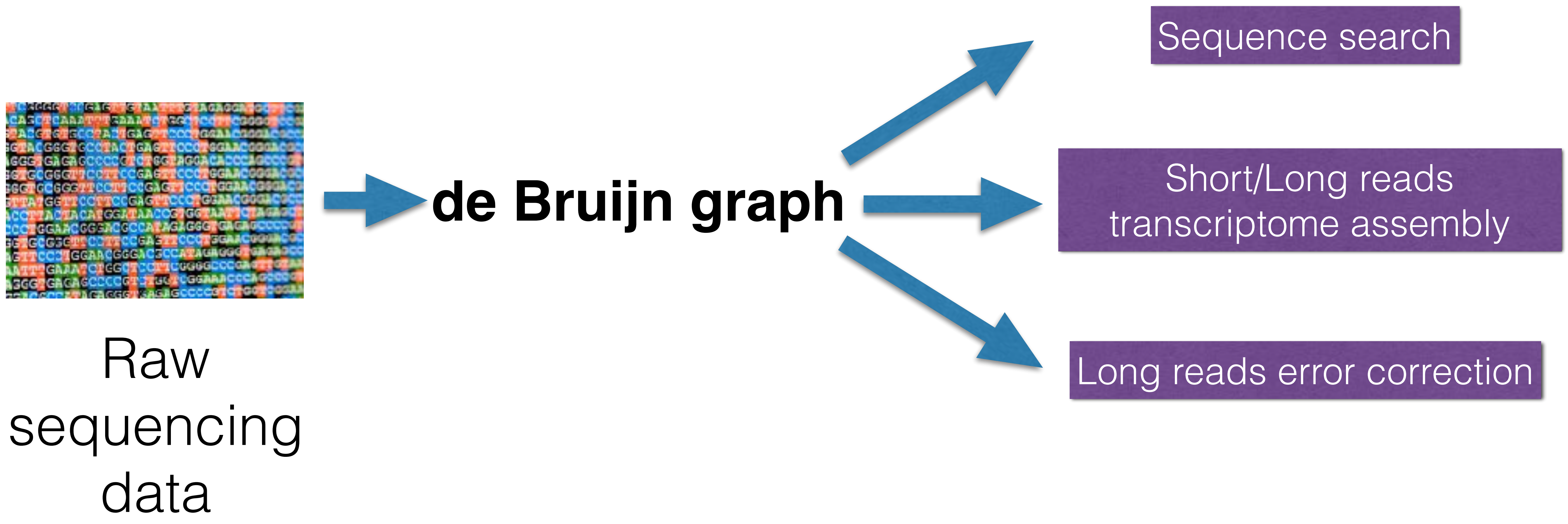
deBGR: An Efficient and Near-Exact Representation of the Weighted de Bruijn Graph

Prashant Pandey[1], Michael A. Bender[1], Rob Johnson[12], and Rob Patro[1]
[1]Stony Brook University, NY USA, [2]VMWare Inc USA

de Bruijn graphs are ubiquitous


Raw sequencing data
de Bruijn graph
Sequence search
Short/Long reads transcriptome assembly
Long reads error correction

A de Bruijn graph is the data representation at the heart of a lot of sequence analyses.

de Bruijn graph (dBG)

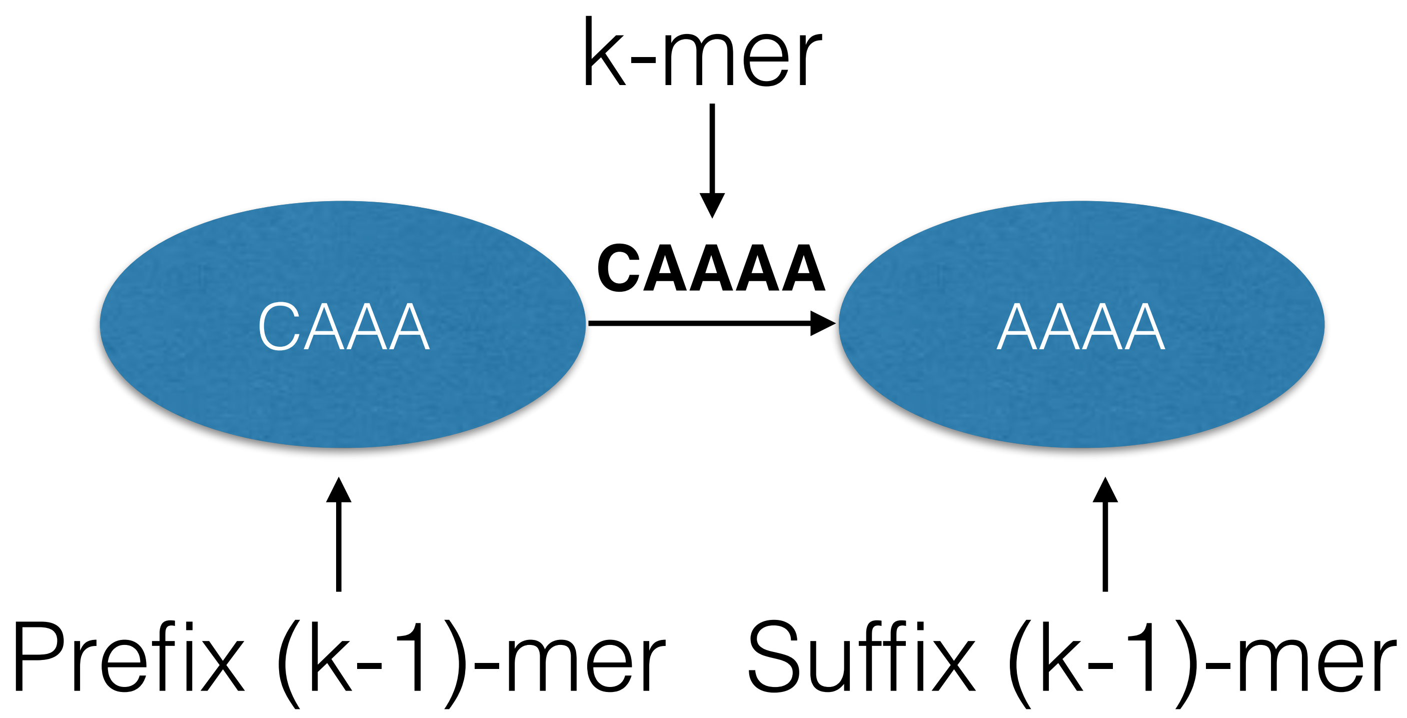
A ***read*** is a string of bases over the DNA alphabet A, C, T, and G. → GGCCAAAATTCG

A **k-mer** is a substring of length k. Here, k is 5. →

```
GGCCA
 GCCAA
  CCAAA
   CAAAA
    AAAAT
     AAATT
      AATTC
       ATTCG
```

de Bruijn graph (dBG)

Read: ….CAAAA….

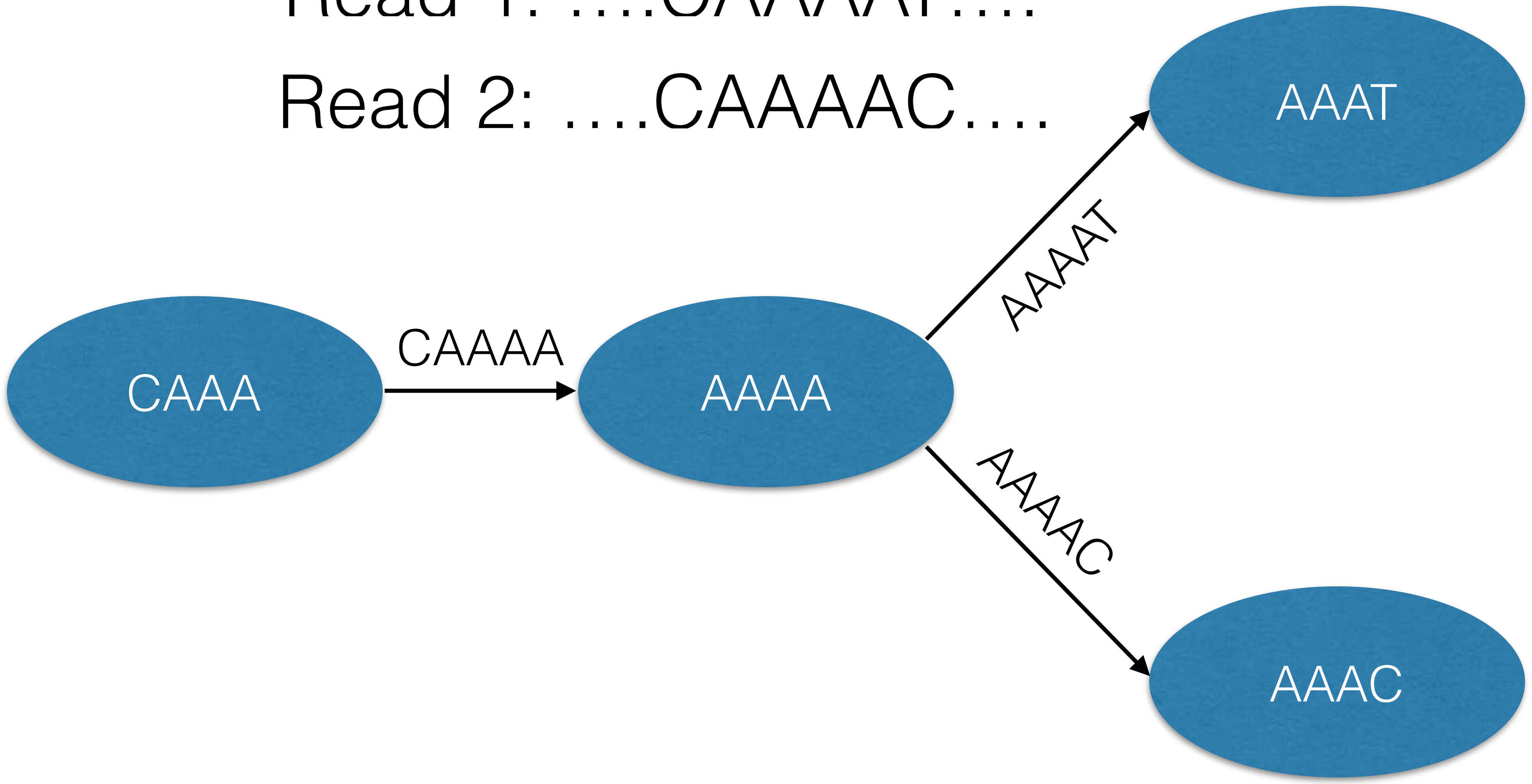


An edge is a k-mer connecting its two k-1 substrings.

de Bruijn graph (dBG)

Read 1: ….CAAAAT….

