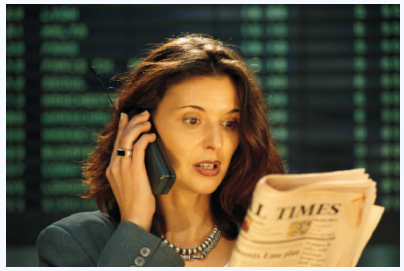
# Project Group
# Text Search Engines

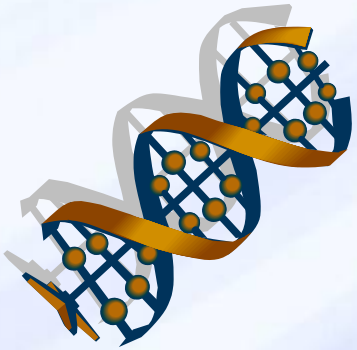
Stefan Böttcher

# Text Search Engines for Big Text Data

Company-wide and beyond



Genom data



Index large text collections
→ where does a keyword / phrase occur

# Search engines

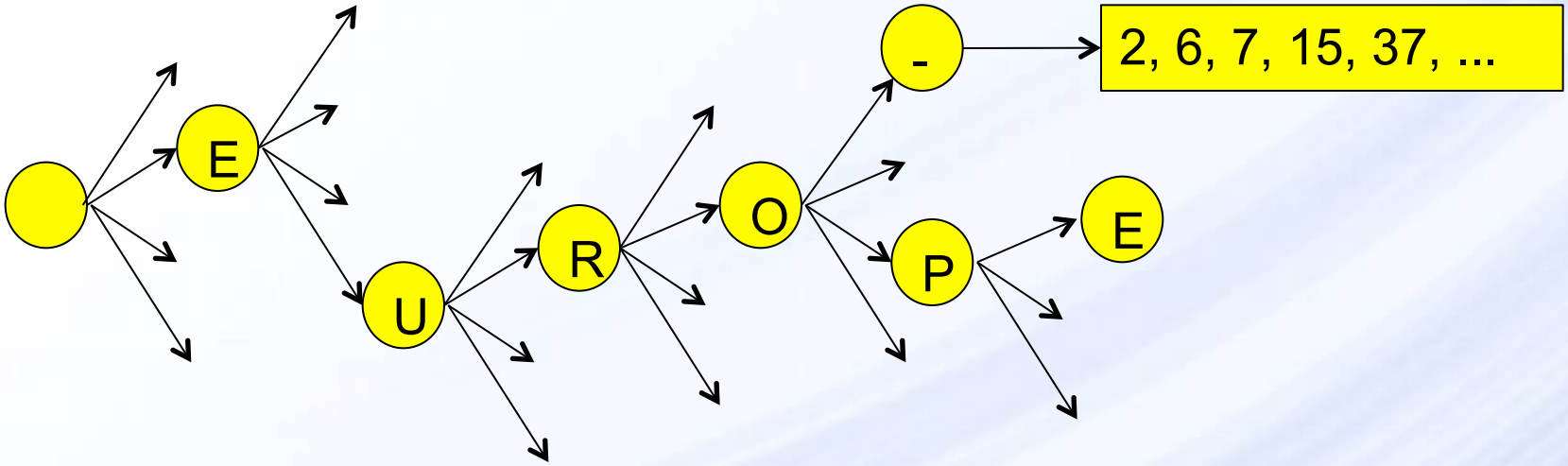
Index large text collections
→ where occurs a keyword / phrase

Challenge: huge collections
→ may not fit into main memory
→ compression of index and/or text required

