LOSSLESS FORMATS (FLAC, ETC.)

MARGARET HOPKINS, MUMT 621

WHAT IS A LOSSLESS FORMAT?

- We compress audio to save memory
  → lossy formats like MP3 remove inaudible information – cannot be restored
- Lossless formats allow us to completely reconstruct the original signal
- Typically use more memory than lossy compression

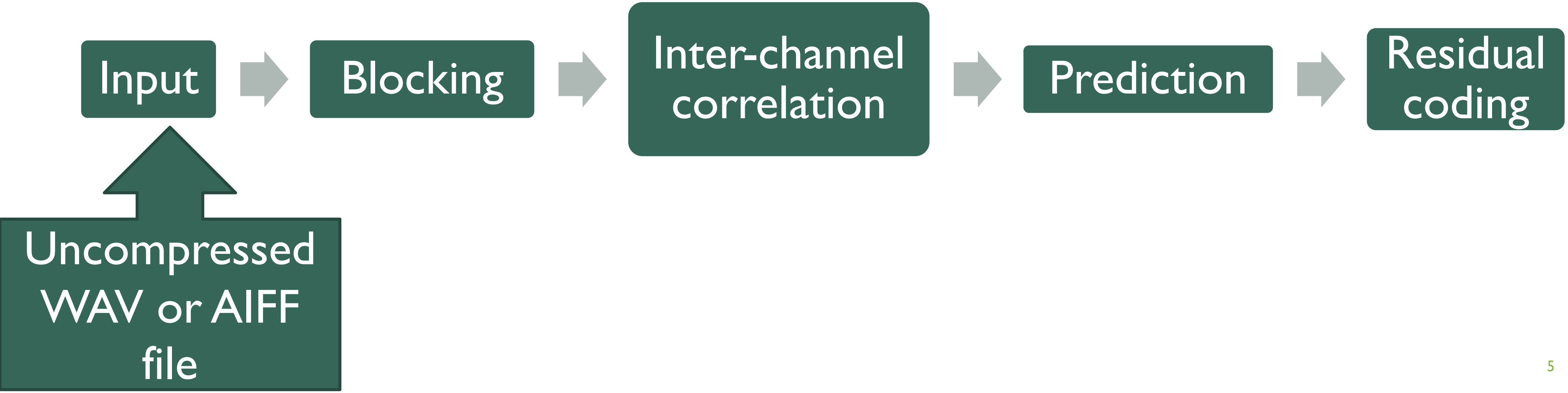
FREE LOSSLESS AUDIO CODEC (FLAC)

- Open format, royalty free compression format
  → compresses to 50-70% of original size
  → all data can be completely recovered from the compressed version
- Fast encoding and decoding, extensive hardware support, and free reference implementation (Rivero & Mishra 2008)

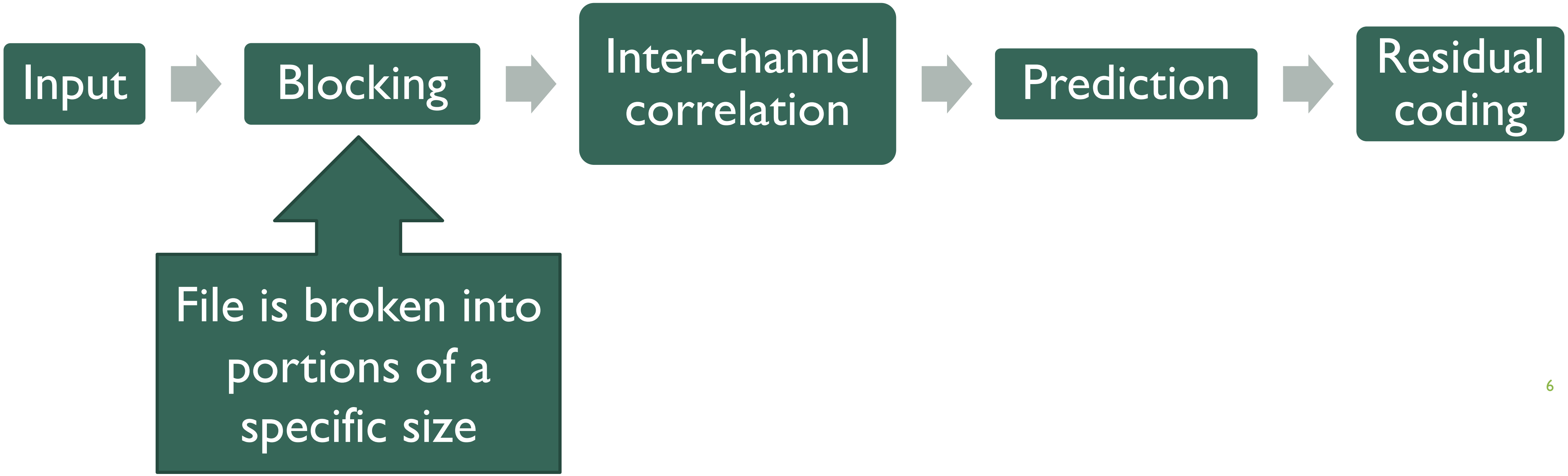
free lossless audio codec

HOW DOES FLAC WORK?

