Relative Select

Jouni Sirén
Wellcome Trust Sanger Institute

with

Christina Boucher, Alexander Bowe, Travis Gagie,
and Giovanni Manzini

Relative data structures

- Relative data compression encodes dataset y relative to dataset x as y|x.

- If x and y are similar in an obvious way, the similarities between f(x) and f(y) should also be obvious, for most reasonable functions f.

- Most data structures D are reasonable functions.

- We should be able to encode D(y) relative to D(x) as D(y|x), and simulate D(y) with D(x) and D(y|x).

An FM-index is a space-efficient full-text index with similar functionality as the suffix array. It is based on rank() queries on the Burrows-Wheeler transform of the text.

BWT.rank(i,c) – the number of c's in BWT[1,i]

We can represent the FM-index of a target sequence S relative to the FM-index of a reference sequence R. This approach is practical for individual human genomes.

Belazzougui, Gagie, Gog, Manzini, and Sirén: **Relative FM-indexes**. SPIRE 2014.

Relative FM-index

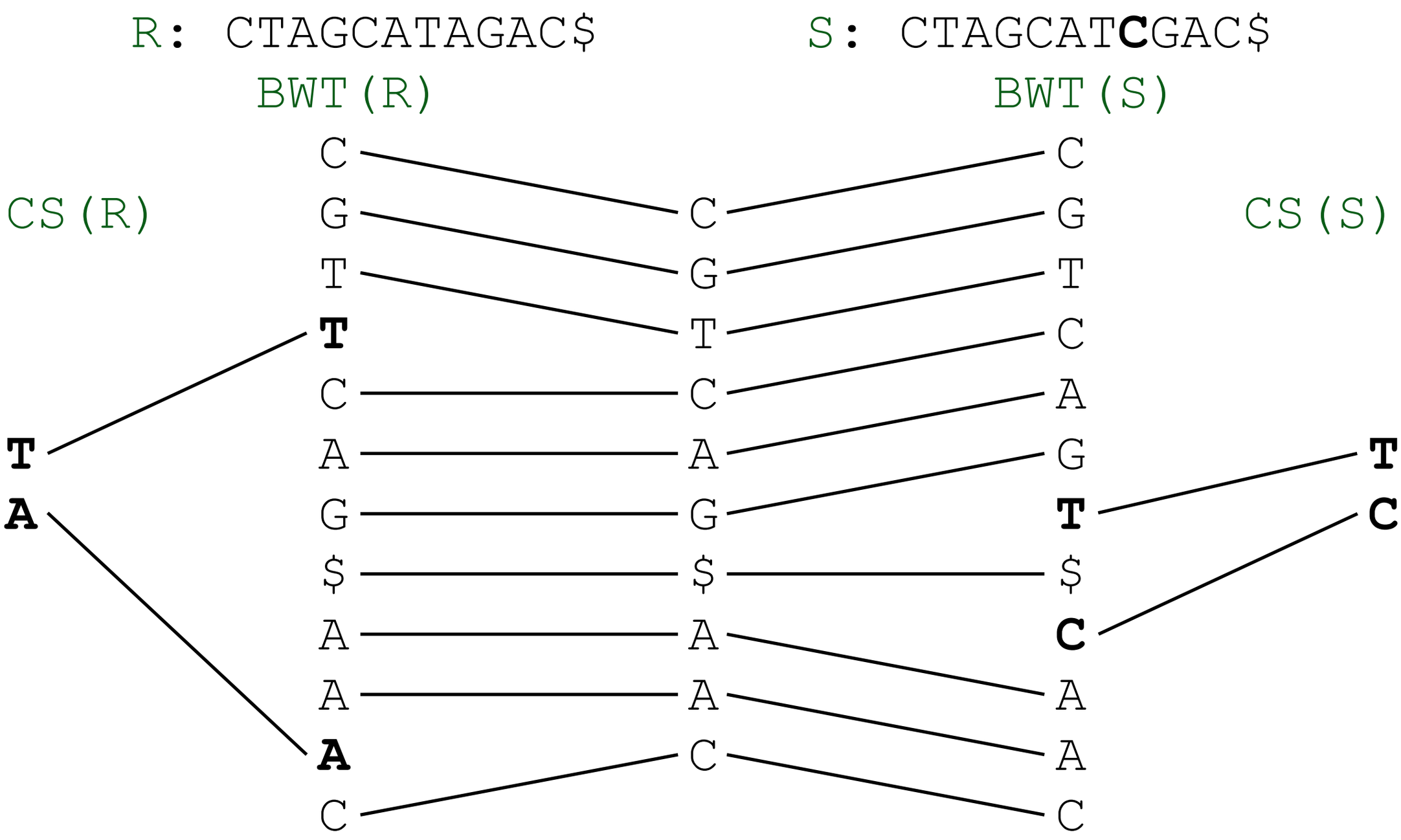
R: CTAGCATAGAC$

$	C
AC$	G
AGAC$	T