Read 2: ….CAAAAC….



Weighted de Bruijn graphs

- Topology-only de Bruijn graphs are not adequate for transcriptome assembly.
- Abundance information of each k-mer is critical for transcriptome assembly.

Weighted de Bruijn graphs

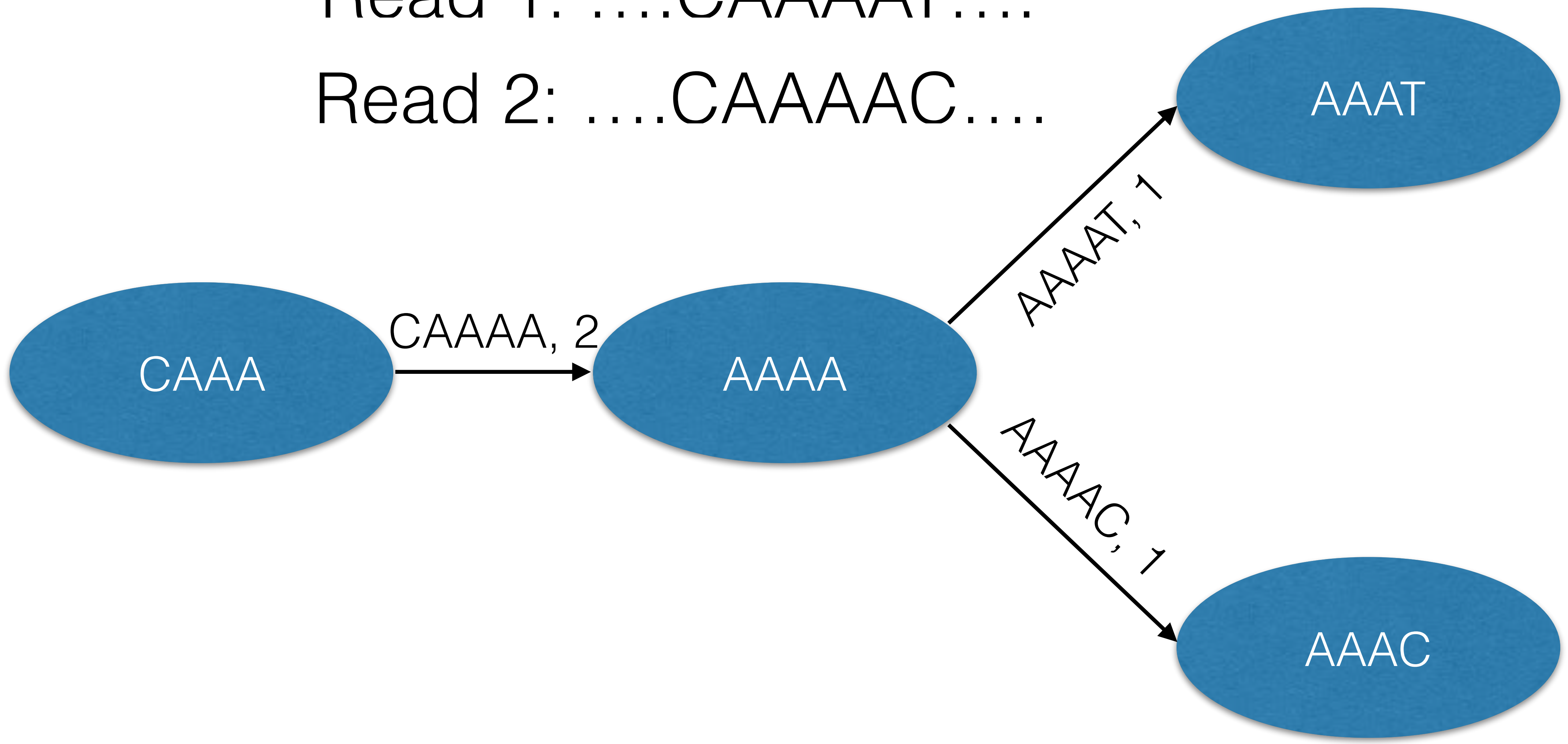
- Topology-only de Bruijn graphs are not adequate for transcriptome assembly.
- Abundance information of each k-mer is critical for transcriptome assembly.

Weighted de Bruijn graphs pose an extra obligation and opportunity.

Weighted de Bruijn graph (WdBG)

Read 1: ….CAAAAT….

Read 2: ….CAAAAC….



A weighted de Bruijn graph associates each edge (k-mer) its abundance in the underlying dataset.

Measuring dBG representation

de Bruijn graphs store only k-mers, memory usage scales with the number of unique k-mers.

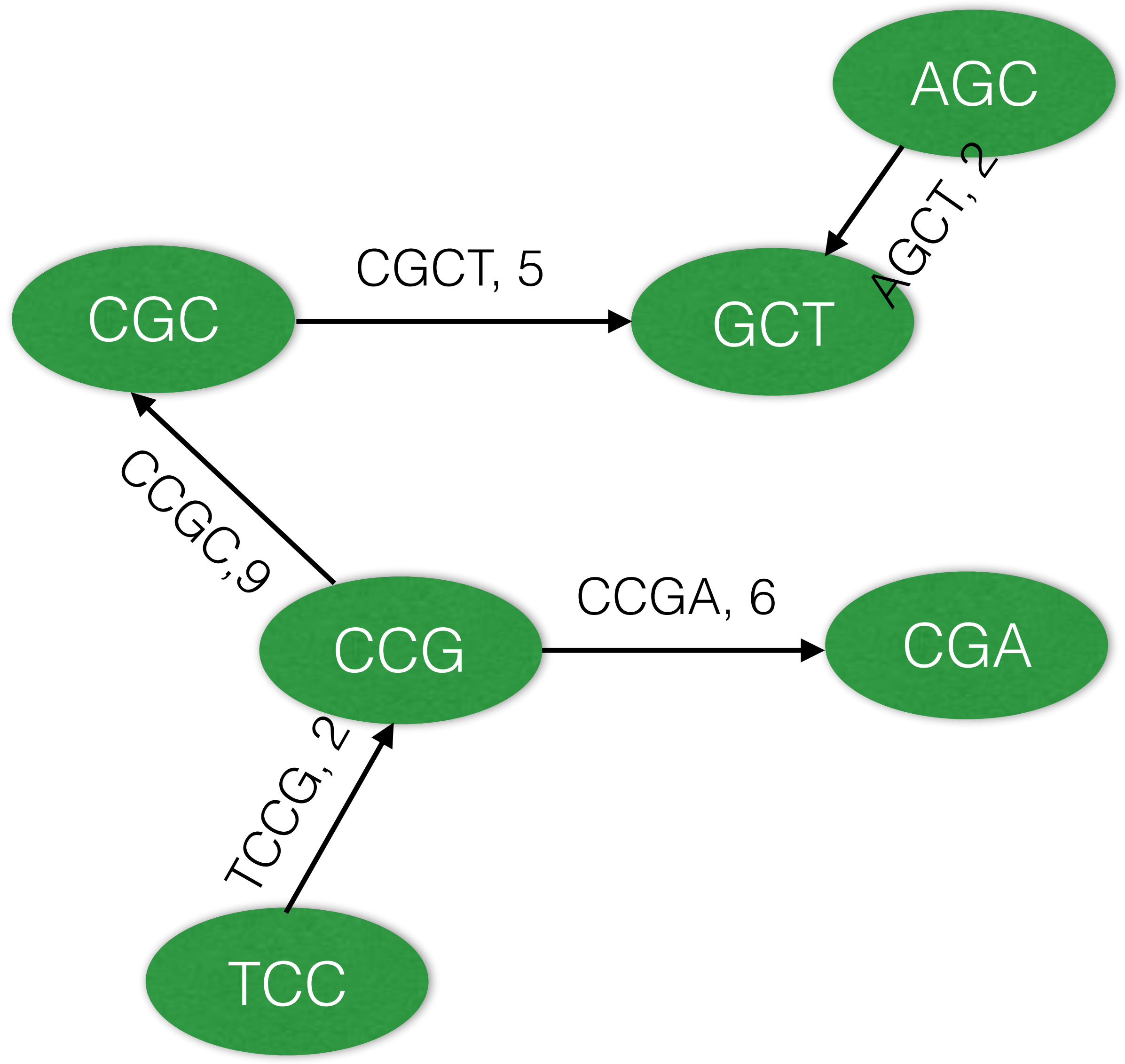
Human genome (few Billion k-mers): >100 GB
Soil metagenomes (few Million species): Few TBs

Beefy server machines are needed to perform weighted de Bruijn graph analysis.