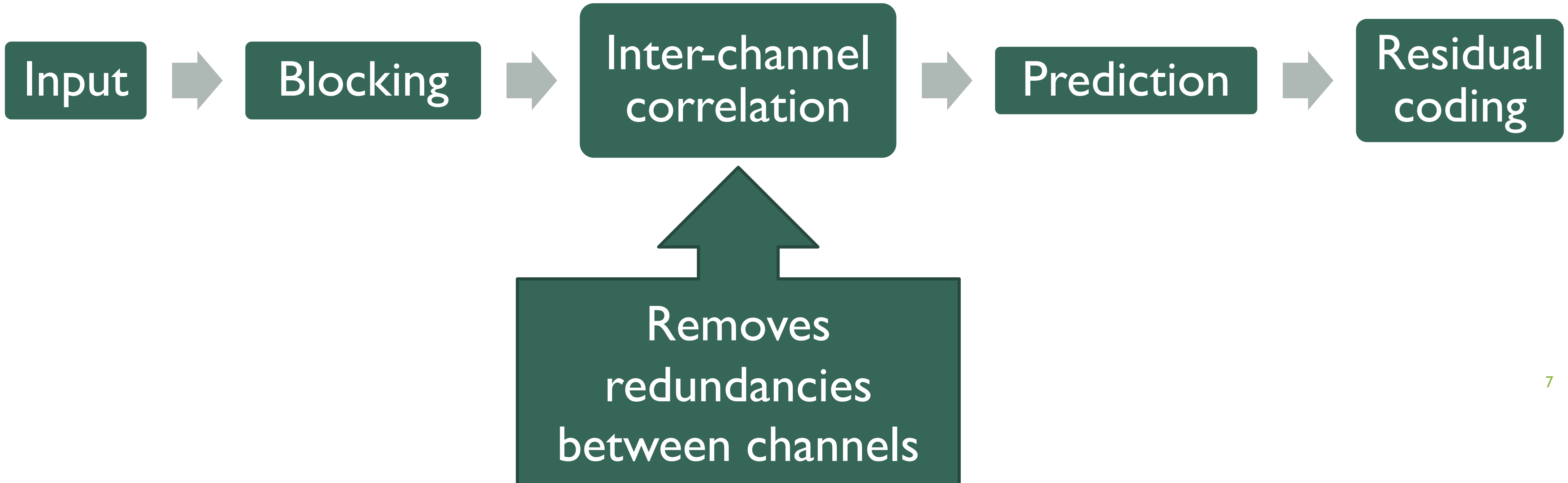
HOW DOES FLAC WORK?



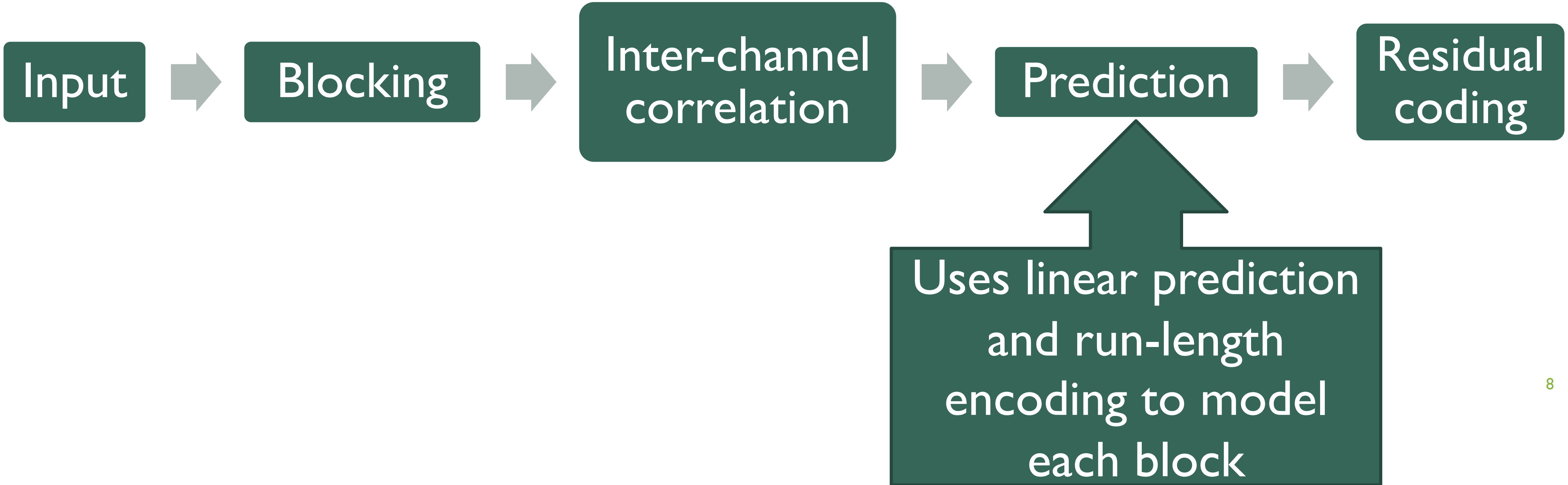
HOW DOES FLAC WORK?