AGCATAGAC$	T
ATAGAC$	C
C$	A
CATAGAC$	G
CTAGCATAGAC$	$
GAC$	A
GCATAGAC$	A
TAGAC$	A
TAGCATAGAC$	C

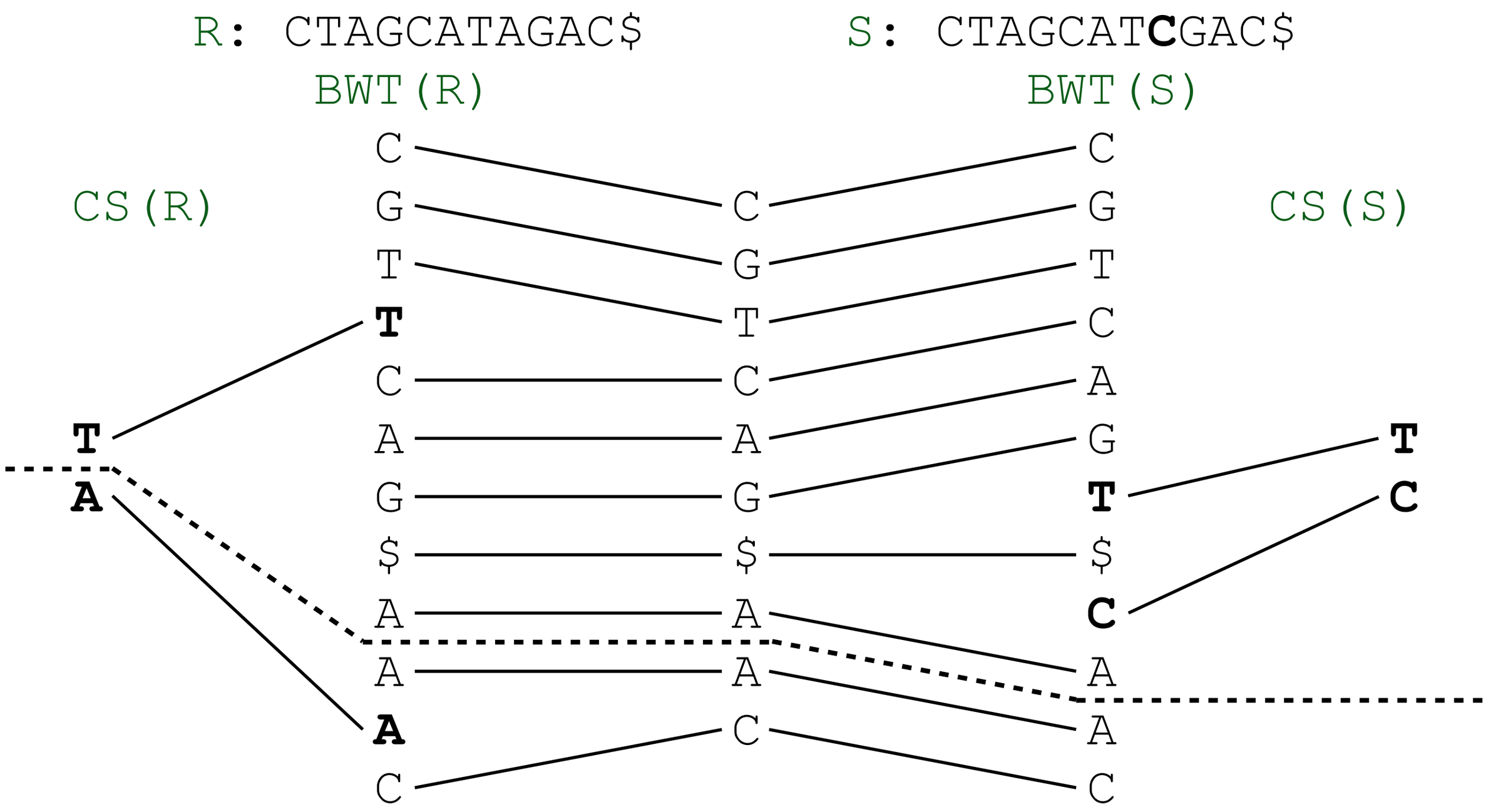
S: CTAGCAT**C**GAC$

$	C
AC$	G
AGCAT**C**GAC$	T
AT**C**GAC$	C
C$	A
CAT**C**GAC$	G
CGAC$	T
CTAGCAT**C**GAC$	$
GAC$	**C**
GCAT**C**GAC$	A
T**C**GAC$	A
TAGCAT**C**GAC$	C

Relative FM-index



Relative FM-index



BWT(S).rank = BWT(R).rank – CS(R).rank + CS(S).rank

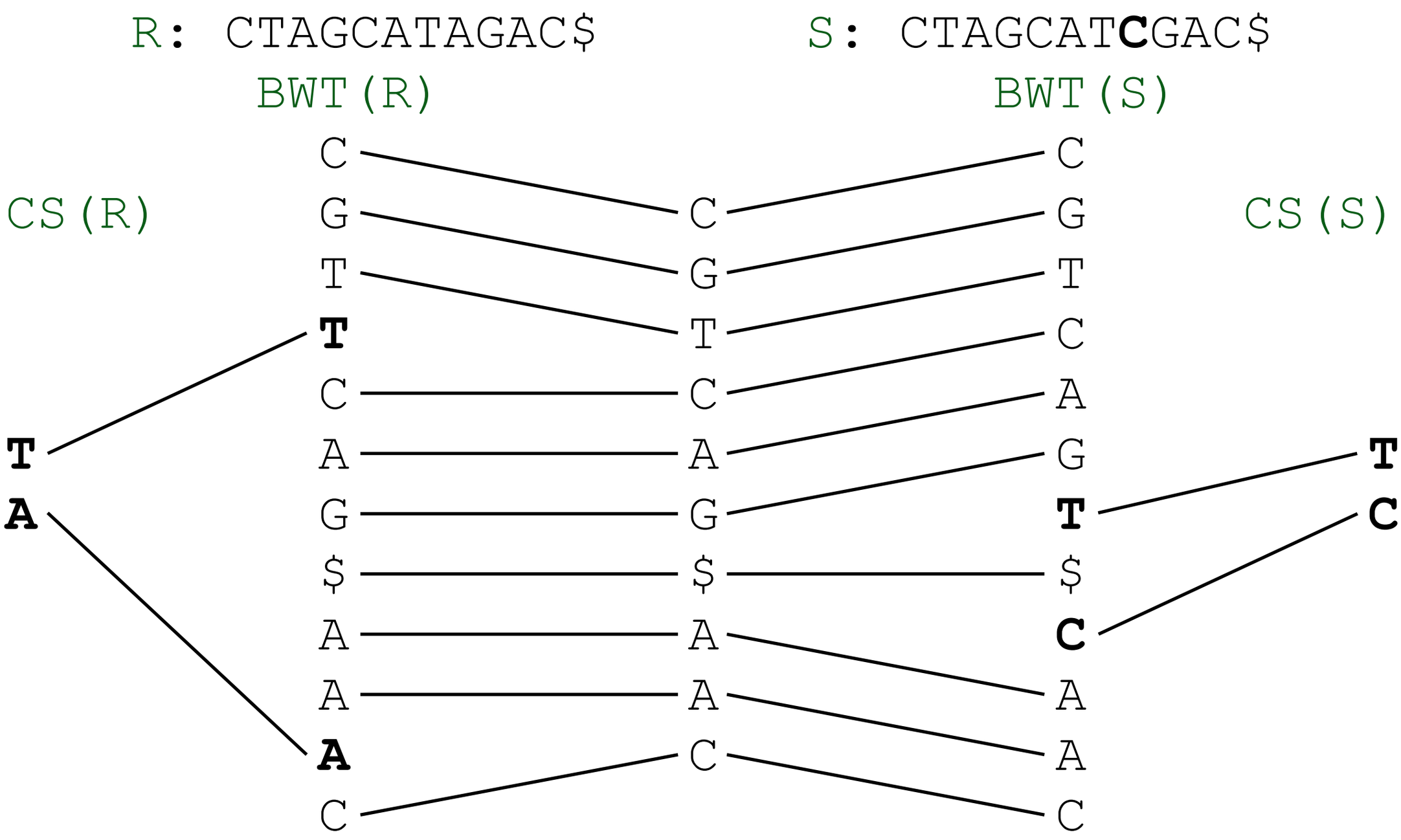
We can move backward in the text with rank() queries on the BWT. To move forward, we need select() queries.

