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

A longer version of this paper can be found in arXiv repository site.

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 repetitive texts attracts much attention. E.g.,
  - RL-BWT, irreducible PLCP arrays, Lex-parse – size r
  - LZ-parse (LZ76) – size z
  - CDAWG (Compact Directed Word Graphs) – size e

  These indices can compress highly-repetitive texts beyond the entropy bounds up to r, z, and e
- Natural questions: What is the relationships among their sizes?; what is the complexities of conversion?

Backgrounds: Brief History

- We focus on the relationship between three compressed indices.

Suffix tree Size n

Irr. PLCP r

- An automata-based index, obtained from the Suffix Tree of T by merging isomorphic subtrees

- BWT is the array of the preceding letters at the starting positions in SA
- r is the number of equal-letter runs

RL-BWT r

CDAWG e

- LZ-parse is a macro scheme based on the previous factors.
- z is the number of equal-letter runs

LZ-parse z

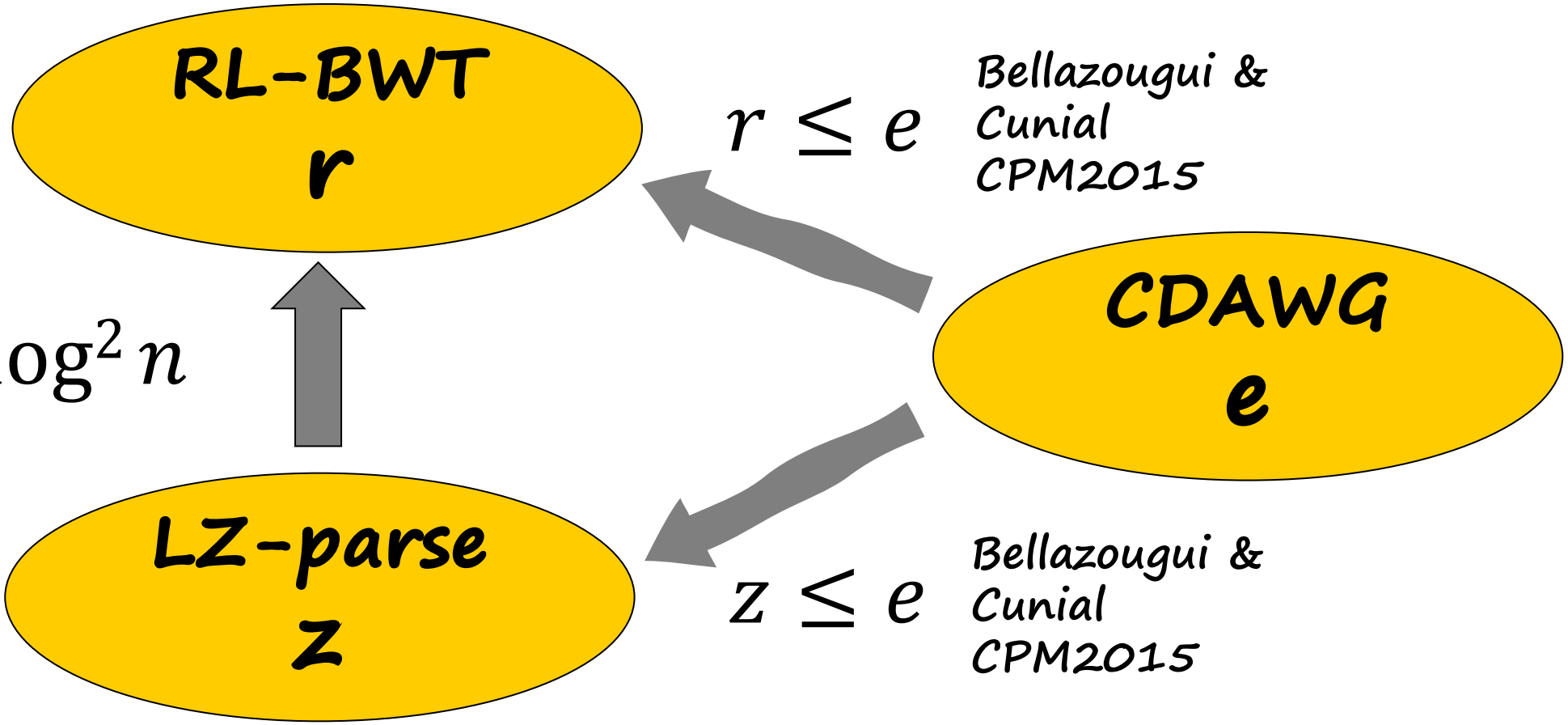
q-Irr. LPF e

- mu: the number of nodes = #maximal extensions
- e is the number of tree- and suffix-edges