WdBG as a multiset

MultiSet
TCCG, 2
CCGC, 9
CCGA, 6
CGCT, 5
AGCT, 2

(Edge, Abundance)



Weighted de Bruijn graph

Past work on Probabilistic dBG representation

- **Pell et al. 2012**: Represented dBG using a Bloom filter.

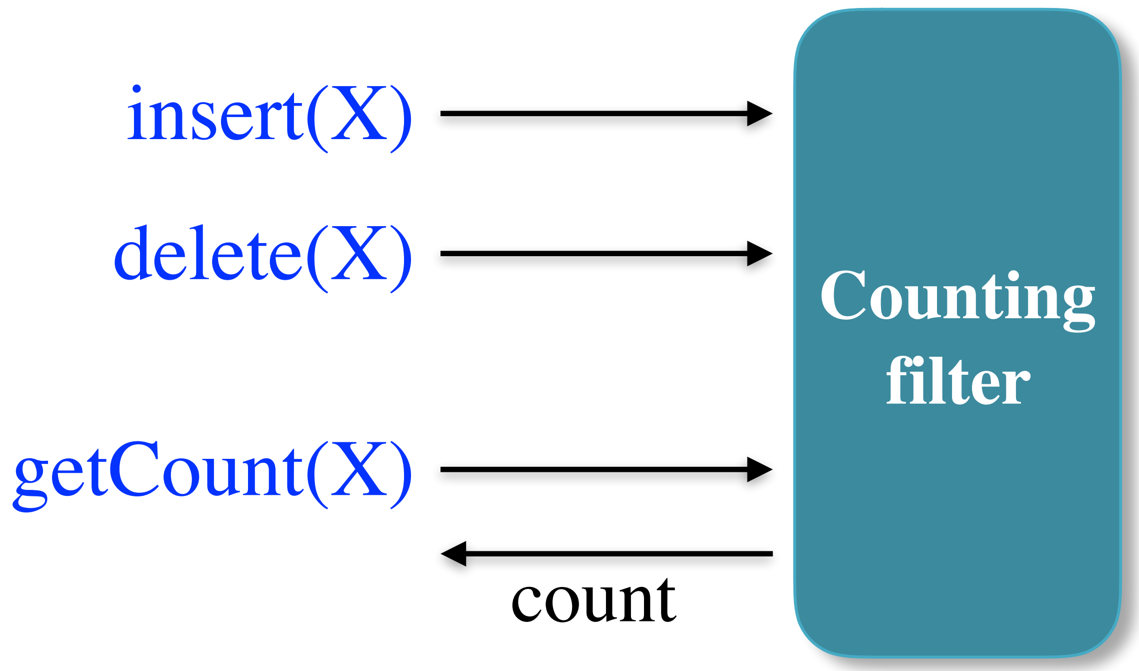
Past work on Probabilistic dBG representation

- **Pell et al. 2012**: Represented dBG using a Bloom filter.
- **Pellow et al. 2016**: Showed how to exploit redundancy in k-mers to reduce the false-positive rate of the Bloom filter without increasing the space.

Past work on Probabilistic dBG representation

- **Pell et al. 2012**: Represented dBG using a Bloom filter.
- **Pellow et al. 2016**: Showed how to exploit redundancy in k-mers to reduce the false-positive rate of the Bloom filter without increasing the space.
- **Chikhi and Rikz 2013 and Salikhov et al. 2013:** They showed how to convert a probabilistic representation into an exact one using a small and exact auxiliary data structure.

Counting filters



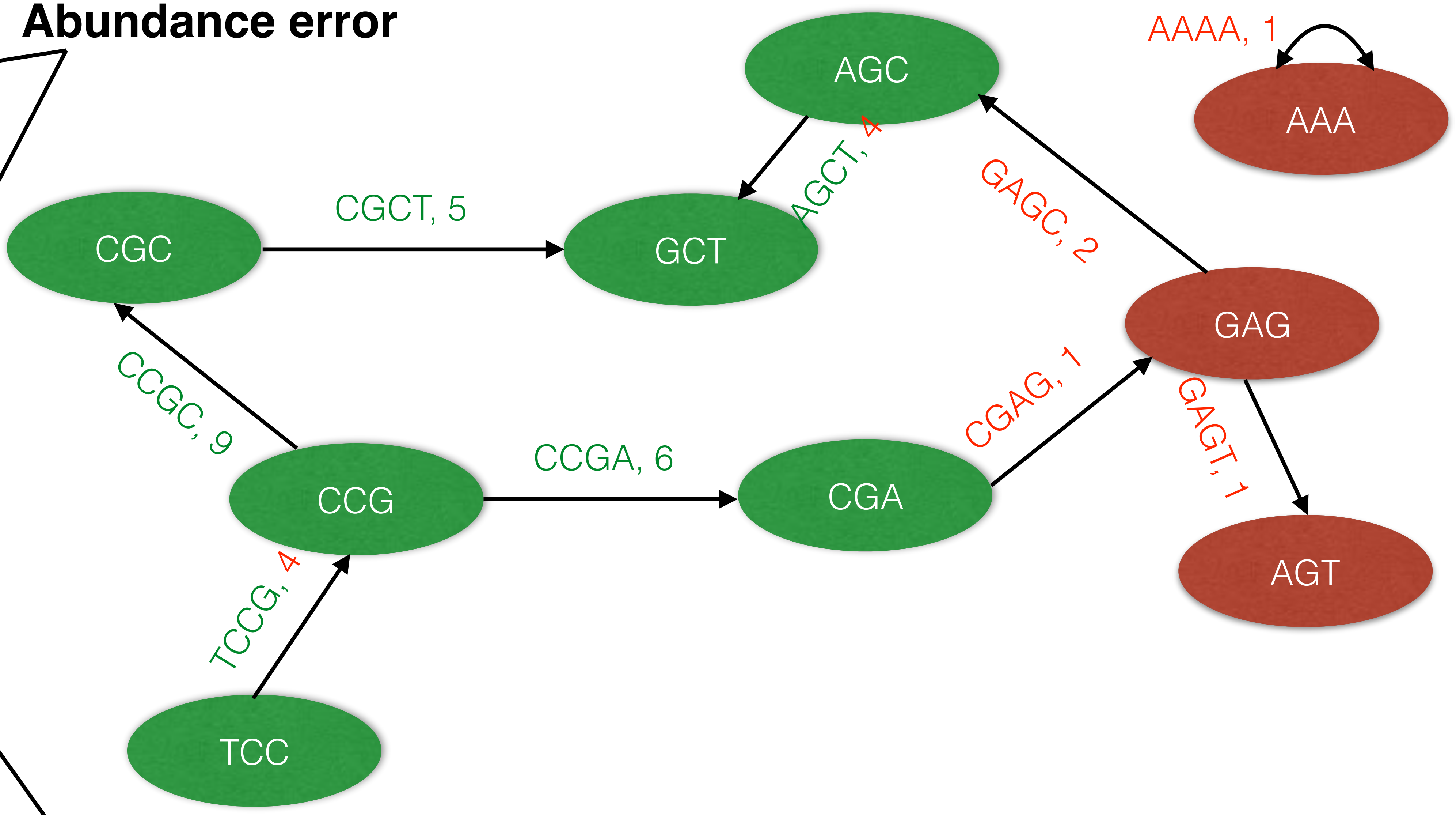
- **A counting filter is a lossy representation of a multiset.**
- **Operations: inserts, count, and delete.**
- **Generalizes AMQs**
  - False positives ≈ over-counts.
- **Counting quotient filter**[Pandey et al. 2017]

Probabilistic weighted de Bruijn graph [Pandey et al. 2017]

Counting quotient filter
TCCG, 4
CCGC, 9
CCGA, 6
CGCT, 5
AGCT, 4
GAGC, 2
CGAG, 1
GAGT, 1
AAAA, 1