BWT.select(i,c) – the occurrence of c with rank i

Forward movement can be useful, when the FM-index is a part of a compressed suffix tree, or when we are using BWT-based de Bruijn graph representations.

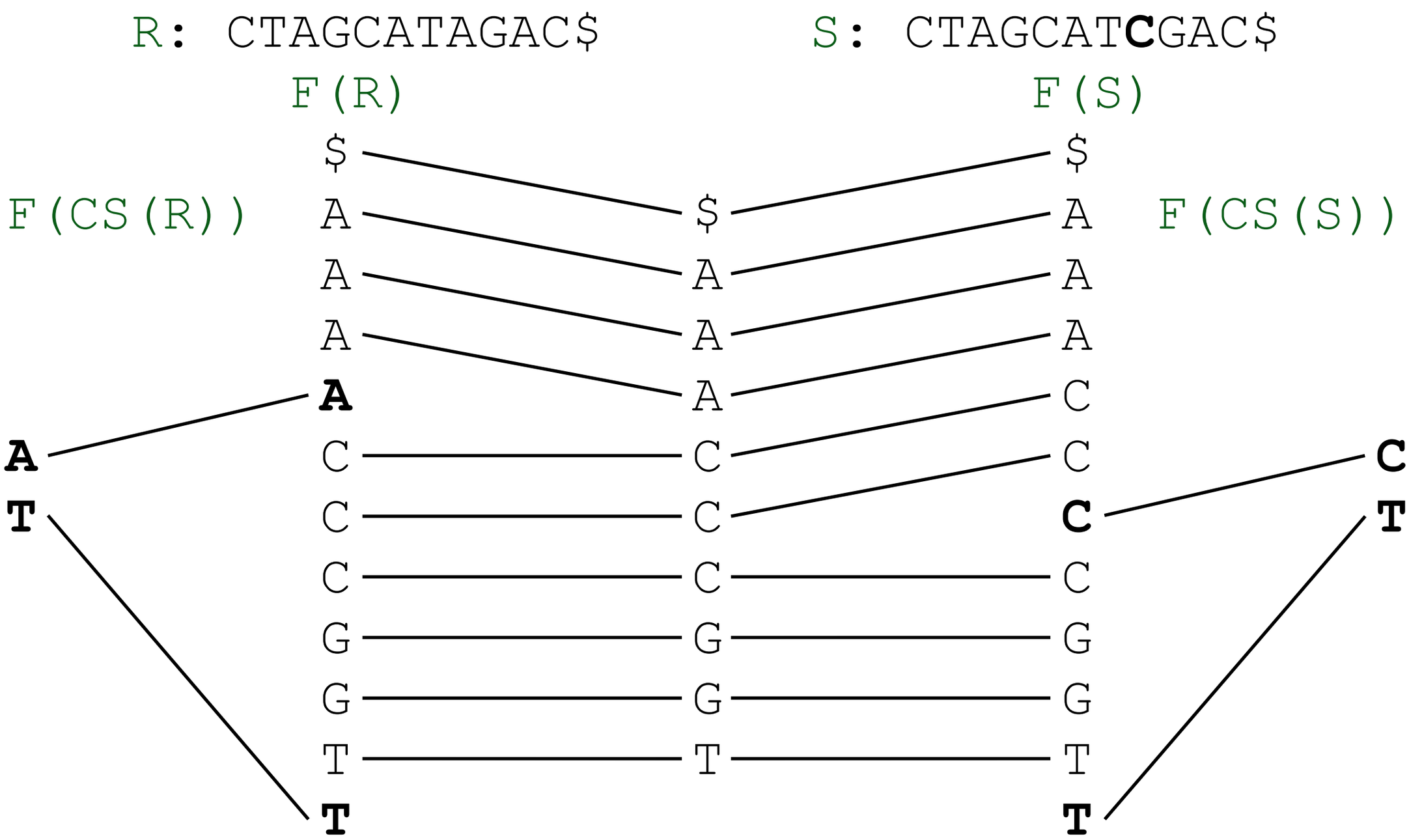
We can solve select() queries by binary searching with rank() queries. Native select() support should be much faster.