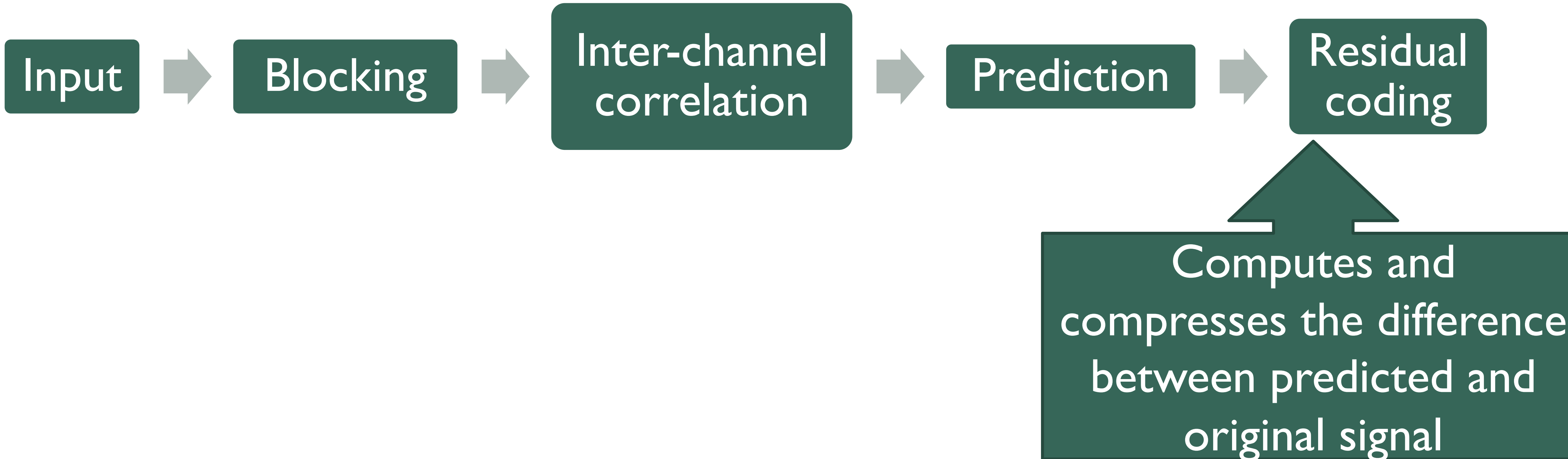
HOW DOES FLAC WORK?



HOW DOES FLAC WORK?

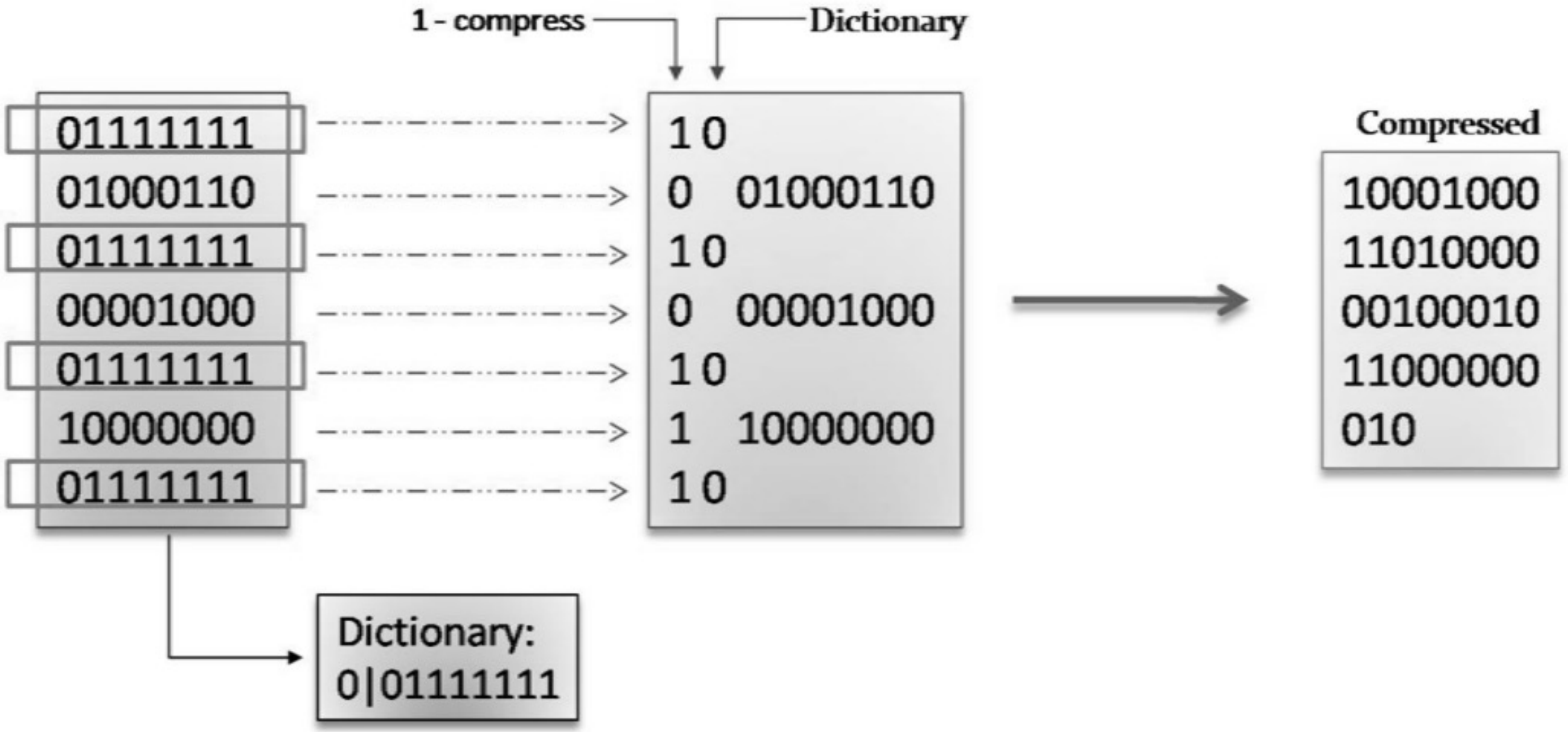


HOW DOES FLAC WORK?