one possible solution:
→ PG

# Search engines

need data structure (=Index) for fast access on Big Data



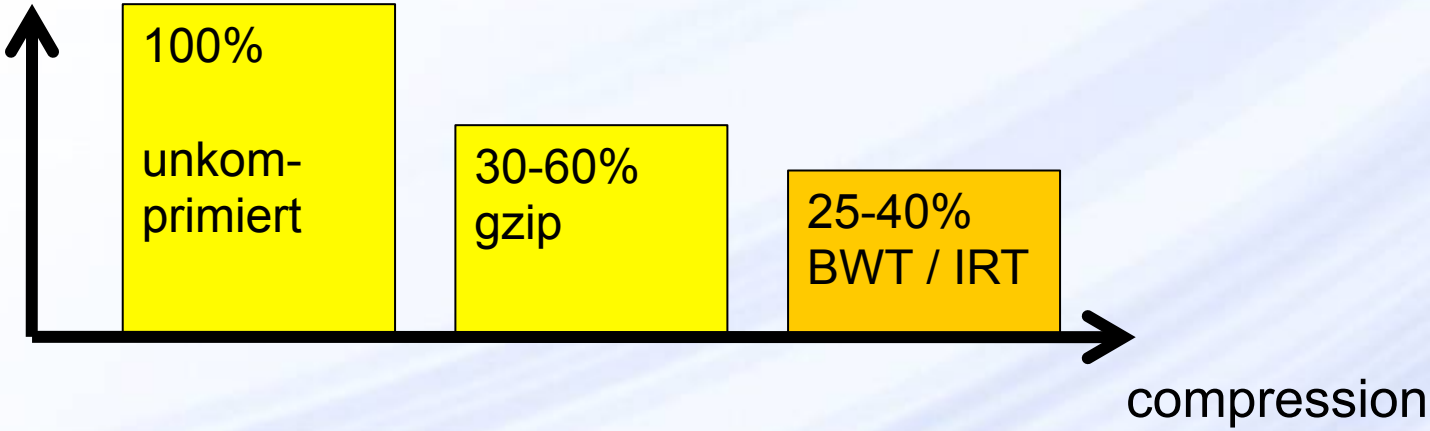

Challenge: Index needs Sorting of Big Data

# upb-approach (IRT) filed for patent application

## IRT – Indexed Reversible Transformation

- based on Burrows Wheeler Transformation
- supports **search operations / modifications** without retransformation

costs for data volume / energy / ...

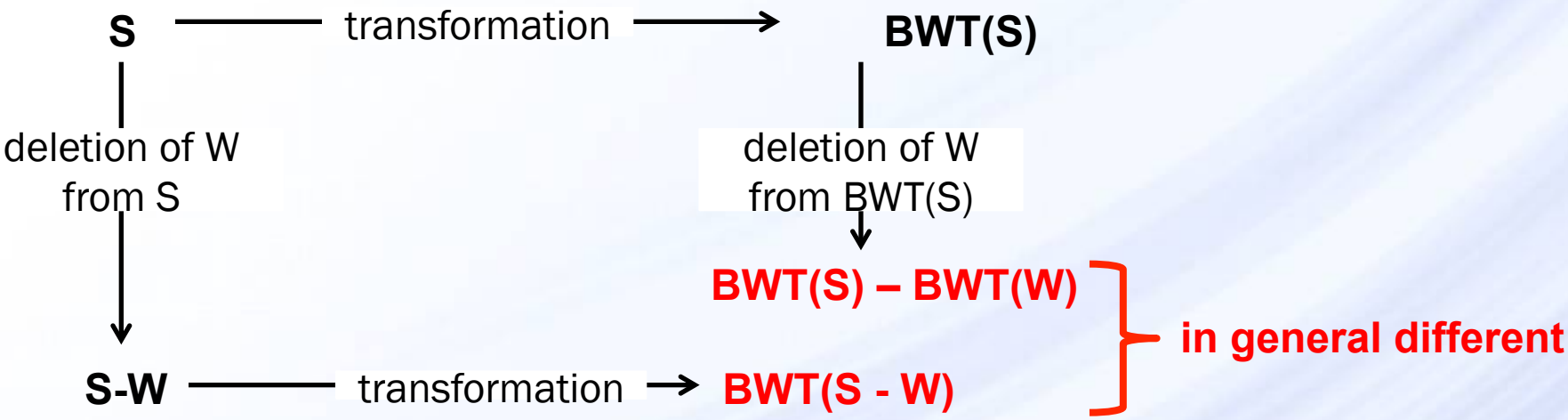



Data Compression Conference (DCC), 2011

# Difference IRT←→BWT: deletion and BWT do not commute

Even if the position of the first/last letter of the word to be deleted is known (e.g. by an index), deletion of a word and transformation by BWT do not commute