Abundance error

Topological errors



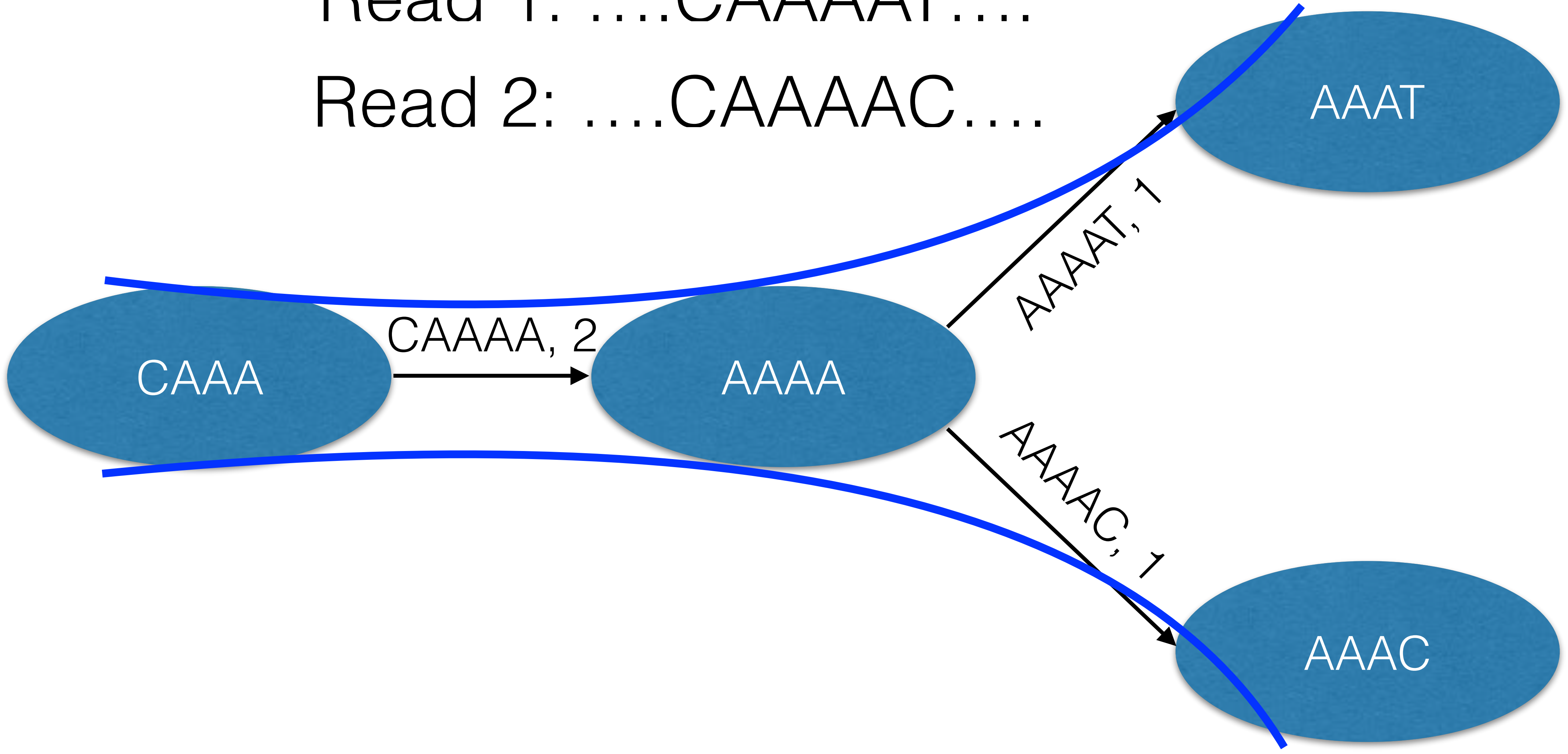
This paper: deBGR

- An **exact representation** of the weighted de Bruijn graph.
  - An algorithm that uses counts in the approximate representation in an AMQ to iteratively **self-correct approximation errors**.
  - It corrects both kinds of errors, **abundance and topological errors** and supports **membership queries**.
  - It **supports deletion** of k-mers from the structure.
  - It takes 18-28% more space than the approximate representation and has **no errors**.

A weighted de Bruijn graph invariant

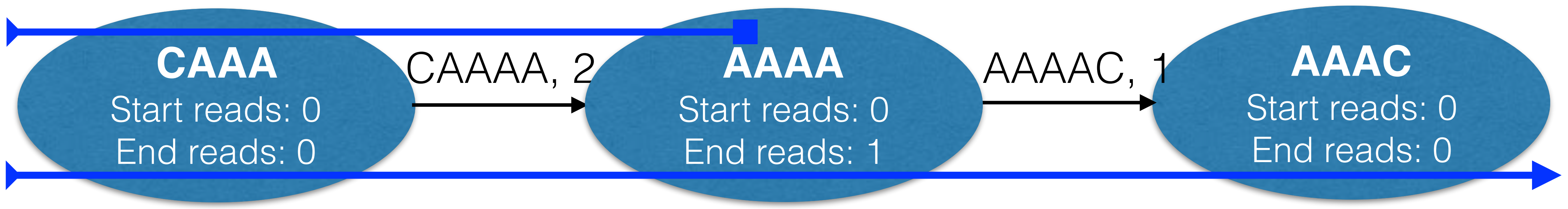
Read 1: ....CAAAAT....

Read 2: ....CAAAAC....



Total incoming abundance = Total outgoing abundance

A weighted de Bruijn graph invariant



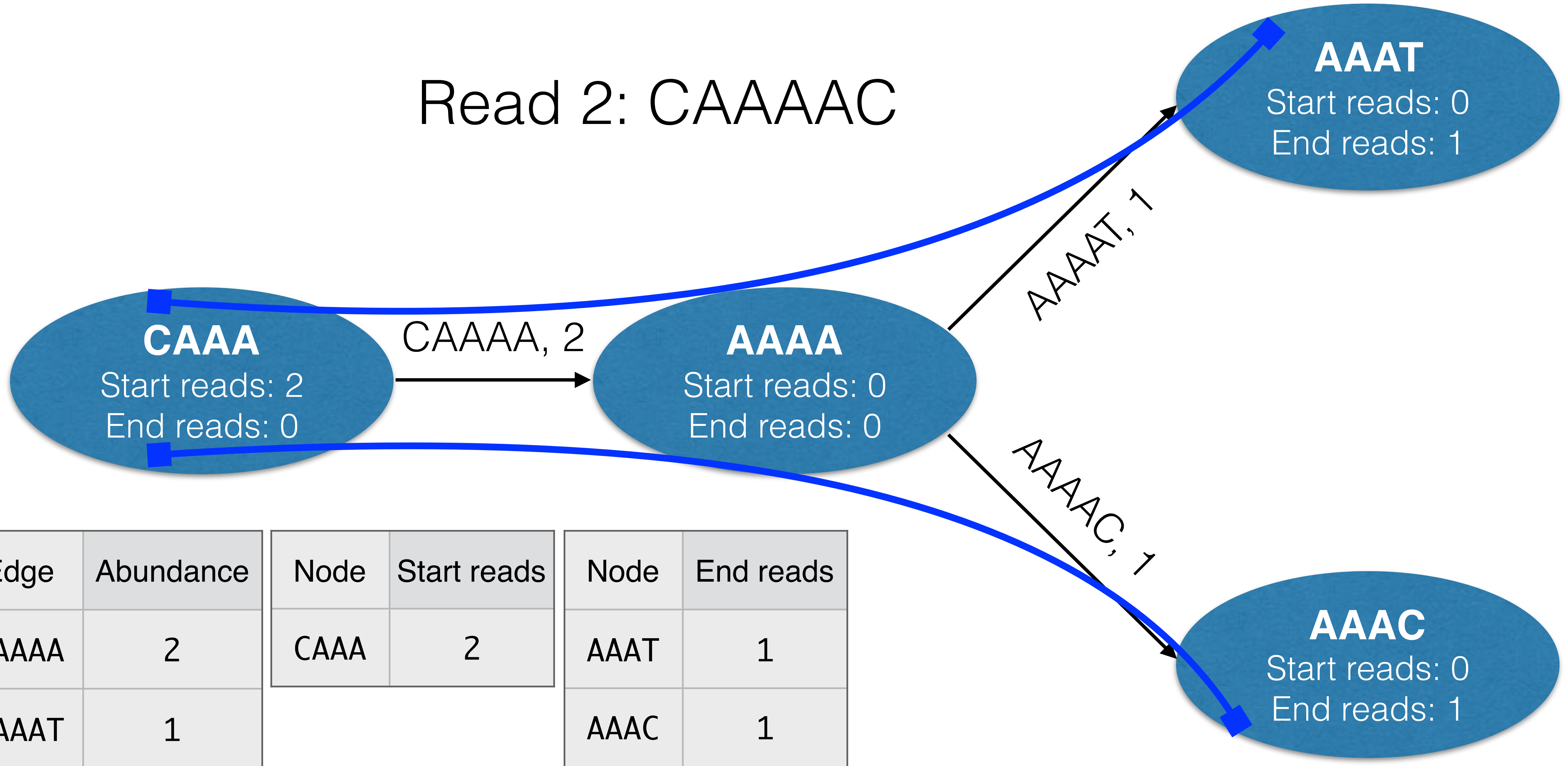
Total incoming abundance = Total outgoing abundance*