Backgrounds: Brief History

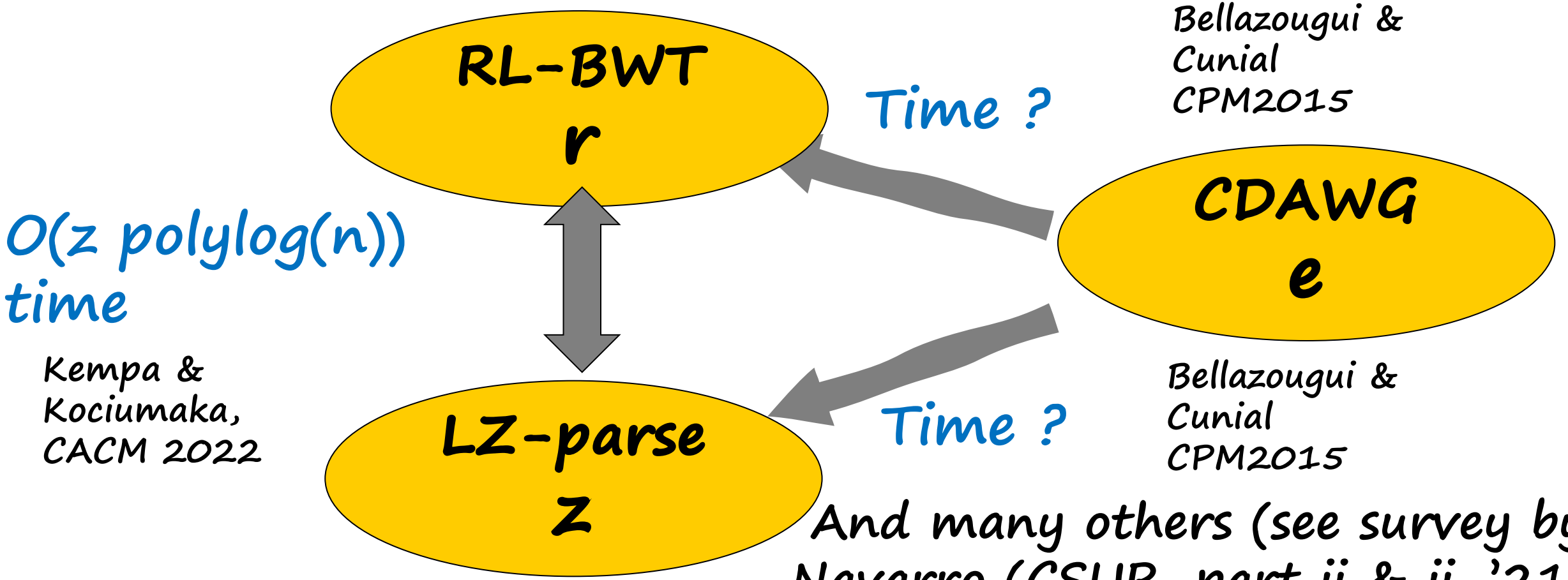
- We focus on the relationship between the indices of the sizes r, z, and e.

