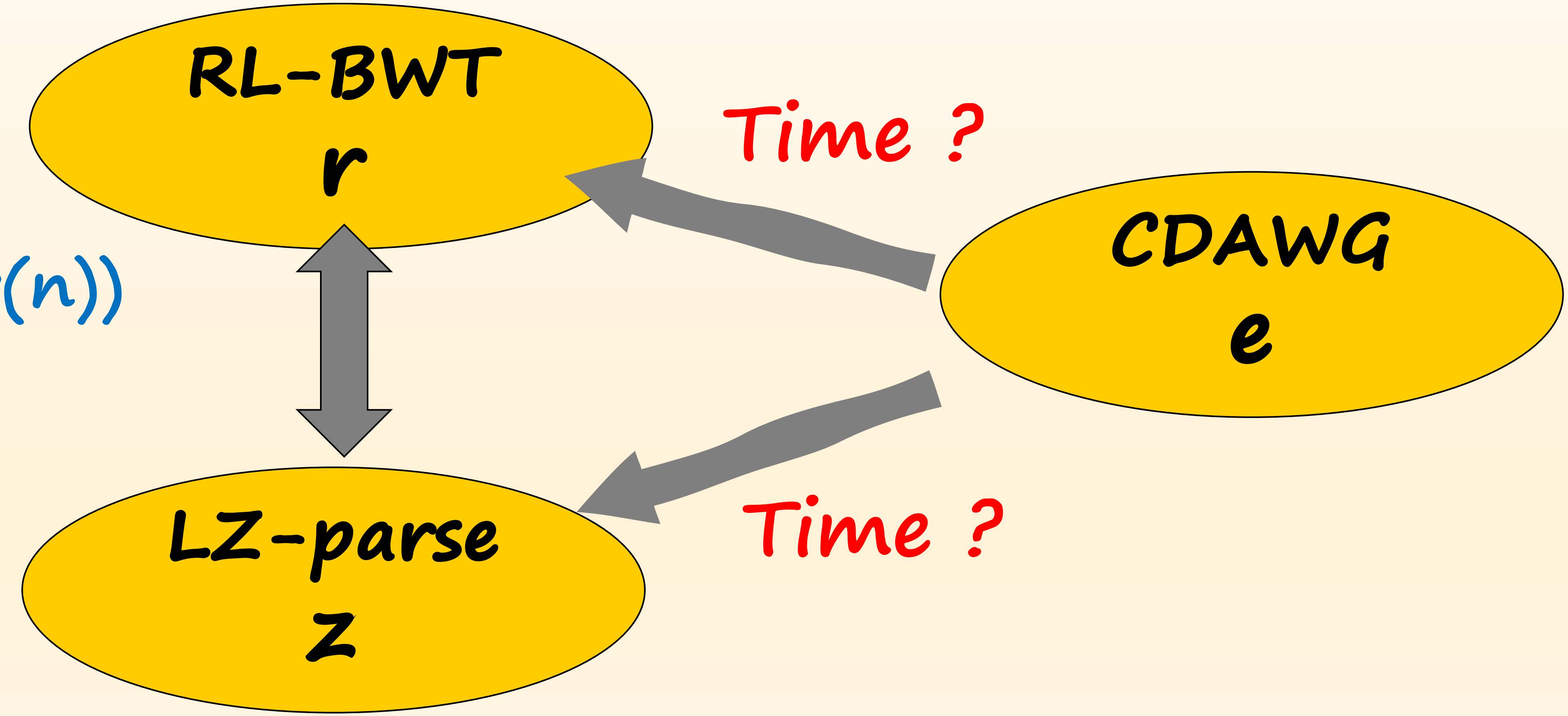
And many others (see survey by Navarro (CSUR, part i, ’21)

Backgrounds: Previous work

- The time and space complexities of conversions
  - have not been studied very much