***After accounting for read starts and ends.**

WdBG representation in deBGR

Read 1: CAAAAT

Read 2: CAAAAC

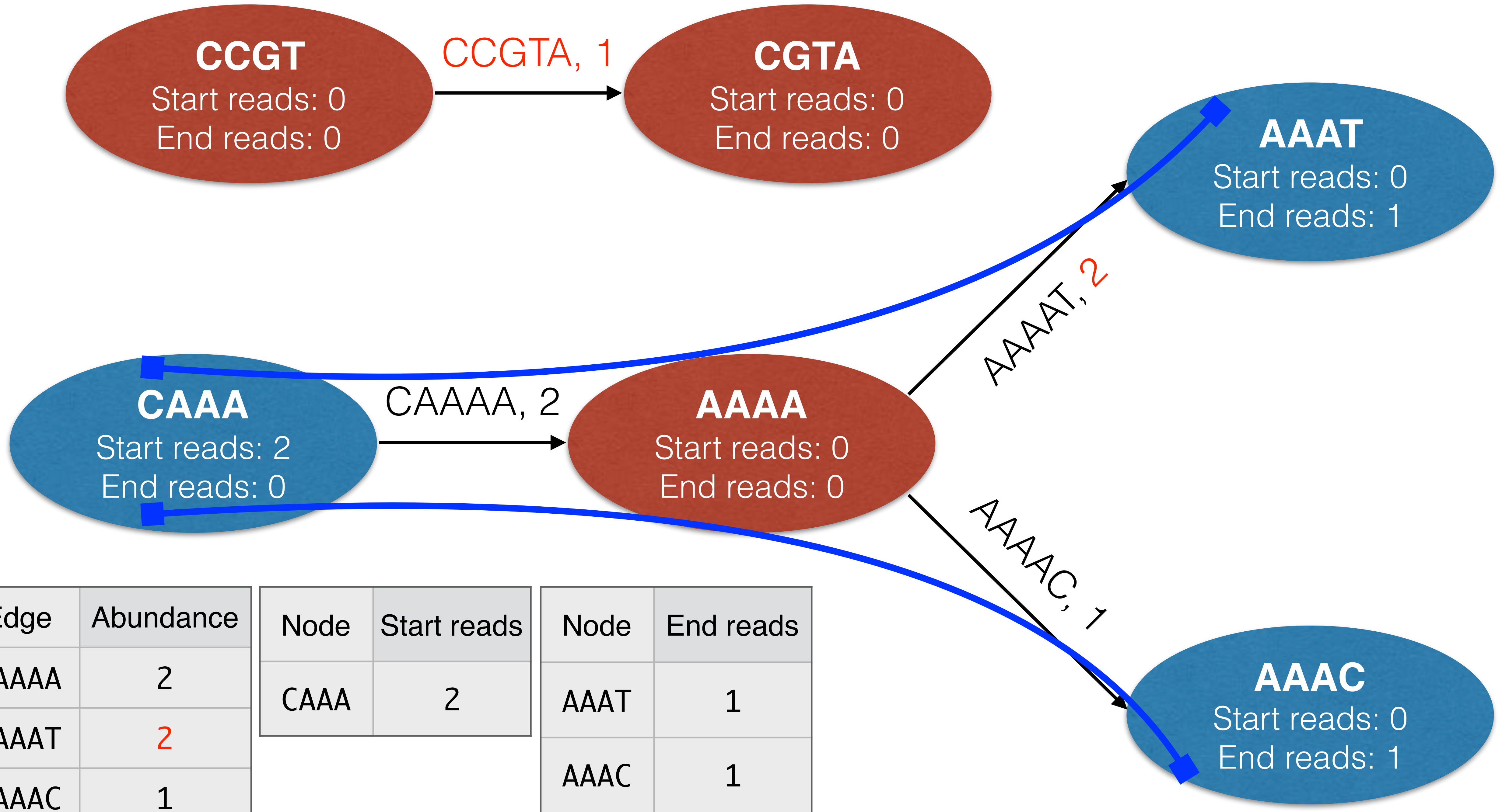


Edge	Abundance
CAAAA	2
AAAAT	1
AAAAC	1

Node	Start reads
CAAA	2

Node	End reads