Relationship on the sizes



Brief History

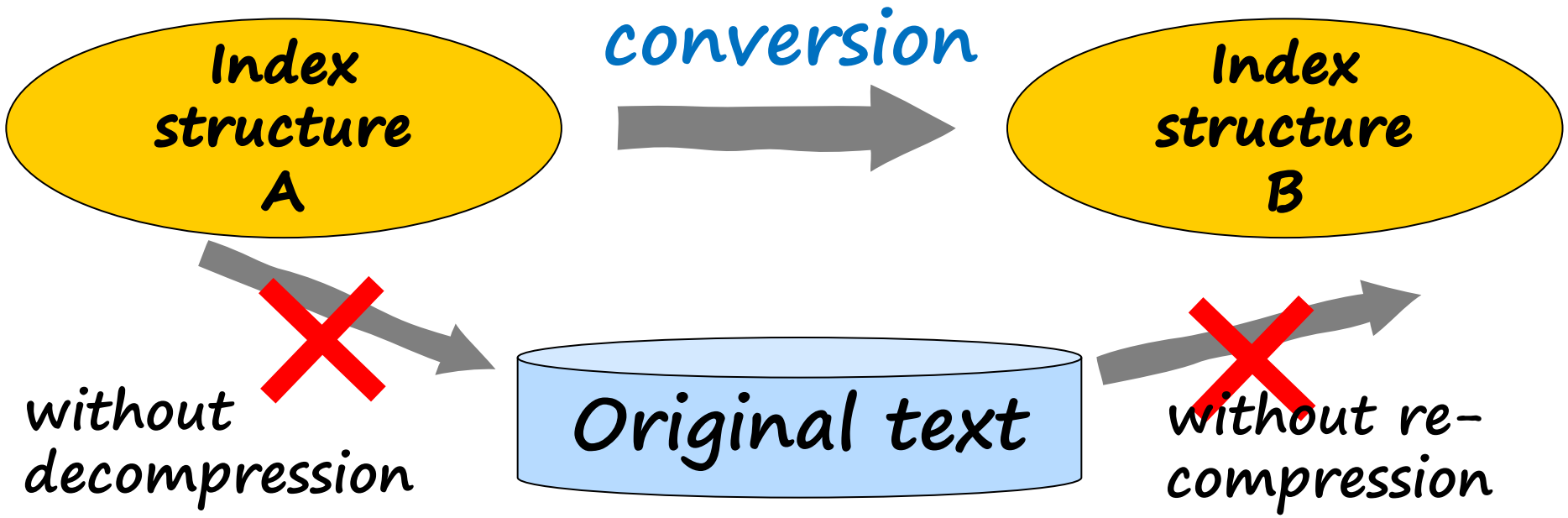
- On the other hand, there are not many researches on the sub-linear complexities of conversions . . .



And many others (see survey by Navarro (CSUR, part ii & ii, '21)

Our Problem: Conversion problem

- Convert a given compressed index A into another compressed index B without decompression
  - We consider the case that A is the CDAWG of a text T
- Our goal: linear time and space in the combined input and output sizes $|A| + |B|$



Related works