DICTIONARY-BASED COMPRESSION

- Uses a static or frequency-based dictionary → commonly used "words" are stored in dictionary and called by their index



Source: Rivero & Mishra (2008)

GRAMMAR-BASED COMPRESSION

- An algorithm creates a **context free grammar** for the string to be compressed
  → "grammars can capture repetitions occurring far apart in the data," – Cherniavsky & Ladner (2004)
- The grammar replaces substrings with their references in a combination that produces the smallest possible encoding (Humphries, Sidorov, Jones, & Marshall 2021)

GRAMMAR-BASED COMPRESSION

Source: Cherniavsky & Ladner (2004)

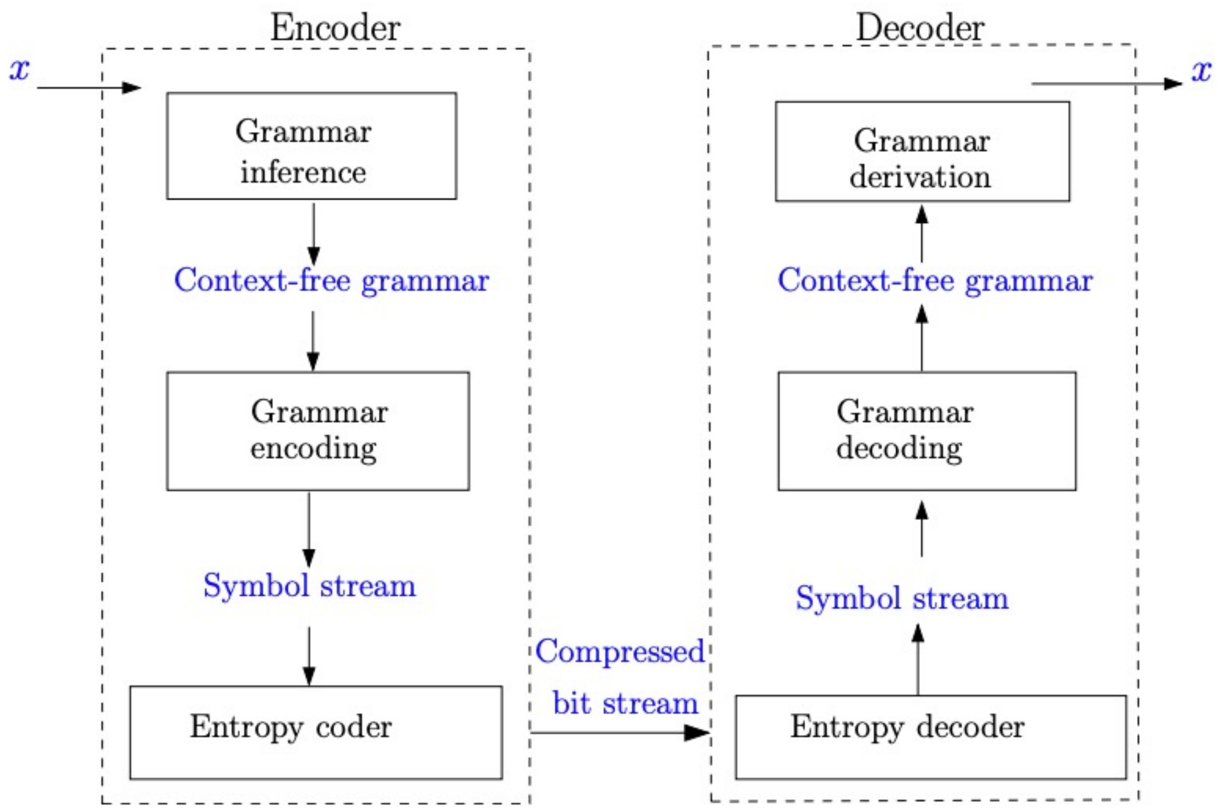


Figure 1: Overview of grammar compression

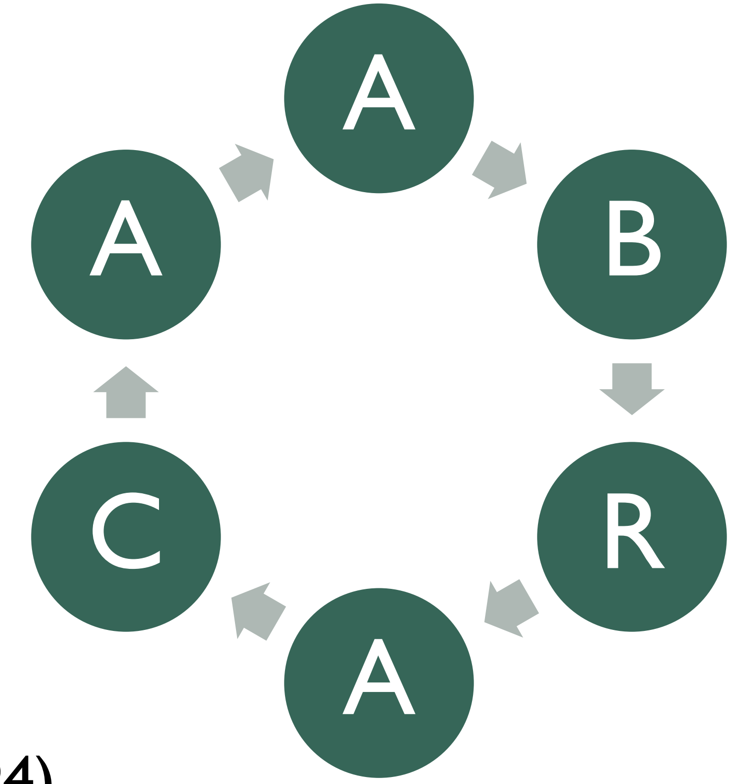
BURROWS-WHEELER TRANSFORM

- Proposed in 1994 as a method to prepare data for compression
- Reorders a string of characters so that there are more segments with repeated characters
  → makes the string more suitable for run-length and move-to-front encoding
- Can be reversed to restore the original string

Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM - EXAMPLE

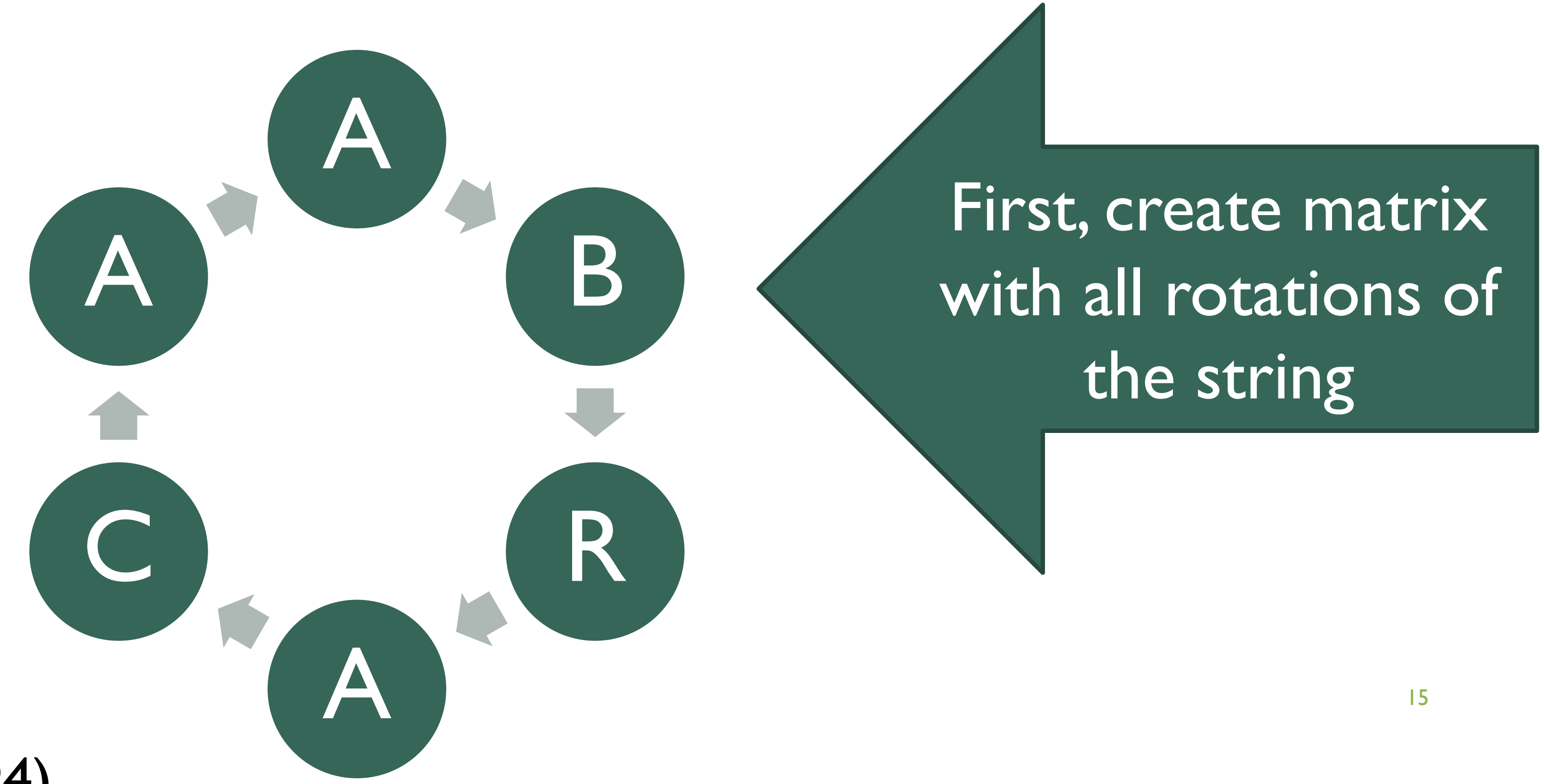
Consider the string "ABRACA":



Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM - EXAMPLE

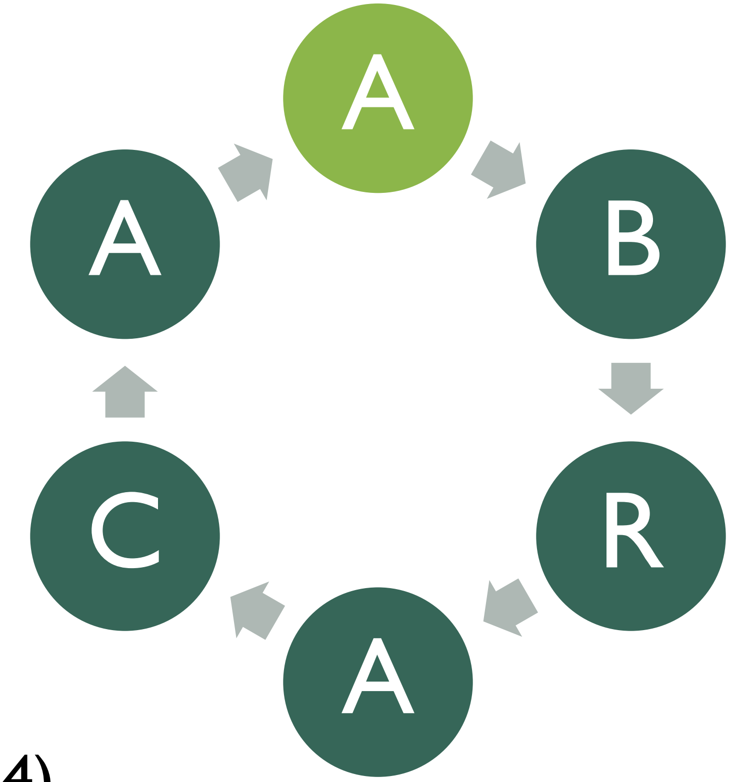
Consider the string “ABRACA”:



Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM - EXAMPLE

Consider the string "ABRACA":

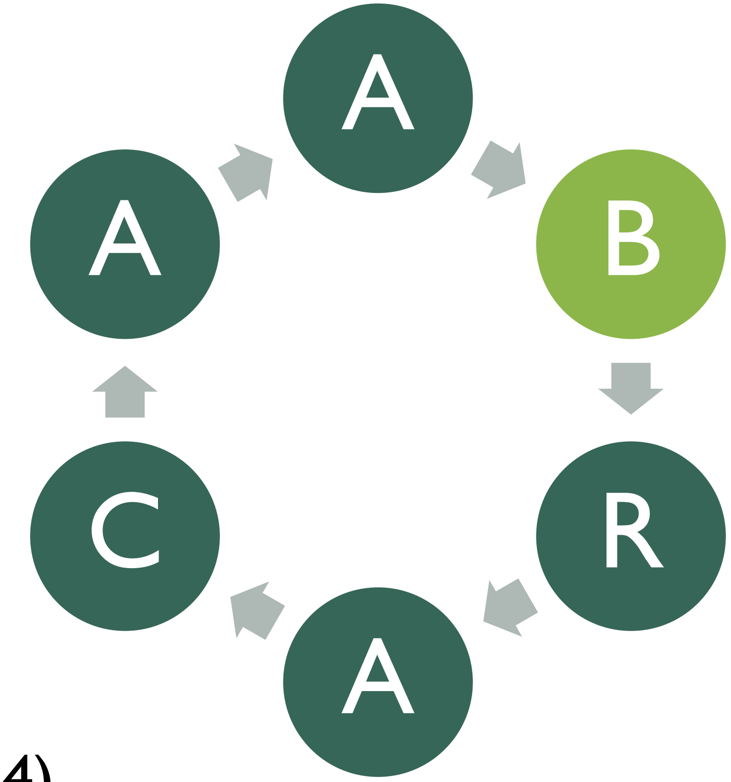


ABRACA

Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM - EXAMPLE

Consider the string “ABRACA”:

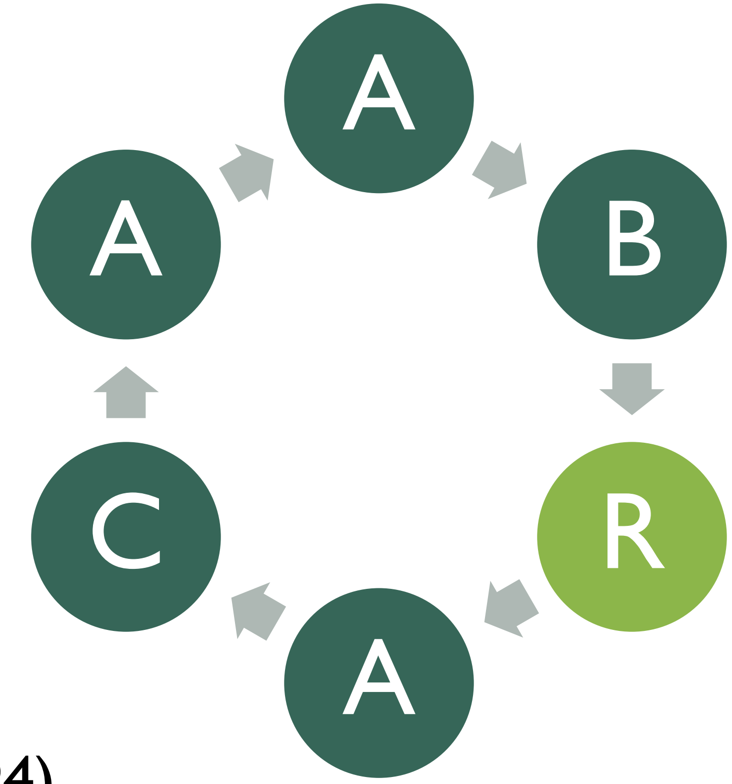


ABRACA
BRACAA

Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM - EXAMPLE

Consider the string "ABRACA":