AAAT	1
AAAC	1

WdBG representation in deBGR

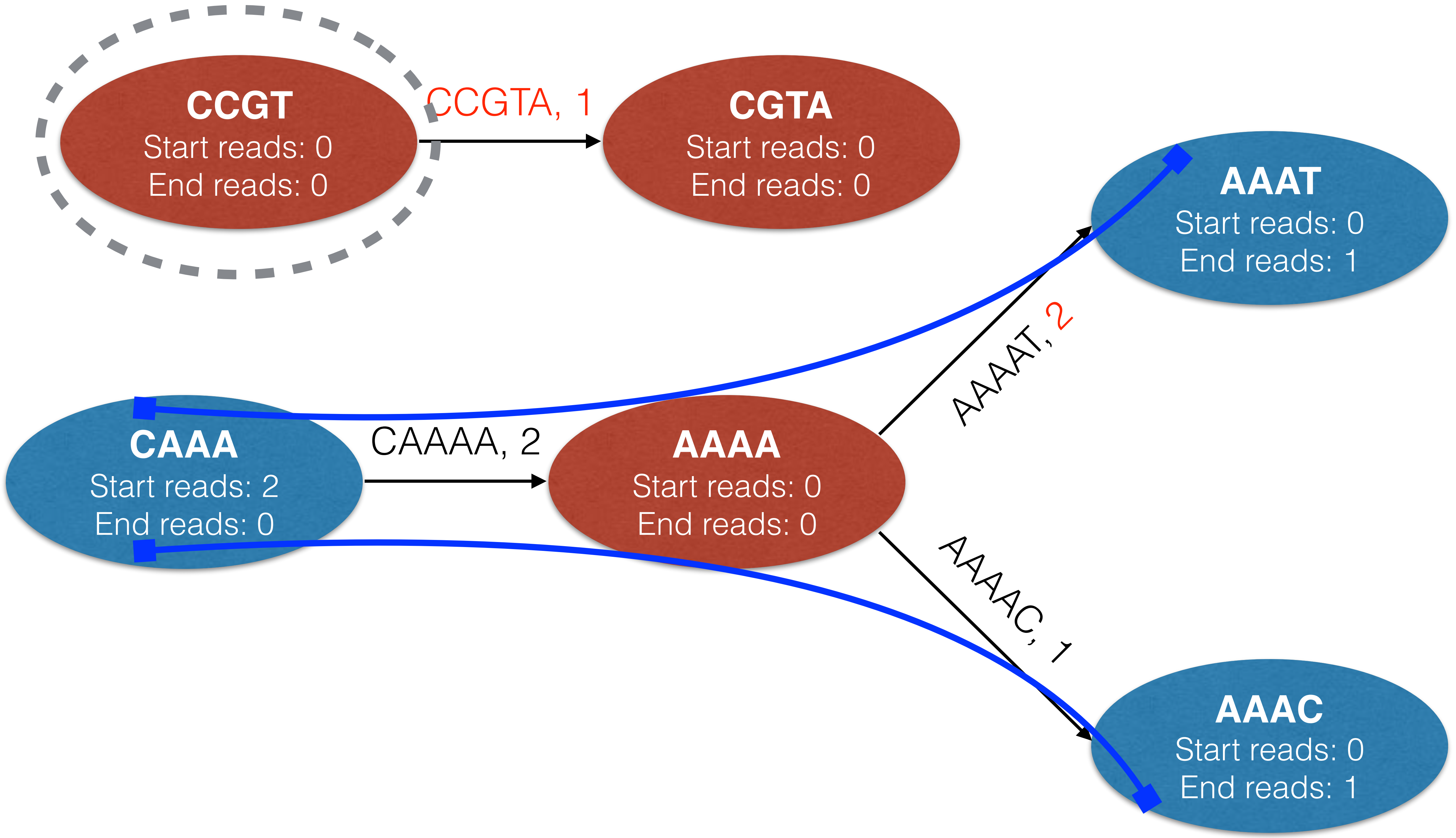


Edge	Abundance
CAAAA	2
AAAAT	2
AAAAC	1
CCGTA	1

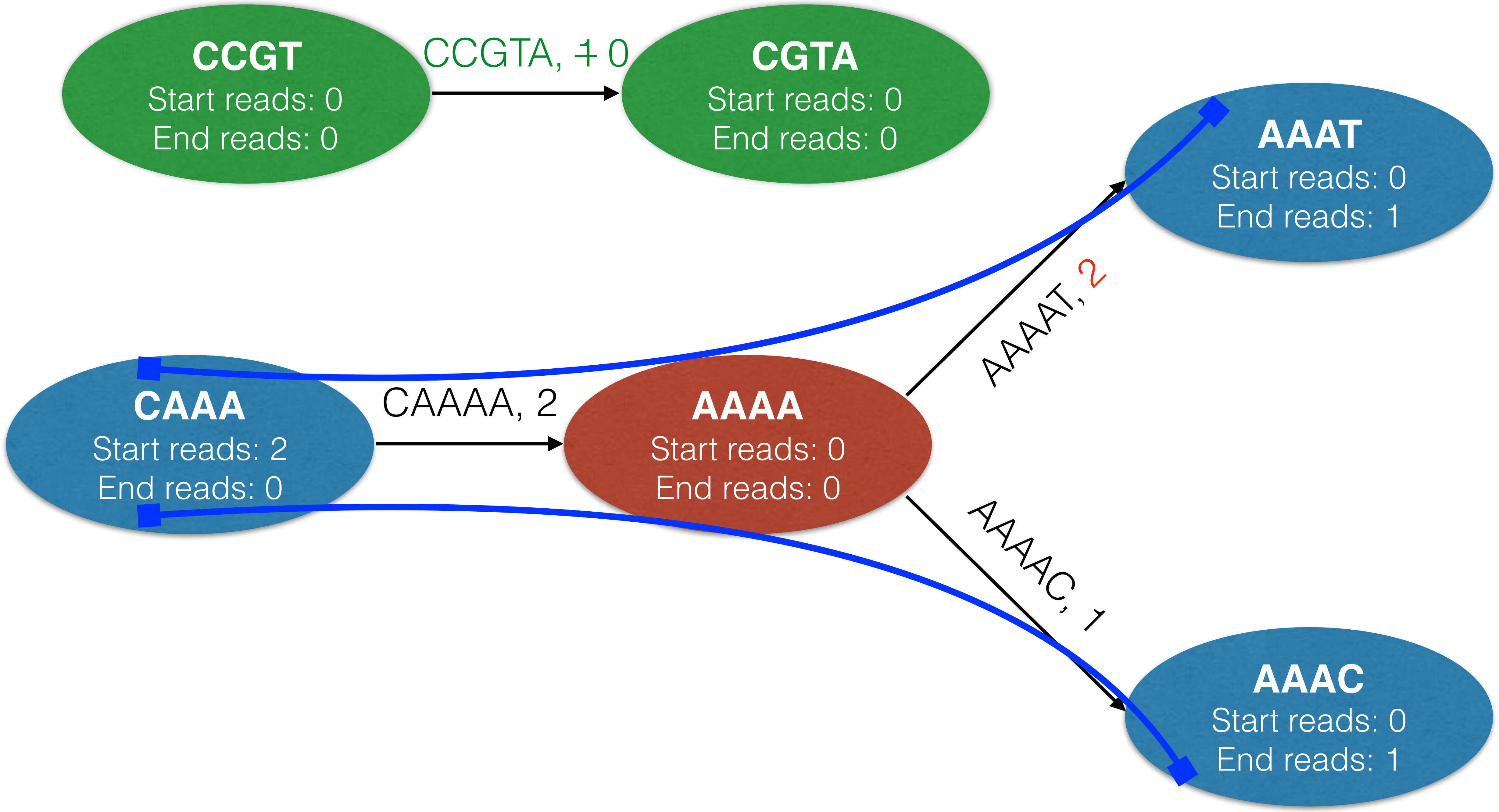
Node	Start reads
CAAA	2

Node	End reads
AAAT	1
AAAC	1