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 $O(n / \log_\sigma n + r \text{ polylog } n)$ 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 $O(z \text{ polylog } n)$ 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:

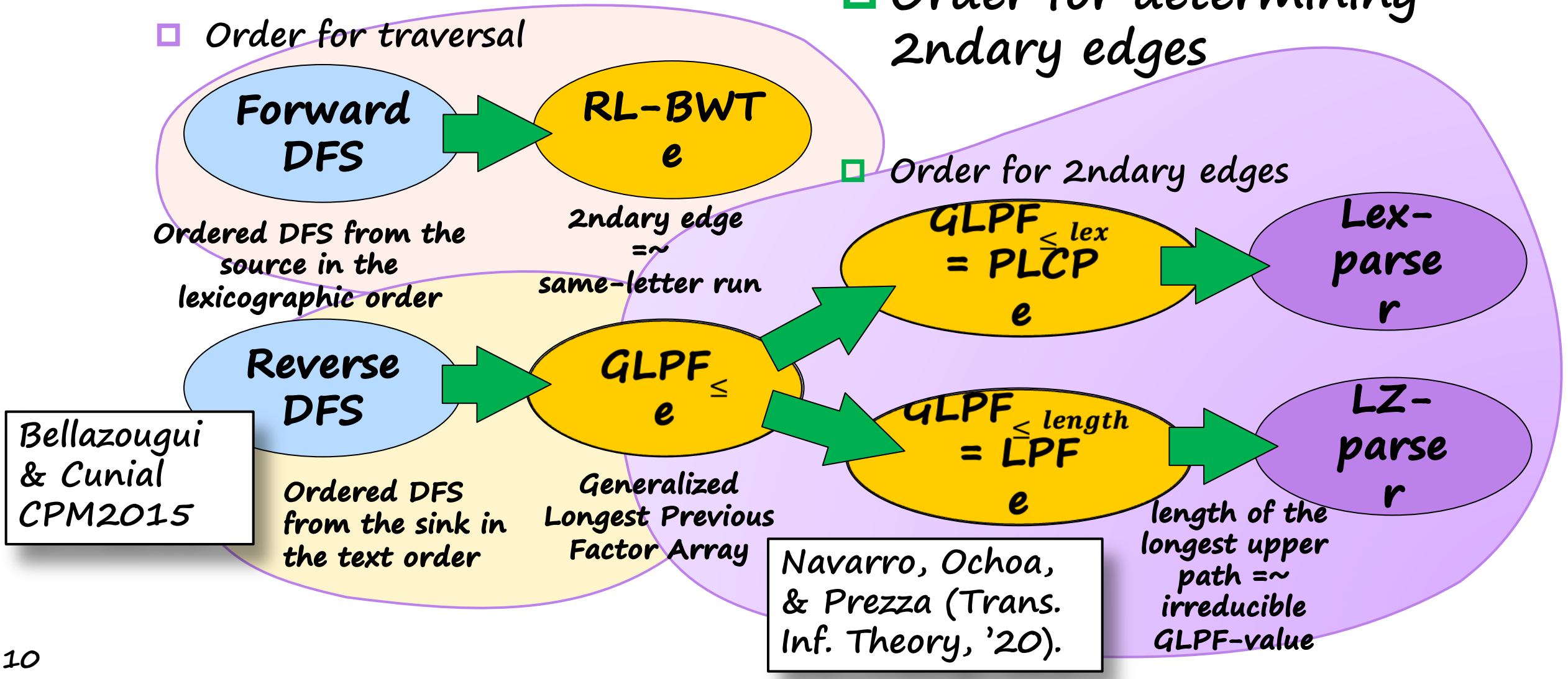
- The RL-BWT (run-length BWT) of size r
- The irreducible PLCP (permuted LCP) array of size r
- The quasi-irreducible LPF (longest previous factor) array of size e (def. Sec. 2 of this paper)
- The Lex-parse of size $2r = O(r)$
- The LZ-parse of size z