Relative FM-index



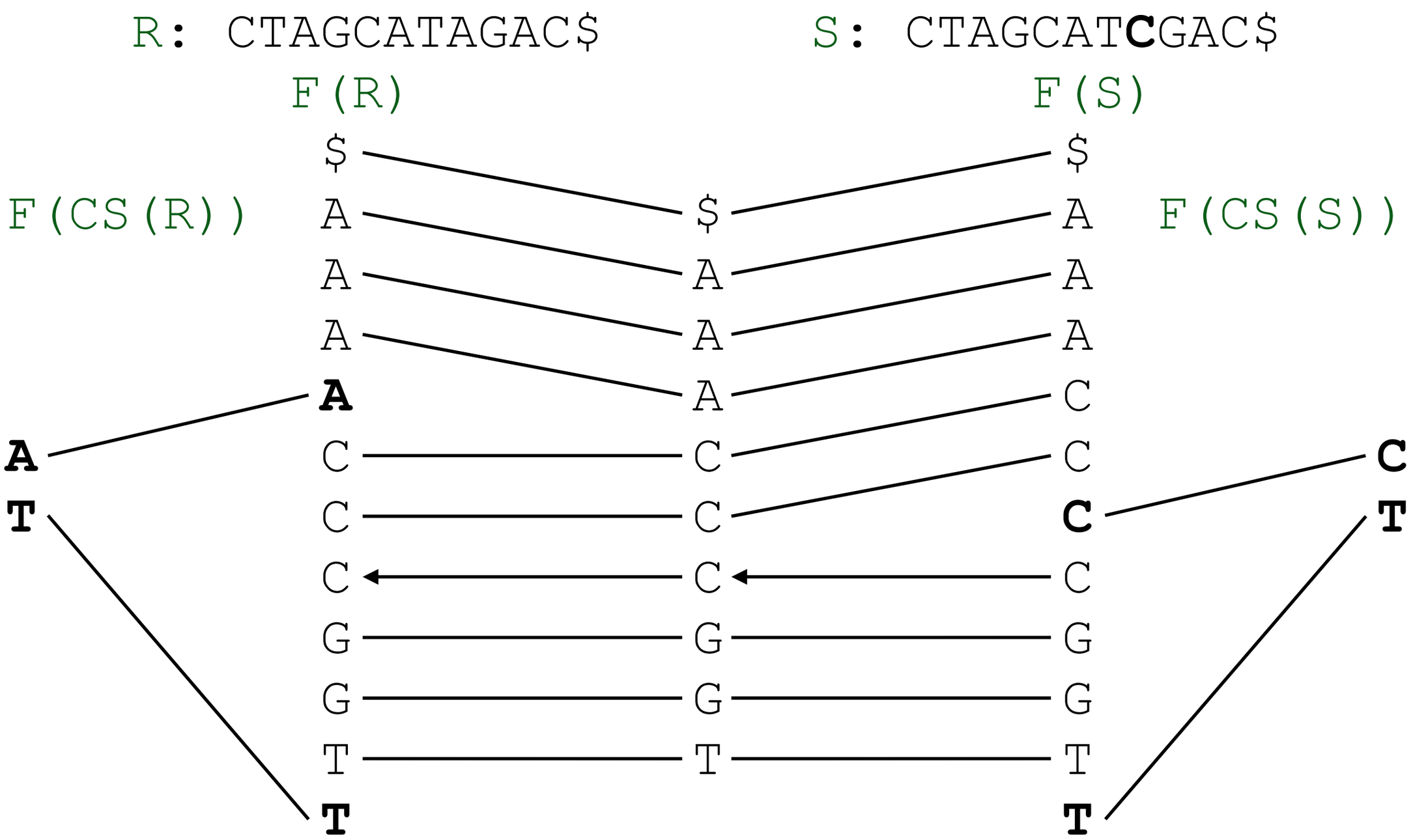
The solution for relative select() is based on stable sorting.