$O(z \text{ polylog}(n))$ time

Kempa & Kociumaka, CACM 2022



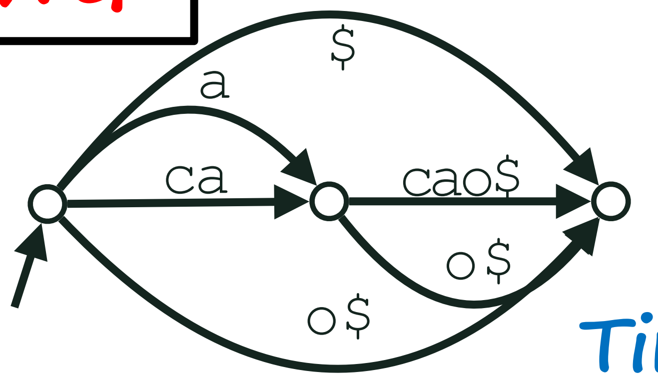
Research Goal:

We devise efficient algorithms that solves the conversion problem from the CDAWG for a text T into various compressed indexes for T in linear time and space in the combined input/output sizes

the conversion problem

CDAWG

Compressed index A



Conversion

Compressed index B

RL-BWT

$o^1\ c^2\ \$^1\ a^2$

LZ-parse

Irr-PLCP

Irr-LPF

Time & Space complexity

Original text

Related work

Sublinear time and space conversion between two indices

- Kempa [SODA'19]
  - Converting an RL-BWT-based index into the irreducible PLCP, CSA, and LZ-parse for a text T of length n in $O(n / \log_\sigma n + r \operatorname{polylog} n)$ time and $O(r)$ space.
- Kempa & Kociumaka [STOC'21, CACM'22]
  - Converting the LZ77-parse of a text T into the RL-BWT for T in $O(z \operatorname{polylog} n)$ time and space.
  - This work solved a long-standing open problem
- Bannai et al. [CPM'13]
  - Converting an SLP of size g into LZ78-parse of size z_{78} in $O(g + z_{78} \log z_{78})$ time and space.
  - Combined with Belazzougui & Cunial [CPM'15], we obtain the conversion from the CDAWG for T into LZ78-parse in $O(e + z_{78} \log z_{78})$ time and space.

Main results

Thm (4.1, 5.1, 5.2): For any integer alphabet Σ, we can convert the CDAWG G of size e for a text T into the following compressed indexing structures for T in $O(e)$ deterministic time and words of space:

- **The RL-BWT** (run-length BWT) of size r
- **The irreducible PLCP** (permuted LCP) array of size r
- **The quasi-irreducible LPF** (longest previous factor) array of size e (def. Sec. 2 of this paper)
- **The Lex-parse** of size $2r = O(r)$
- **The LZ-parse** of size z