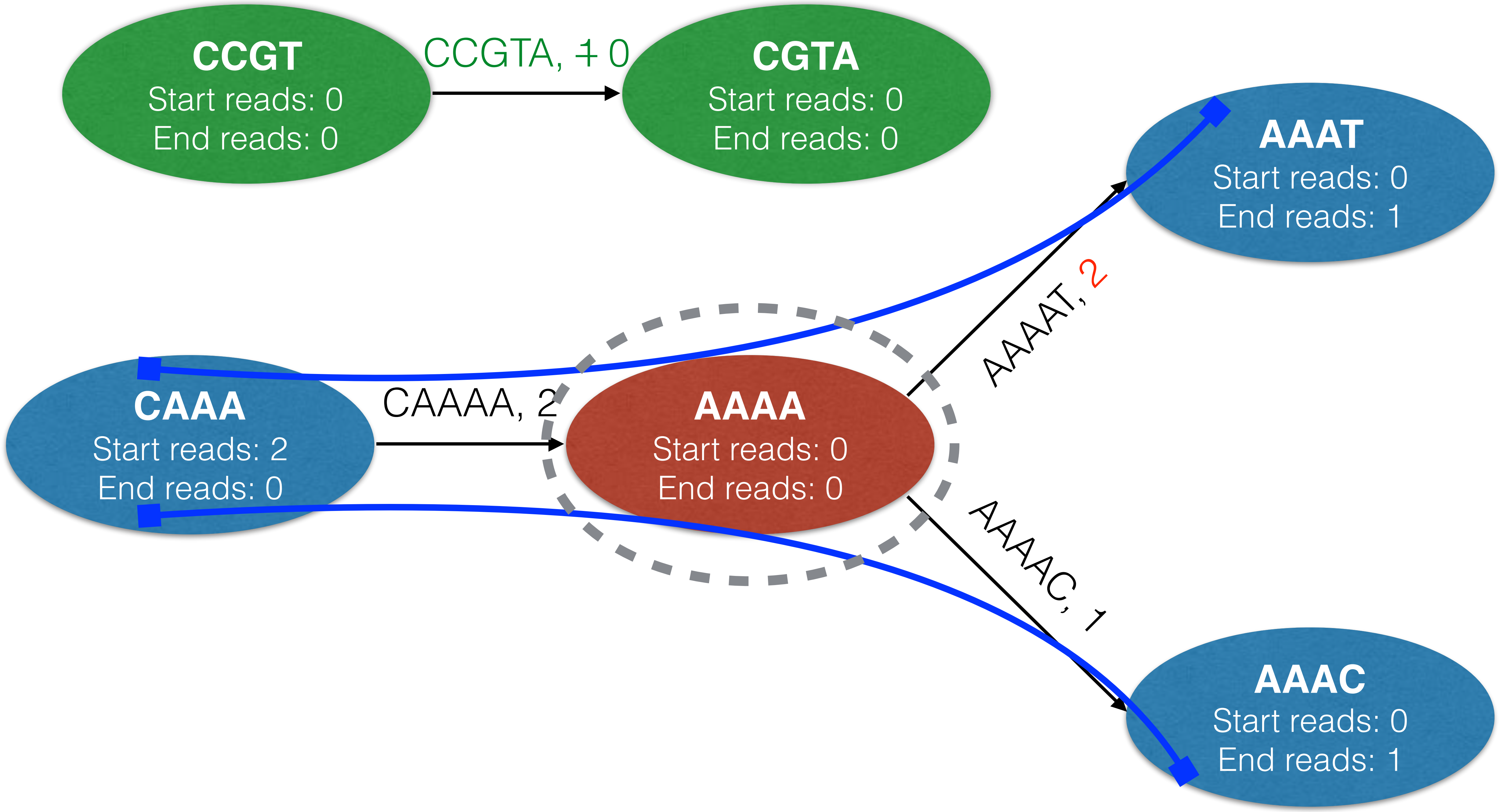
Error correction



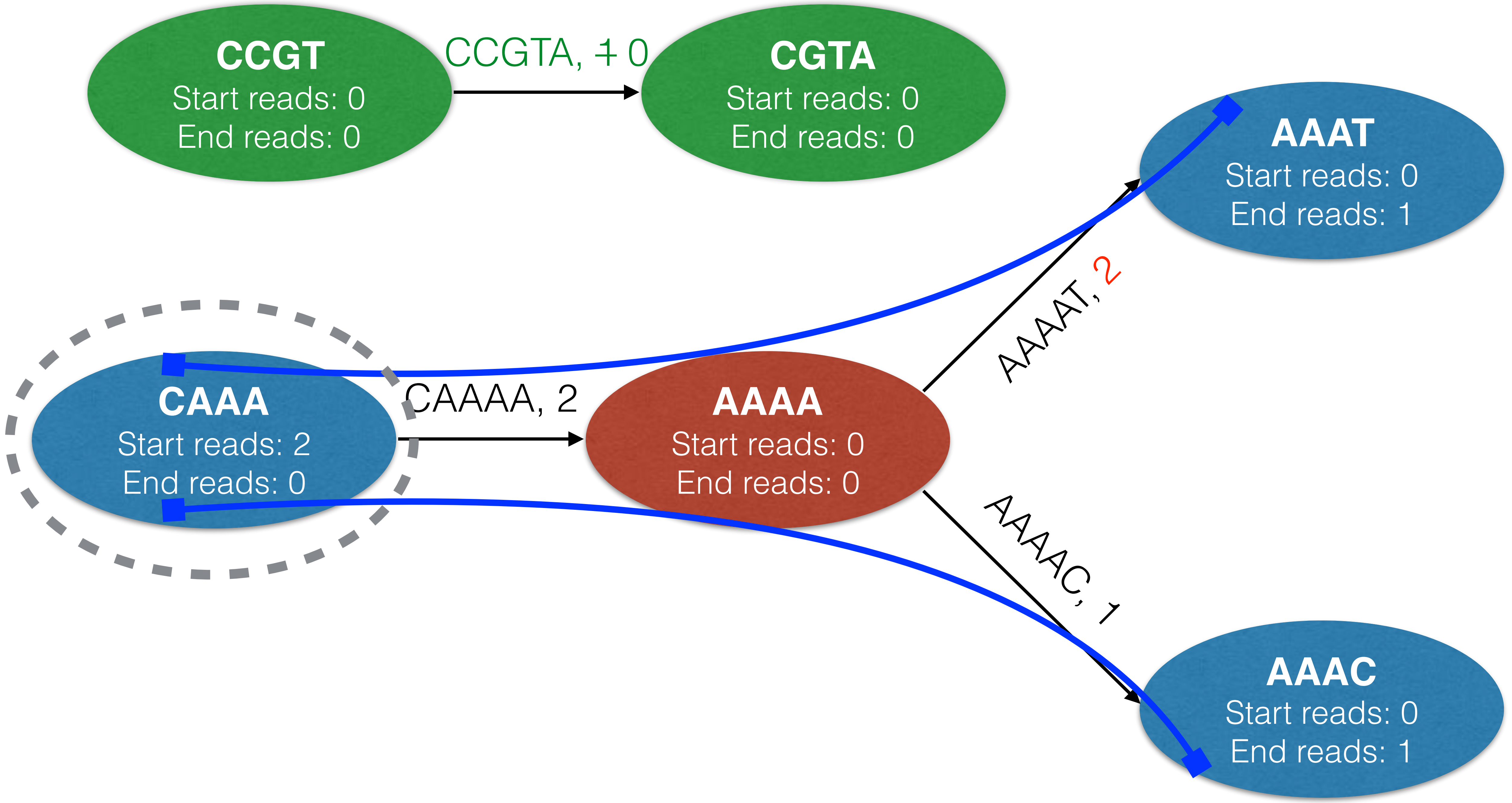
Error correction



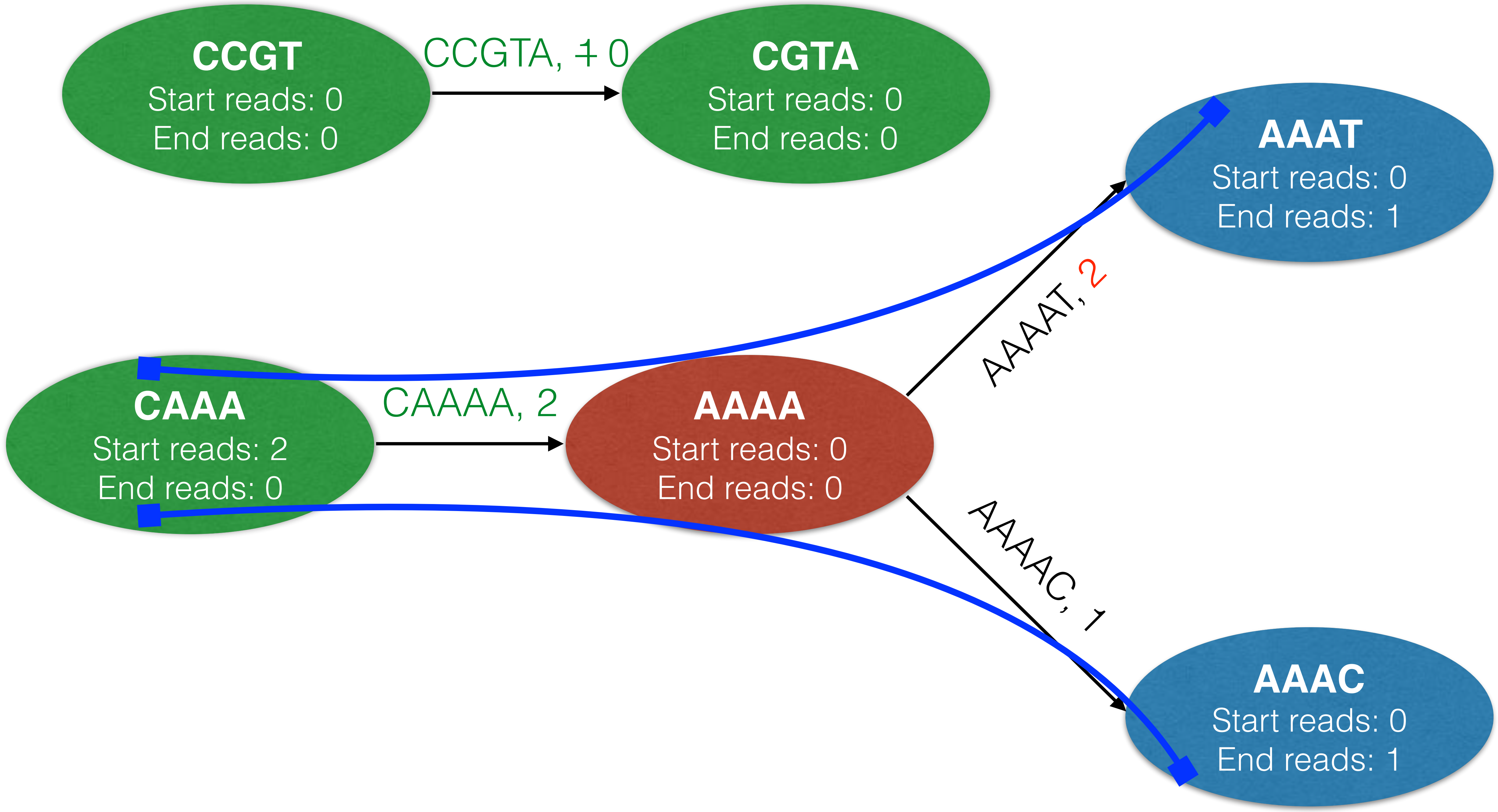
Error correction



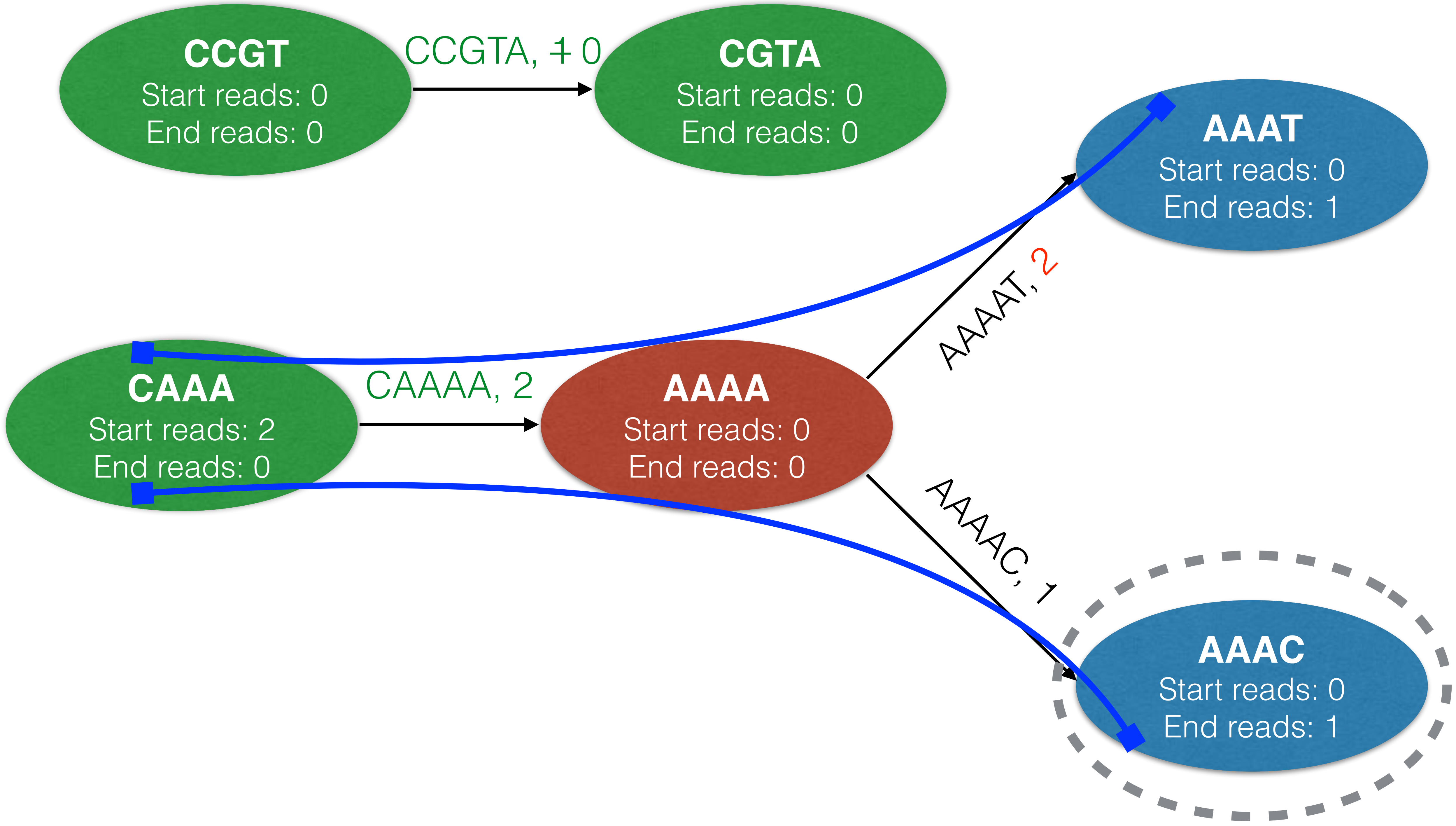