Let S be a text and W an arbitrary word of S, then



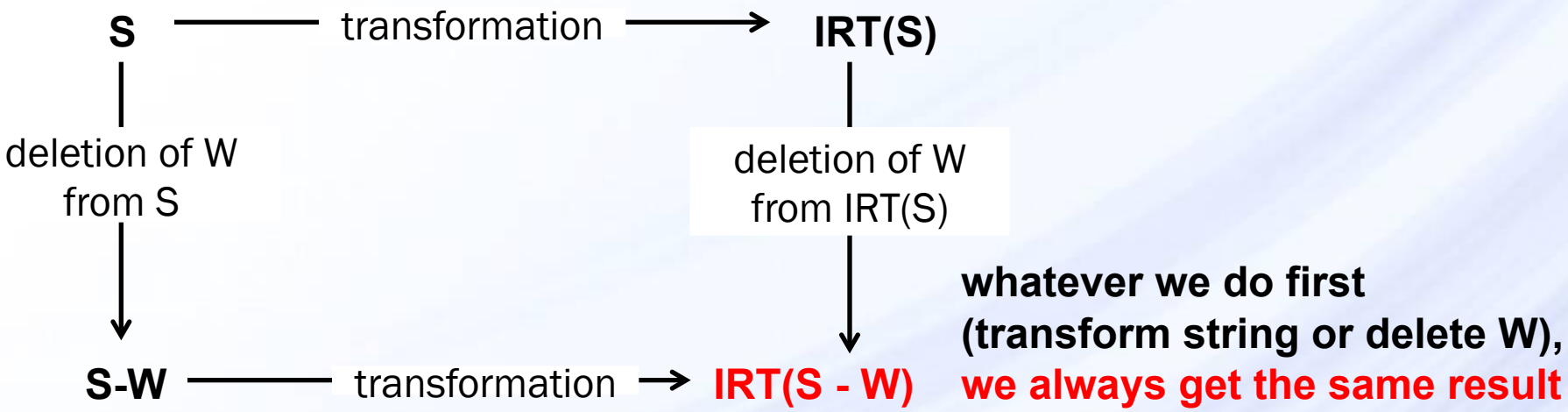

**BWT(S) – BWT(W) ≠ BWT(S-W)**

i.e. in contrast to IRT, a word cannot be deleted from BWT(S) without retransformation of BWT(S) to S.