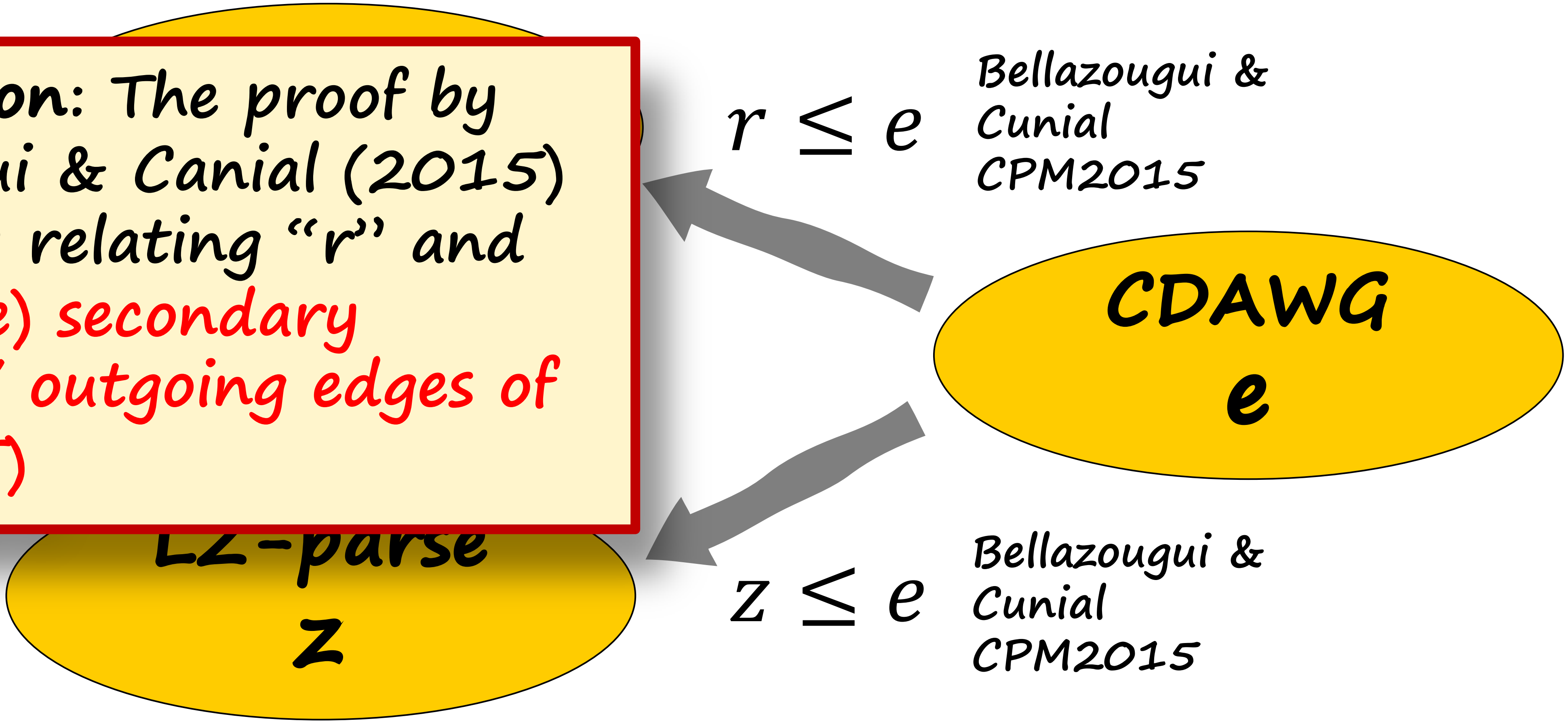
G is given in either

- the CDAWG of size e with the read only text of length n,
- the self-index version of CDAWG of size $O(e)$ without a text

Algorithms

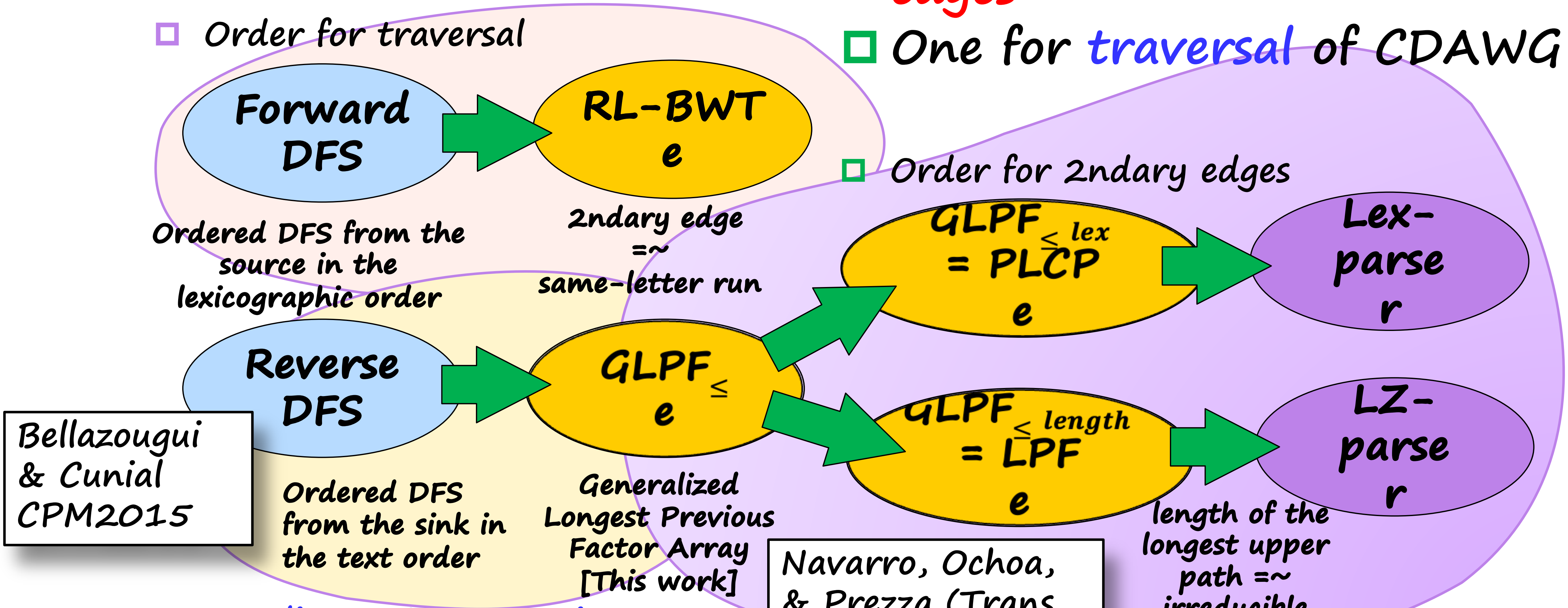
Coming back to the relationship between the sizes …