Relative select



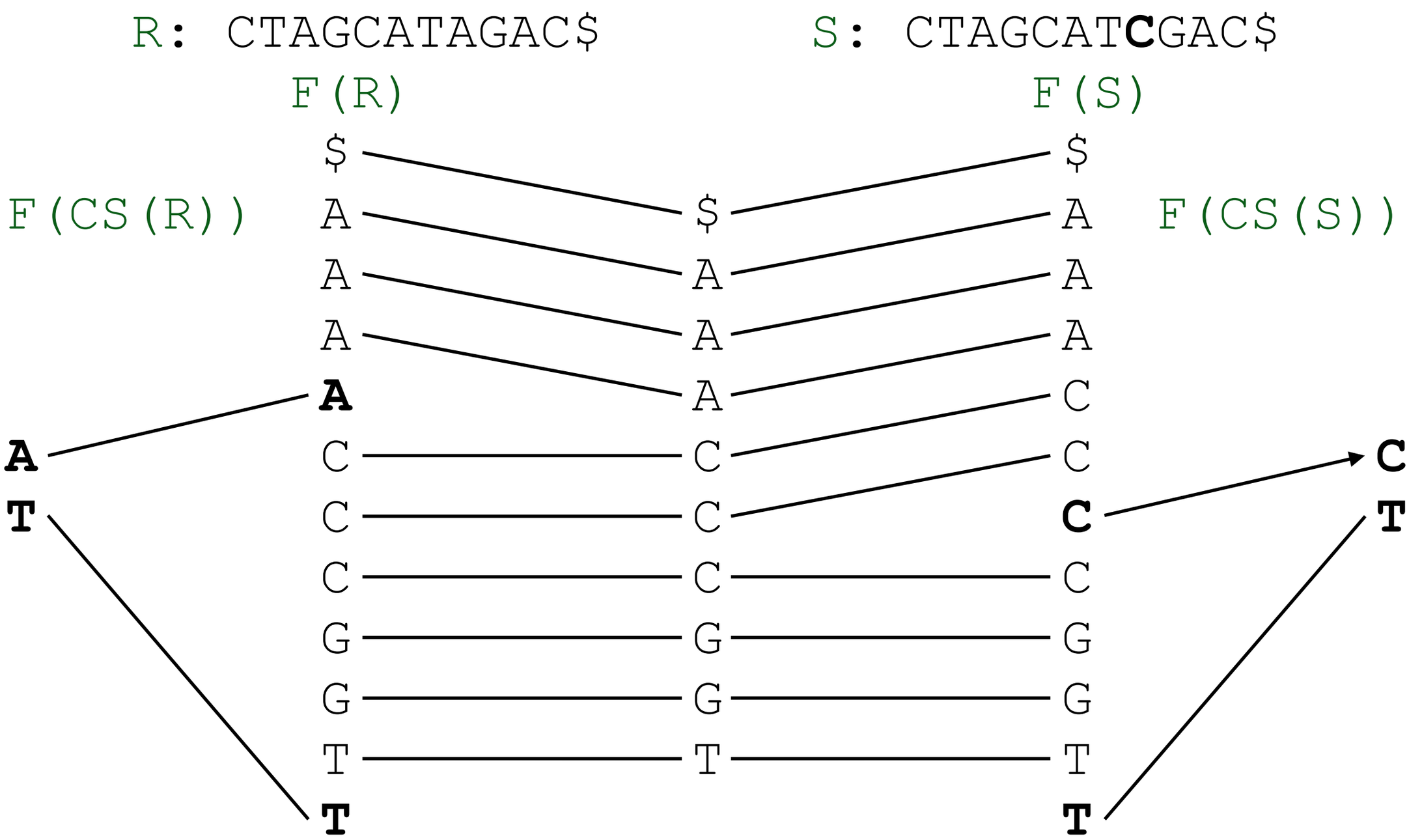
Is select(i,c) in the common subsequence or in the complement?