# Difference IRT←→BWT: but deletion and IRT commute

However, deletion of a word and transformation by IRT commute

Let S be a text and W an arbitrary word of S, then



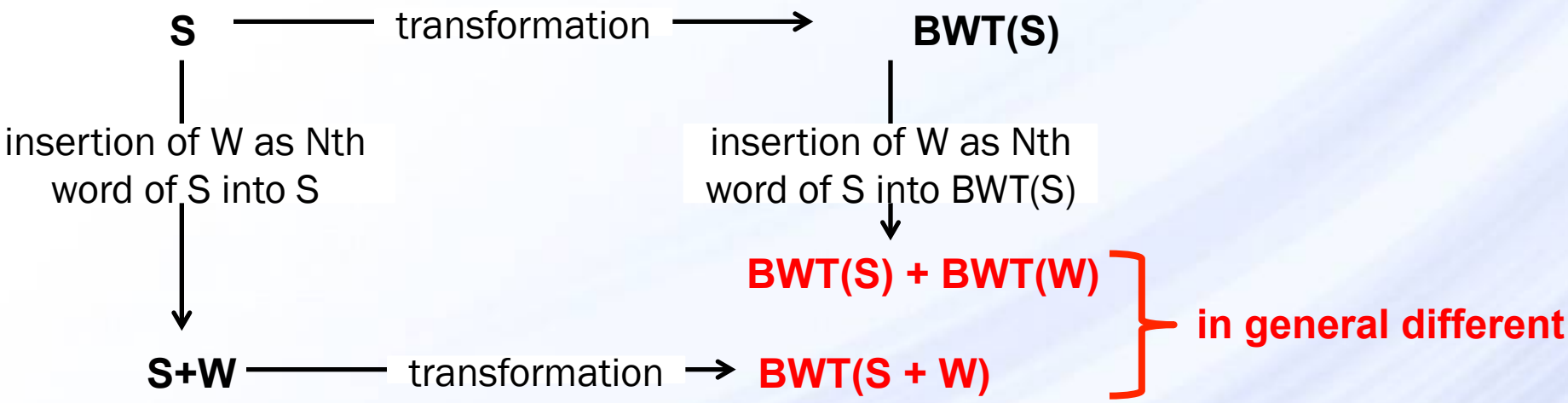

**IRT(S) – IRT(W) = IRT(S-W)**

i.e. in contrast to BWT(S), a word in IRT(S) can be deleted from IRT(S) without retransformation of IRT(S) to S.

# Difference IRT←→BWT: insertion and BWT do not commute

Even if the position of the first/last letter of the word to be inserted is known (e.g. by an index), insertion of a word and transformation by BWT do not commute

Let S be a text and W an arbitrary word of S, then



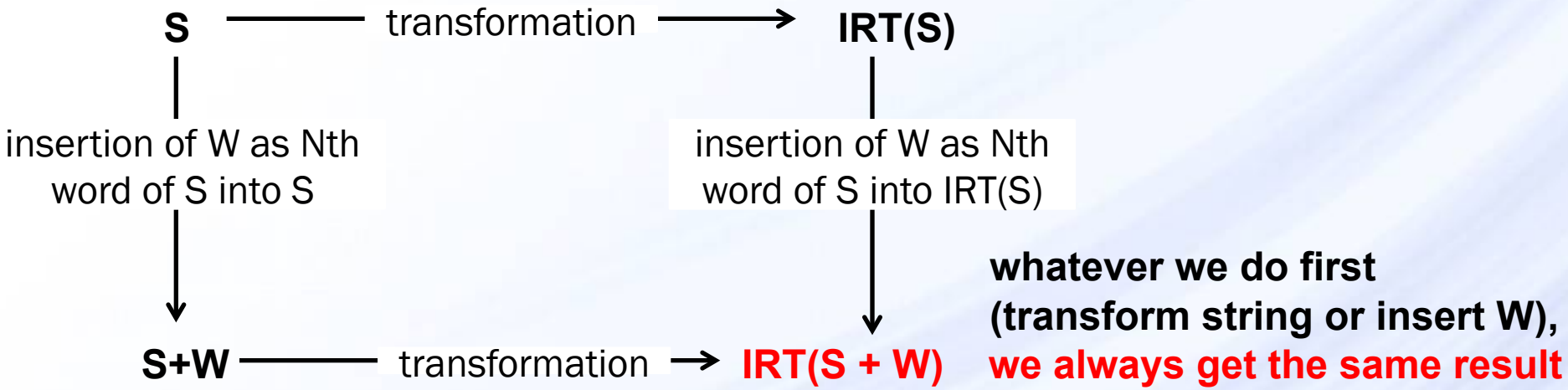

**BWT(S) + BWT(W) ≠ BWT(S+W)**

i.e. in contrast to IRT, a word cannot be inserted into BWT(S) without retransformation of BWT(S) to S.