Observation: The proof by Bellazougui & Canial (2015) is done by relating "r" and "z" to O(e) secondary incoming/ outgoing edges of CDAWG(T)



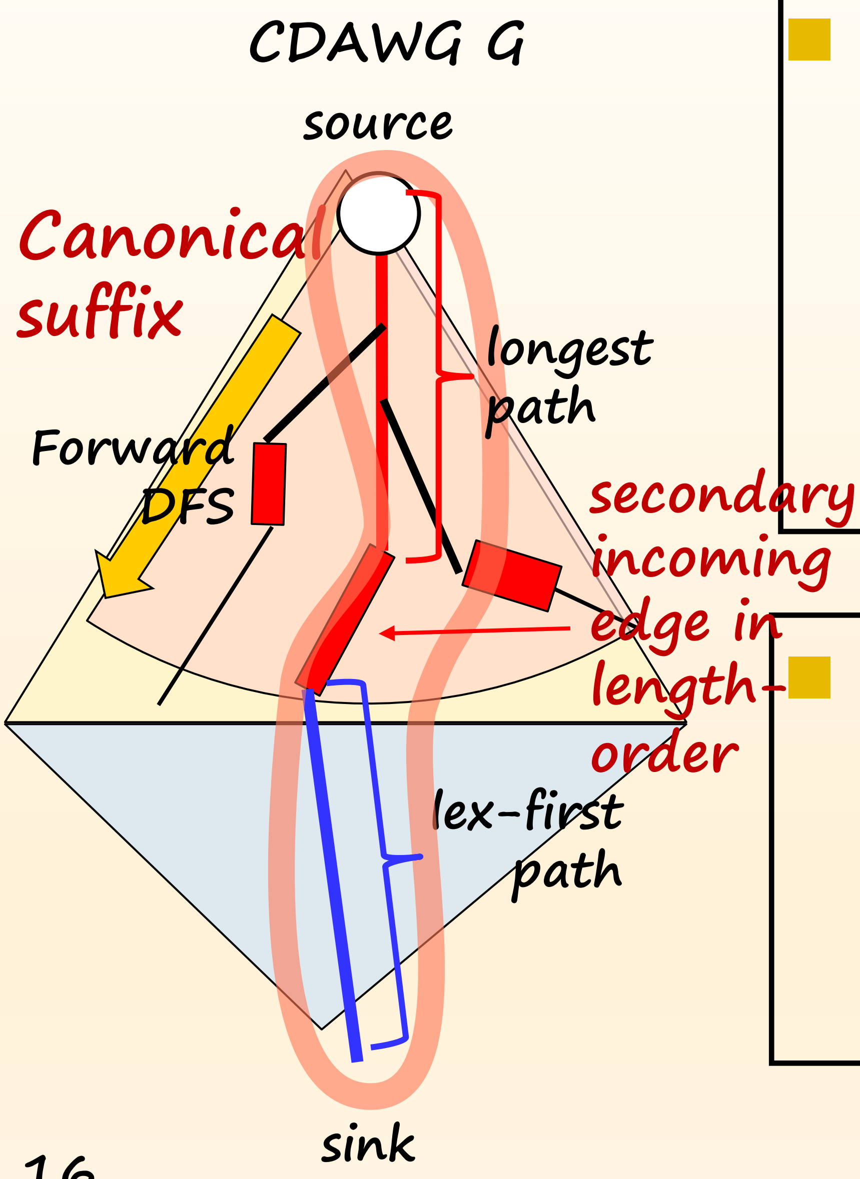
Our approach

- We use two orders of paths
  - Order for defining 2ndary edges
  - One for traversal of CDAWG



- We generalize PLCP & LPF into GLPF by the framework of (NOP'20)

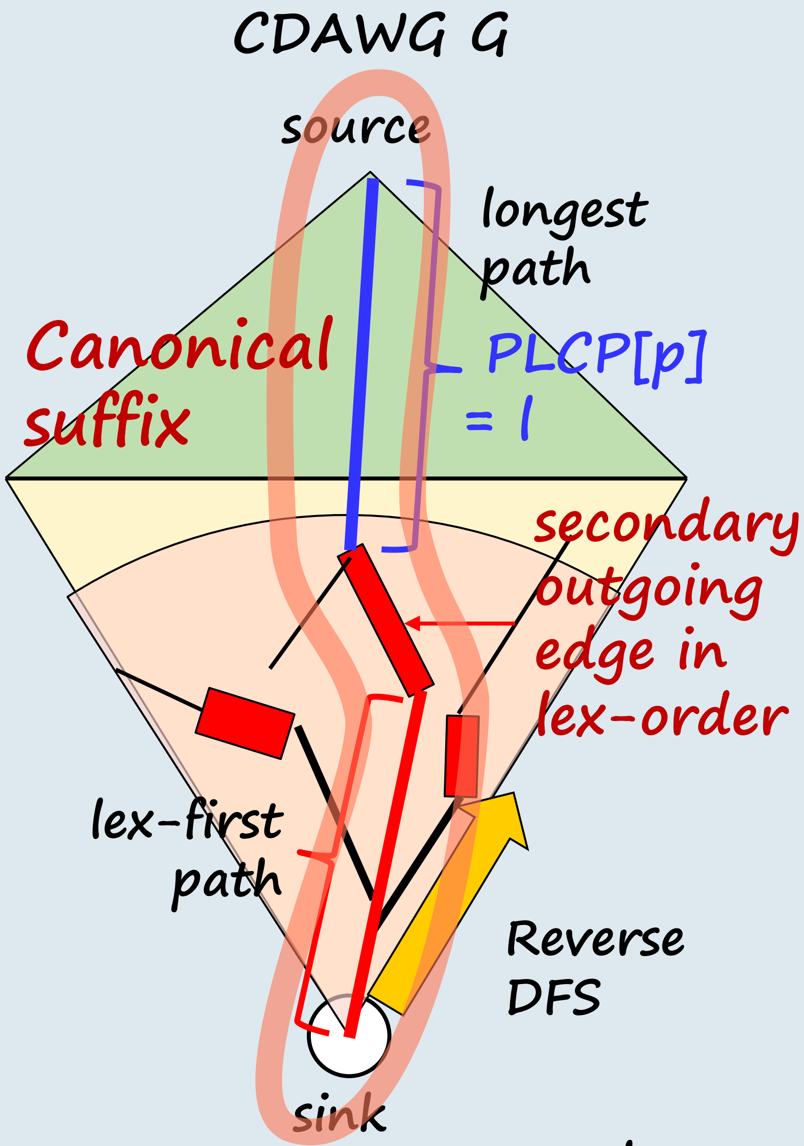
Sec4. Computing RL-BWT in O(e) time&space



- Observation A1: O(e) secondary incoming edges of CDAWG(T) under the length-order correspond to subintervals of the same-letter runs of the BWT.
  (this is because such a search path defines a non-left-maximal factor in T)

- Observation A2: O(e) incoming edges of CDAWG(T) can be enumerated in the lexicographic order of its "canonical suffix" by the forward DFS from the source.