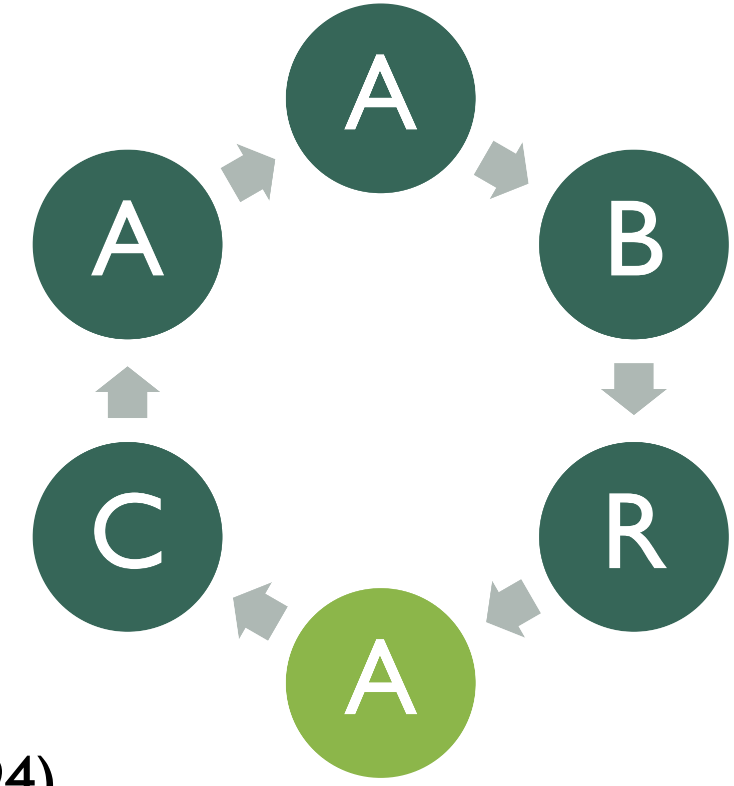


ABRACA
BRACAA
RACAAB

Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM - EXAMPLE

Consider the string "ABRACA":

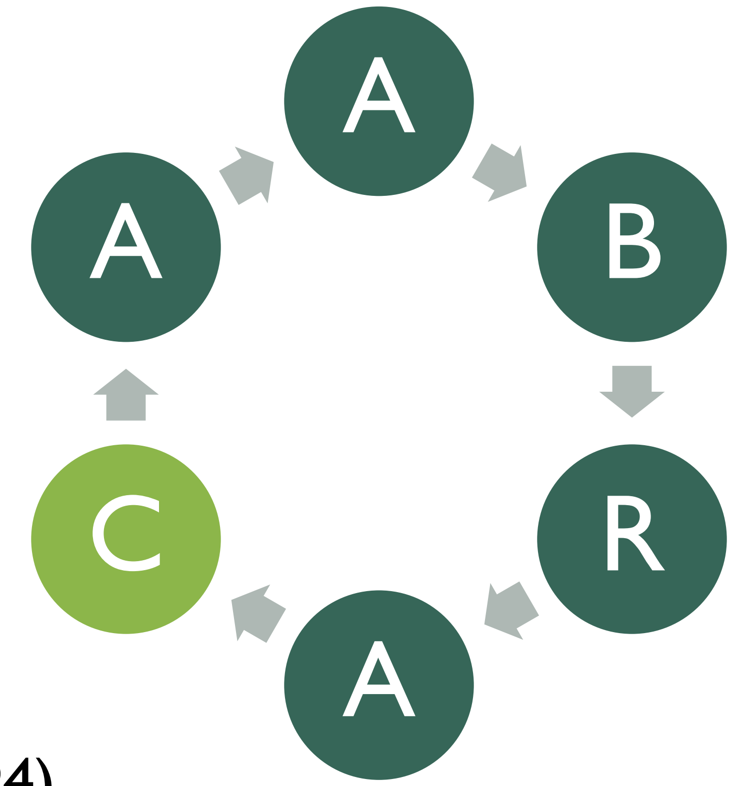


ABRACA
BRACAA
RACAAB
ACAABR

Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM - EXAMPLE

Consider the string "ABRACA":

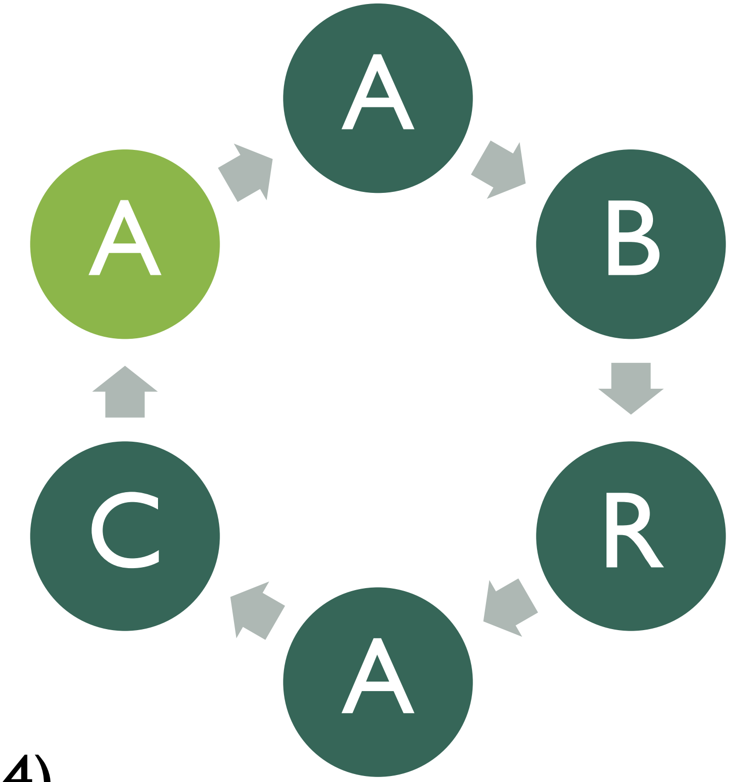


ABRACA
BRACAA
RACAAB
ACAABR
CAABRA

Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM - EXAMPLE

Consider the string "ABRACA":