# Difference IRT←→BWT: but insertion and IRT commute

Insertion of a word and transformation by BWT commute

Let S be a text and W an arbitrary word of S, then



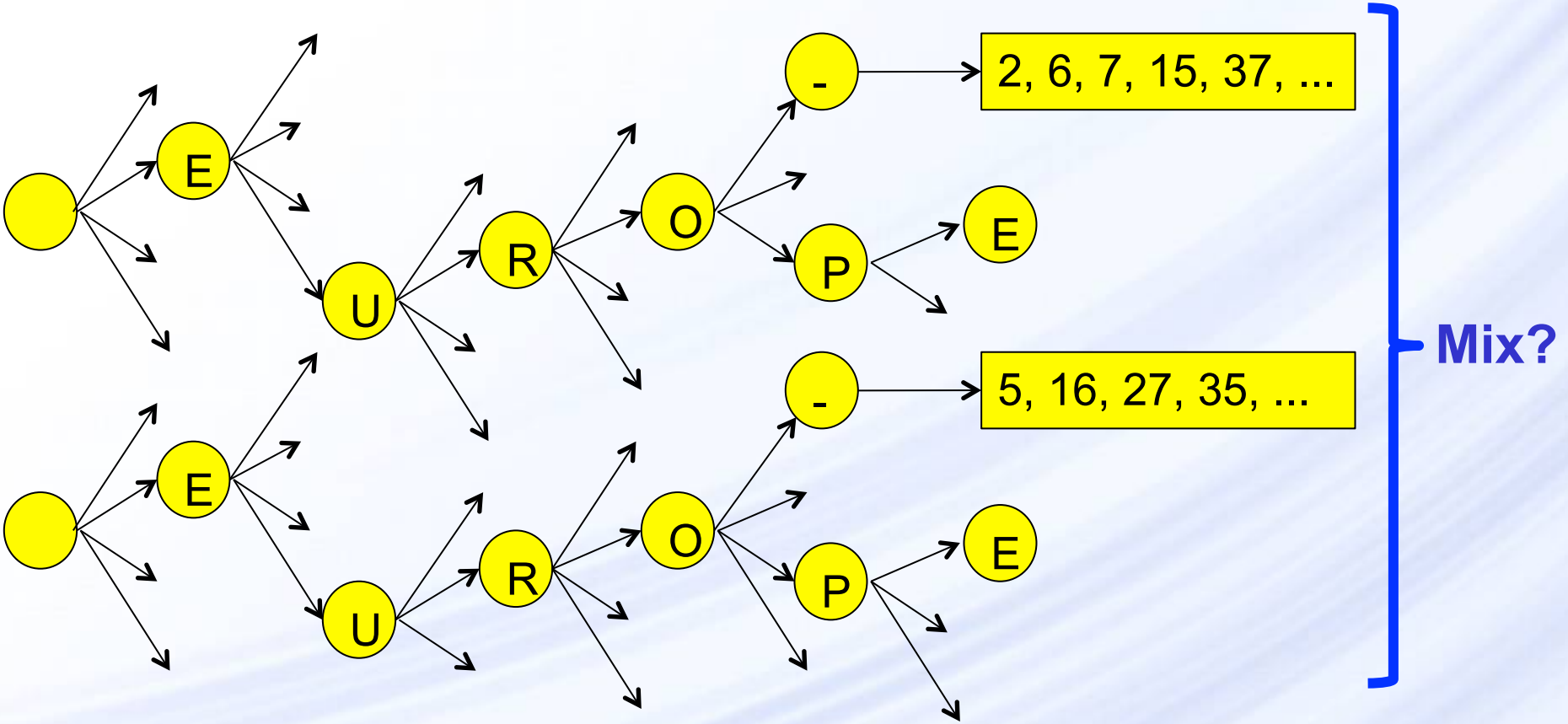

**IRT(S) + IRT(W) = IRT(S+W)**

i.e. in contrast to BWT, a word can be inserted into IRT(S) without retransformation of IRT(S) to S.

# Desired properties to be supported on IRT

given a sequence S of words,

- parallel construction of IRT(S) for large S – how to do that?



# Parallel construction of an Index?

How to increment
a pre-sorted index?

Fast insert is up to **120x** faster than Standard Insertion

How can I know
where to insert chars?

DEXA 2013

# PG Organization

**First meeting (kick-off) for the project group:**
Tuesday 23.9.2014 9-11 c.t. , F2.211

**Meeting times of the project group for regular meetings during the winter term:**
Thursday 9-11 , Friday 9-11 + on demand

**Prerequisites:**
Knowledge of Java programming (i.e., arrays, trees, threads)

**Contact:**
Prof. Dr. Stefan Böttcher , stb@upb.de

# Join us!


If you want to join,
please **register in PAUL** and **send an email to stb@upb.de containing**

1. your contact data, (name, email, phone, ...)
2. a short statement of why you like to join the project group and
3. optionally, i.e., if you want, the grades of master courses taken so far