Sec5. Computing PLCP in O(e) time&space



- **Observation A1**: O(e) secondary outgoing edge of CDAWG(T) under the length-order determines the irreducible value PLCP[p] = l by the length l of the longest path from the source to the corresponding branching node

- **Observation A2**: O(e) secondary outgoing edges can be enumerated in the text order of its "canonical suffix" by the reverse DFS from the sink.

We can extend the above result from PLCP to PLPF by employing the definition of 2ndary outgoing edges in length-order

Conclusions

- Conversion problem from the CDAWG into other compressed indices for highly-repetitive texts:
  - Input: either the CDAWG of a text T or its self-index
  - Output: RL-BWT, irreducible PLCP and LPF, Lex- and LZ-parse
- We obtained Optimal O(e) time and space conversion algorithms for the above indices:
  - Effective version of the result by Belazzougui & Cunial (CPM'15) that r <= e and z <= e to actual conversion.
- Techniques:
  - Characterization of the "irreducible values" by secondary edges.
  - Forward/reverse DFS under the lexicographic/text order
- Future Work:
  - Conversion from RL-BWT & LZ-parse into CDAWG in O(e) time & space; Extension of the techniques to other indexing structures

Thank you!

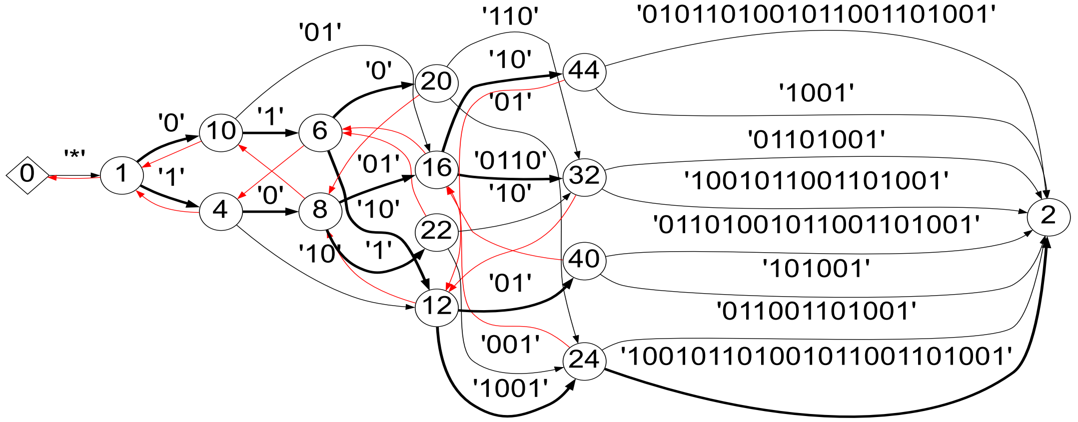
For details and proofs, visits the arXIv site of this talk:

- Manuscript at https://arxiv.org/abs/2308.02269 ;
- Slide pdf at "Code section" or Github https://ikndeva.github.io)

Appendix: When e competes r?

- **Facts: For Thue-Morse words, the CDAWG & RL-BWT have the sizes:**
  - $e = O(\log n)$
    [Radoszewski & Rytter '12]
  - $r = \Theta(\log n)$ [Brlek+ '19].

CDAWG for the k-th Thue-Morse word with Depth = O(k) and #nodes = 2O(k)



Thue-Morse word
$\tau(k) = \phi^k(0)$

With the morphism
- $\phi(0) = 01$
- $\phi(1) = 10$

- **Observation:** In this case, there is a chance for our O(e)-time conversion method on CDAWG to compete a method based on RL-BWT with O(r polylog(n))-time to compute, e.g., LZ or irr. PLCP.

[14], Radoszewski, J., Rytter, W., JDA, Vol.11 (2012)
[6] Brlek, S., Frosini, A., Mancini, I., Pergola, E., Rinaldi, S., IWOCA 2019 (2019)