ABRACA
BRACAA
RACAAB
ACAABR
CAABRA
AABRAC

Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM - EXAMPLE

Next, we reorder the matrix lexographically, noting the index of the row with the original string

ABRACA	→	AABRAC
BRACAA		ABRACA
RACAAB		ACAABR
ACAABR		BRACAA
CAABRA		CAABRA
AABRAC		RACAAB

Index = 2

Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM - EXAMPLE

The Burrows-Wheeler transform of of the string is the last column of the new matrix

ABRACA	→	AABRAC
BRACAA		ABRACA
RACAAB		ACAABR
ACAABR		BRACAA
CAABRA		CAABRA
AABRAC		RACAAB

Index = 2

Source: Burrows & Wheeler (1994)

BURROWS-WHEELER TRANSFORM

ABRACA → CARAAB

The resulting string has more runs of repeated characters, especially if there were repeated words in the input

LOSSLESS IMAGE/VIDEO COMPRESSION

- Other types of media can also be compressed into lossless formats:
  → photos: PNG, GIF, TIFF
  → videos: H.264 Lossless, H. 265 Lossless, Motion JPEG Lossless, Apple Animation Quicktime RLE

APPLICATIONS TO MIR

ARCHIVAL PURPOSES

- Cooper (2020)
  → archiving a large collection donated to the University of Leeds by Trevor Jones
  → digitized recordings and stored using FLAC in order to retain all information while saving memory
- Lai, Li, & Fujinaga (2005)
  → digitizing album covers from David Edelberg's record collection
  → tested two lossless image compression formats: TIFF and PNG
  → chose PNG for significant file size reduction, open format, and good metadata

HUMPHREYS, SIDOROV, JONES, & MARSHALL (2021)

- Used grammar-based compression to perform musicological tasks
  → detecting transcription errors
  → classifying pieces by melodic characteristics
  → segmenting pieces for musical analysis
  Dataset: 7928 scores from Acadia Early Music Archive, CPDL and Musopen
- Used string processing techniques designed for compression and pattern matching within text
- Algorithms used: Lempel-Ziv Welch, Burrows-Wheeler with run-length encoding, and GZIP

HUMPHREYS, SIDOROV, JONES, & MARSHALL (2021)

- Error detection: compared size of compressed original data and altered data
  → if the size was different, there was an alteration
  → all algorithms could detect a single error
  → logarithmic response to increasing number of errors
- Classification: used compression distance between two scores to rate their similarity
  → used the Meertens Tune Collections
  → compressing scores with common components produces a smaller model
  → success rates: 0.92 for ZZ and 0.83 for IIR-MC

HUMPHREYS, SIDOROV, JONES, & MARSHALL (2021)

- Segmentation: hypothesized that using a grammar-based compressor would divide the data in a musicologically significant way
  → ZZ performed well on pieces containing exact matches, like Bach's Fugue no. 20 from the Well-Tempered Clavier
  → strong correlation between rules from the grammar-based compressors and analysis by Bruhn (1993)

CONCLUSIONS

- Lossless compression allows us to store audio and visual data with reduced memory use without losing any information
- Compression algorithms can also be used for pattern recognition that can be applied to structural music analysis, classification, and error detection

REFERENCES

Burrows, M. and D. J. Wheeler. 1994. "A Block-sorting Lossless Data Compression Algorithm." *Systems Research Center Research Report*, May 10, 1994. https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=1E68984AB08B8212758B54156463A6B5?doi=10.1.1.37.6774&rep=rep1&type=pdf.

Cherniavsky, Neva and Richard Ladner. 2004. "Grammar-based Compression of DNA Sequences." *University of Washington Computer Science & Engineering Technical report*, May 28, 2004. https://dada.cs.washington.edu/research/tr/2007/05/UW-CSE-07-05-02.pdf.

Cooper, David. 2020. "The Trevor Jones Archive: Issues in the Establishment and Management of a Film and Television Music Archive." *Sources and Research from the Institute of Music* no. 5: 101–113. http://onlinepublishing.cini.it/index.php/arno/article/view/175/282.

Humphreys, David, Kirill Sidorov, Andrew Jones, and David Marshall. 2021. "An investigation of music analysis by the application of grammar-based compressors." *Journal of New Music Research* 50 no. 4: 312–341. https://doi.org/10.1080/09298215.2021.1978505.

Lai, Catherine, Beinan Li, and Ichiro Fujinaga. 2005. "Preservation digitization of David Edelberg's Handel LP Collection: A Pilot Project." In *Proceedings of the 6th International Conference on Music Information Retrieval*, 570–575. Queen Mary, University of London. https://ismir2005.ismir.net/proceedings/1090.pdf.

Rivero, Cristobal and Prabhat Mishra. 2008. "Lossless Audio Compression: A Case Study." *Computer Information Science & Engineering Technical Report 08-415*, August 7, 2008. https://esl.cise.ufl.edu/Publications/audioTR.pdf.