Relative select



BWT(S).select(4,C): map BWT(R).select(3,C) to BWT(S)

Relative select



BWT(S).select(3,C): map CS(S).select(1,C) to BWT(S)

Experiments: FM-index

- Reference sequence: Human reference genome with and without chromosome Y. Target sequence: Maternal haplotypes of NA12878.

- Queries: 100 million random backward (rank(), LF()) and forward (select(),Ψ()) queries in a single thread.

- The implementation is based on SDSL.

https://github.com/jltsiren/relative-fm
https://github.com/simongog/sdsl-lite

Experiments: FM-index

ChrY	Index	Size	Backward	Forward
Yes	SSA	1090 MB	0.55 µs	1.22 µs
Yes	RFM	218 MB	3.95 µs	48.0 µs
Yes	RFM rselect	382 MB	3.95 µs	6.11 µs
No	SSA	1090 MB	0.55 µs	1.11 µs
No	RFM	181 MB	3.84 µs	44.8 µs
No	RFM rselect	331 MB	3.84 µs	6.12 µs

Experiments: CST

- Gagie, Navarro, Puglisi, and Sirén: **Relative Compressed Suffix Trees**. arXiv:1508.02550.

- Comparison against the SDSL implementations of CSTs using NA12878 and the reference without chromosome Y.

- Full traversal using SDSL iterators.

- Matching statistics for another assembly of chromosome 1 using forward searching.

Experiments: CST

Index	Size	Traversal	Matching statistics
cst_sada	12.33 bpc	5 min	315 min
cst_sct3 PLCP	10.79 bpc	18 min	195 min
cst_sct3 LCP-byte	18.08 bpc	18 min	120 min
cst_fully	4.98 bpc	–	–
RCST	3.16 bpc	39 min	910 min
RCST rselect	3.61 bpc	39 min	389 min

Conclusions

- We augmented the relative FM-index with native select() queries.

- The native select() support is 7–8 times faster than using binary search with rank() queries.

- The augmented RFM index yields competitive time/space trade-offs for forward searching in compressed suffix trees.