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

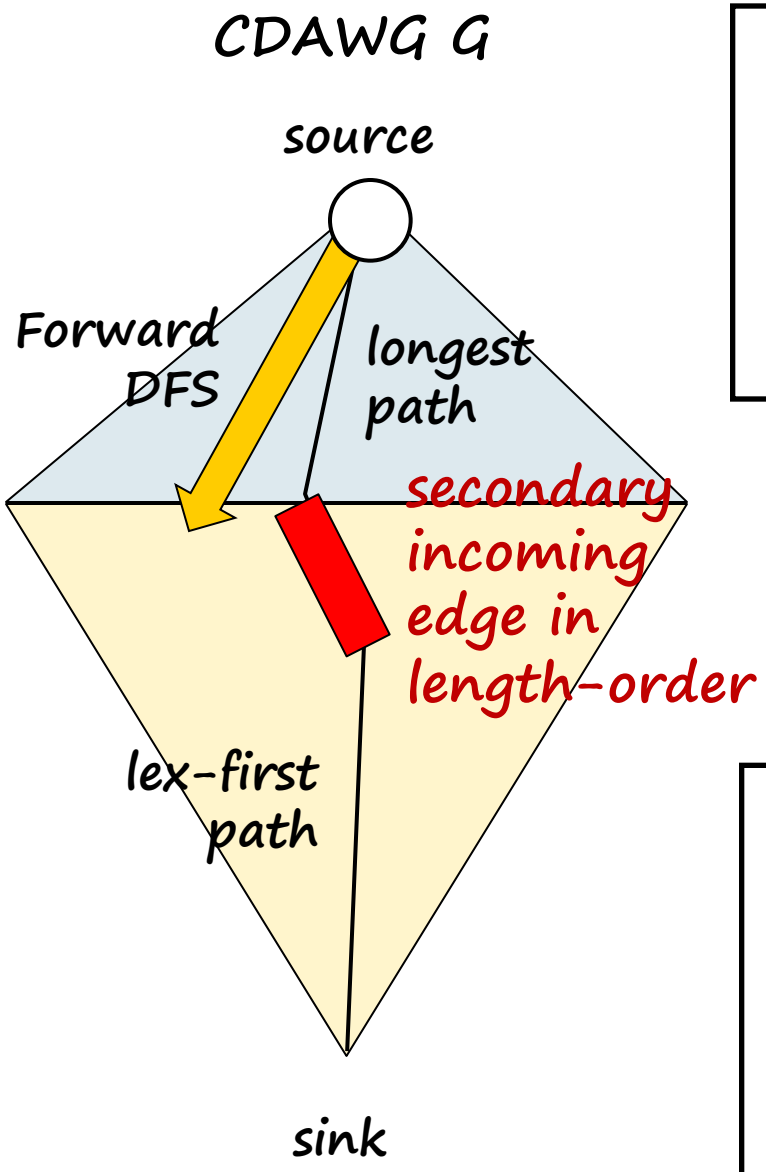
Algorithms

Our approach

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



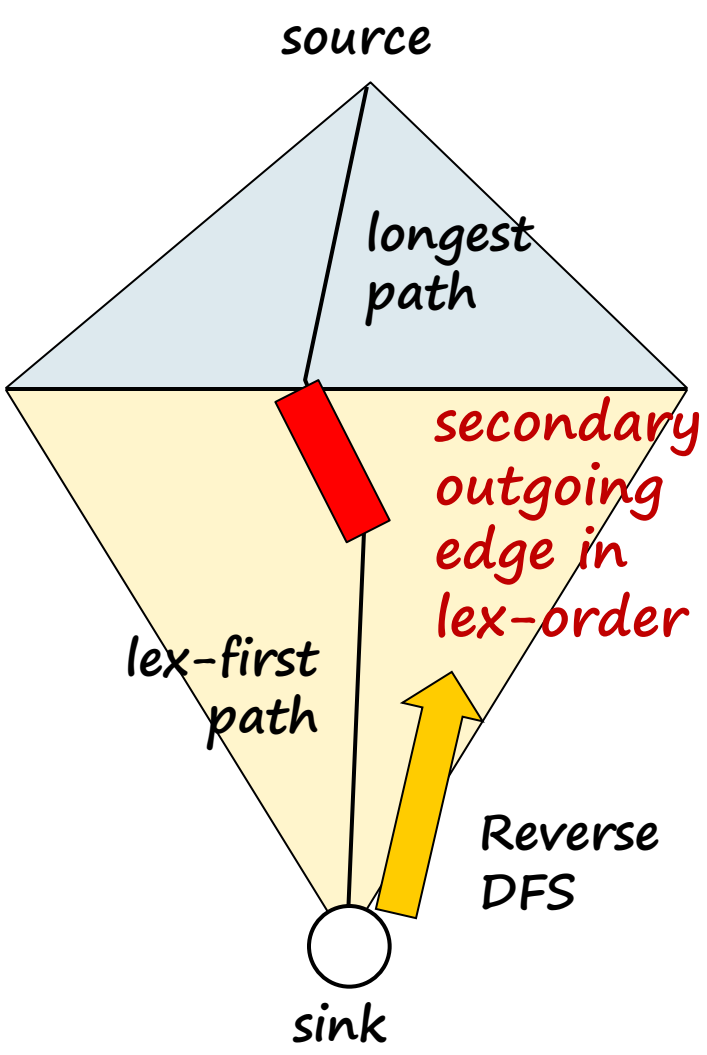
Part A: Computing RL-BWT



- **Observation A1**: Each secondary incoming edge of the CDAWG for T, under length-order, corresponds to (a subinterval) of the same-letter run of the BWT
  - Canonical suffix associated to a secondary incoming edge f
  - = [the longest path from the source to f] . [edge f] . [the lexicographic path from f to the sink]

- **Observation A2**: Such secondary incoming edges can be enumerated in the lexicographic order (of its "canonical suffix") by the forward DFS from the source.

Part B: Computing PLCP



- **Observation A1**: Each secondary outgoing edge of the CDAWG for T, under length-order, determines the branching node v that gives PLCP[p] := max lcp(Tp, Tq) over all Tq such that Tq < Tp
  - in the sense of Navarro, Ochoa, & Prezza (Trans. Inf. Theory, '20).
  - Canonical suffix associated to a secondary outgoing edge f [the longest path from the source to f] . [edge f] . [the lexicographic path from f to the sink]
- **Observation A2**: Such secondary outgoing edges can be enumerated in the text order (of its "canonical suffix") by the reverse DFS from the sink.

Conclusions

- Conversion problem between compressed indices for highly-repetitive texts, when an input is the CDAWG
- O(e) time and space conversion from either the CDAWG of a text T or its self-index into the following structures:
  - RL-BWT, (quasi-) irreducible PLCP and LPF arrays, Lex-parse, and LZ-parse for T.
  - 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:
  - Sub-linear time and space conversion from RL-BWT and LZ-parse into CDAWG.

Thank you!