Error correction



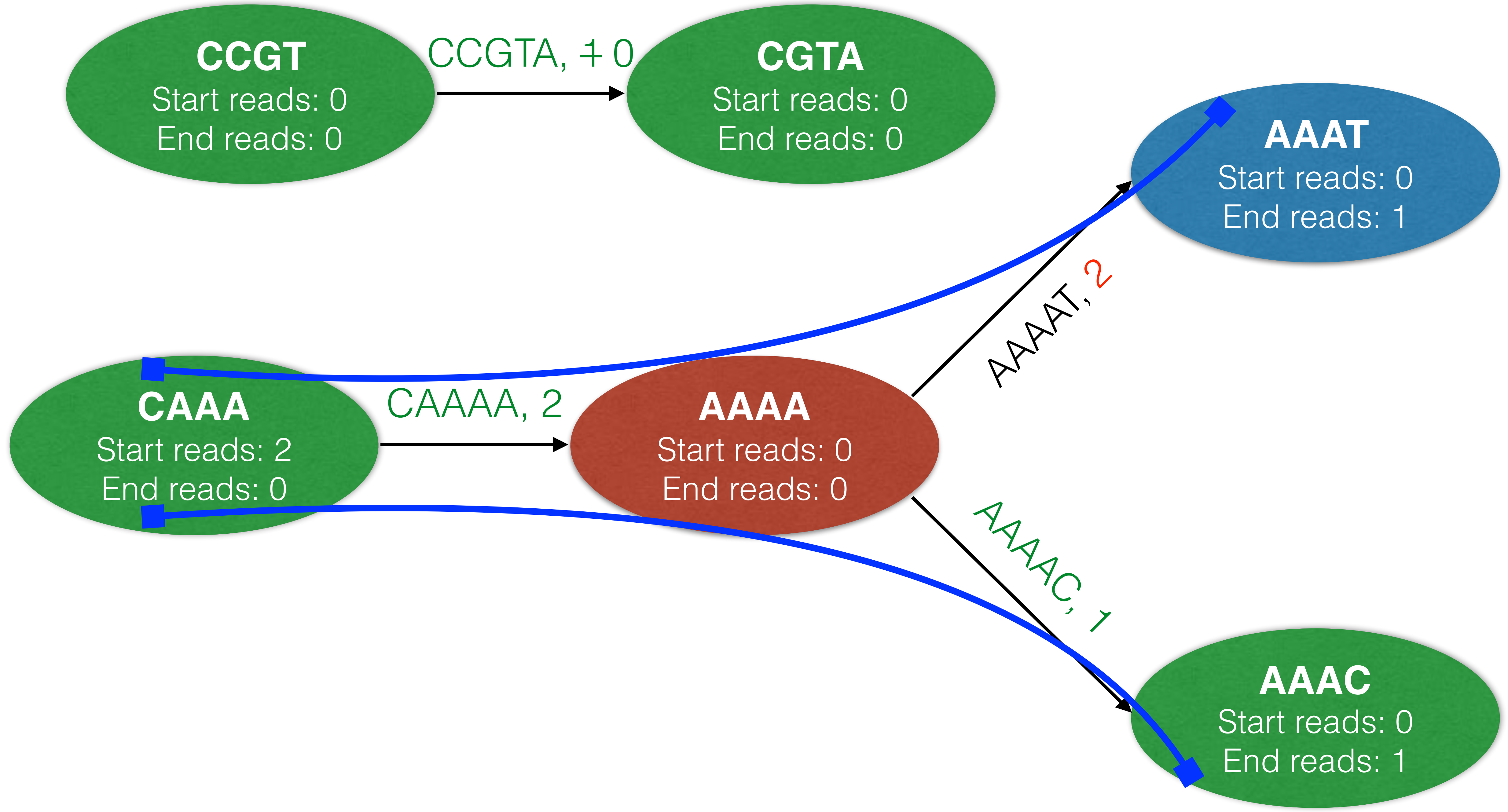
Error correction



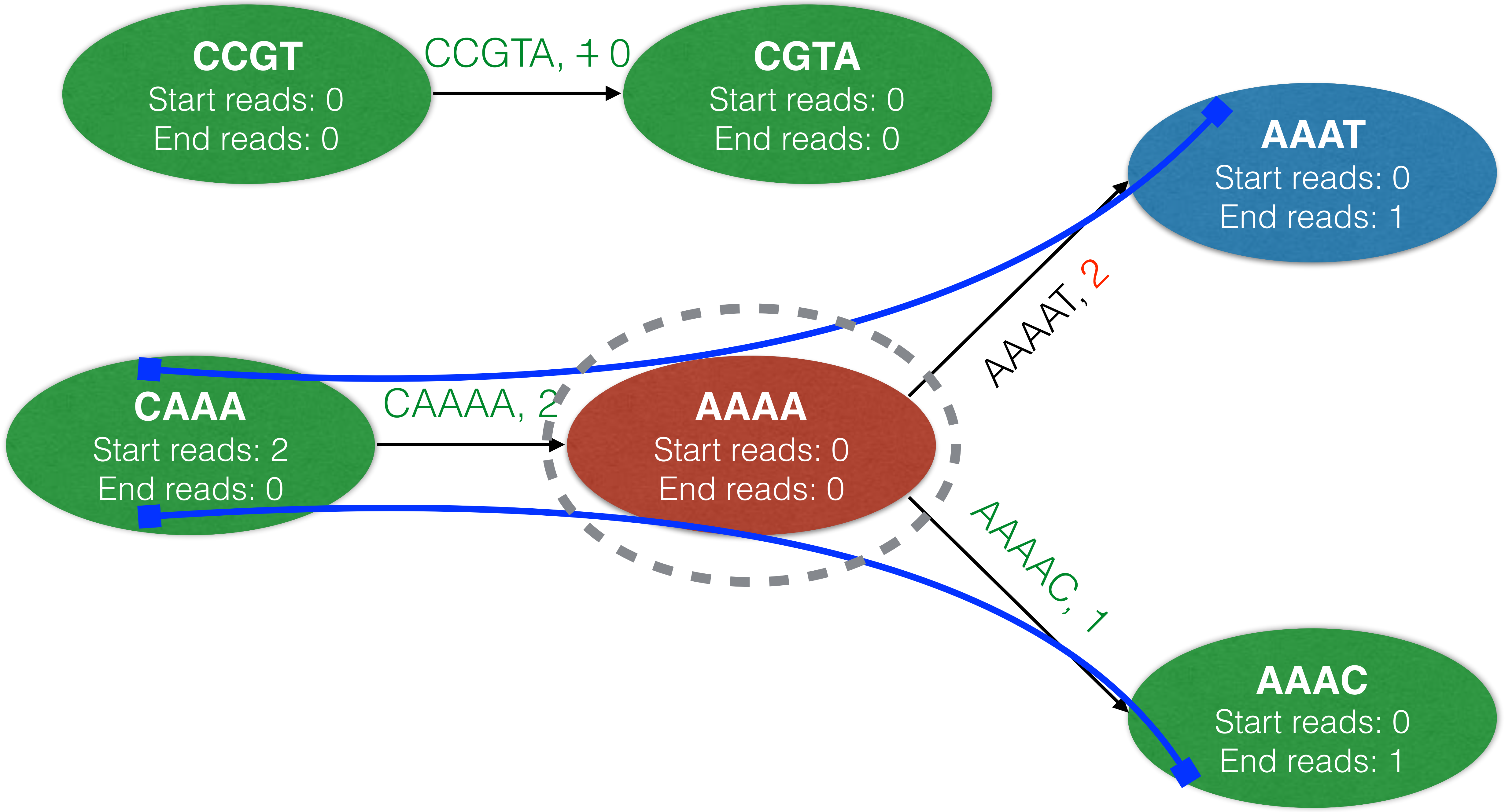
Error correction



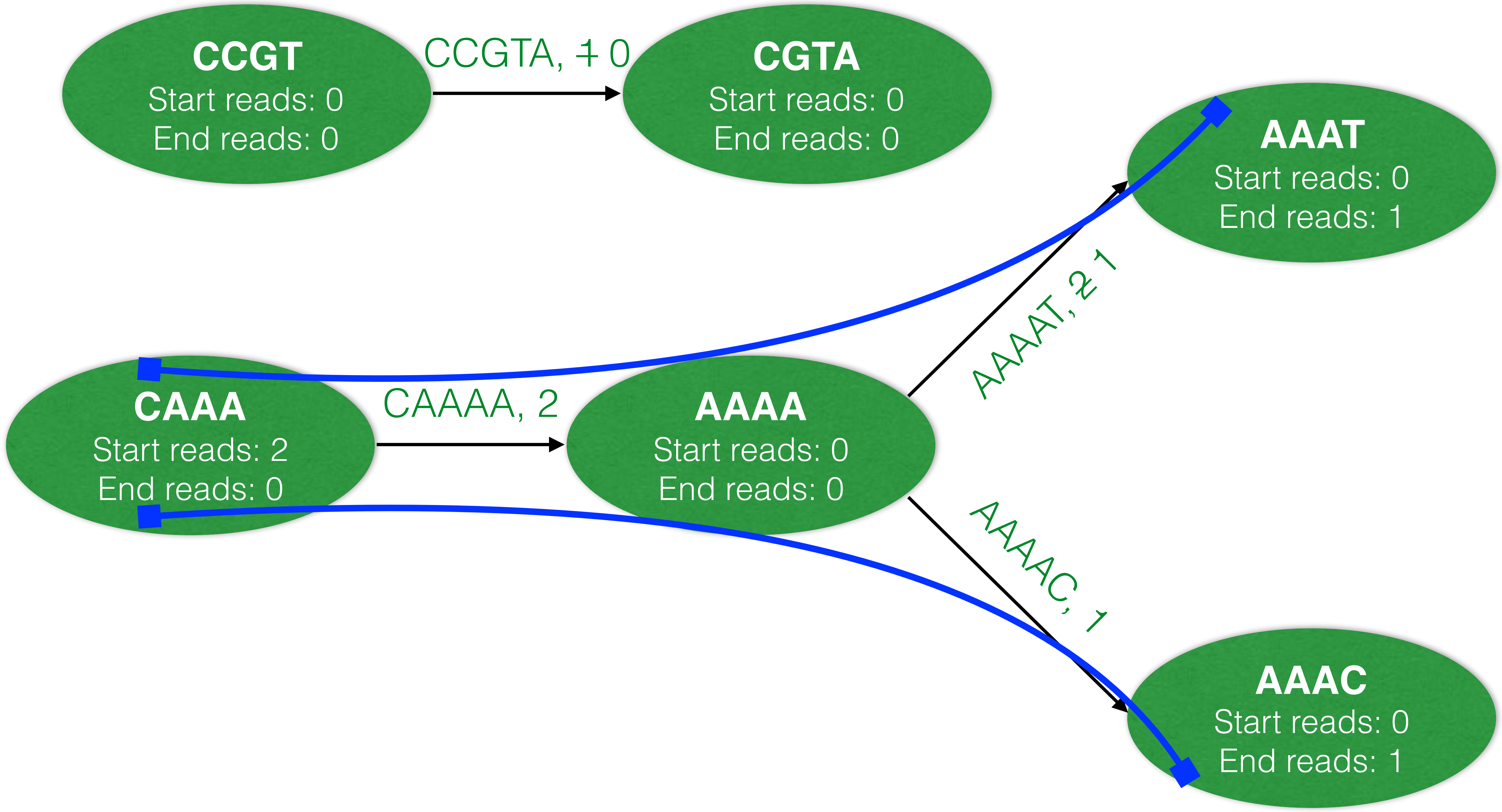
Error correction



Error correction



Error correction



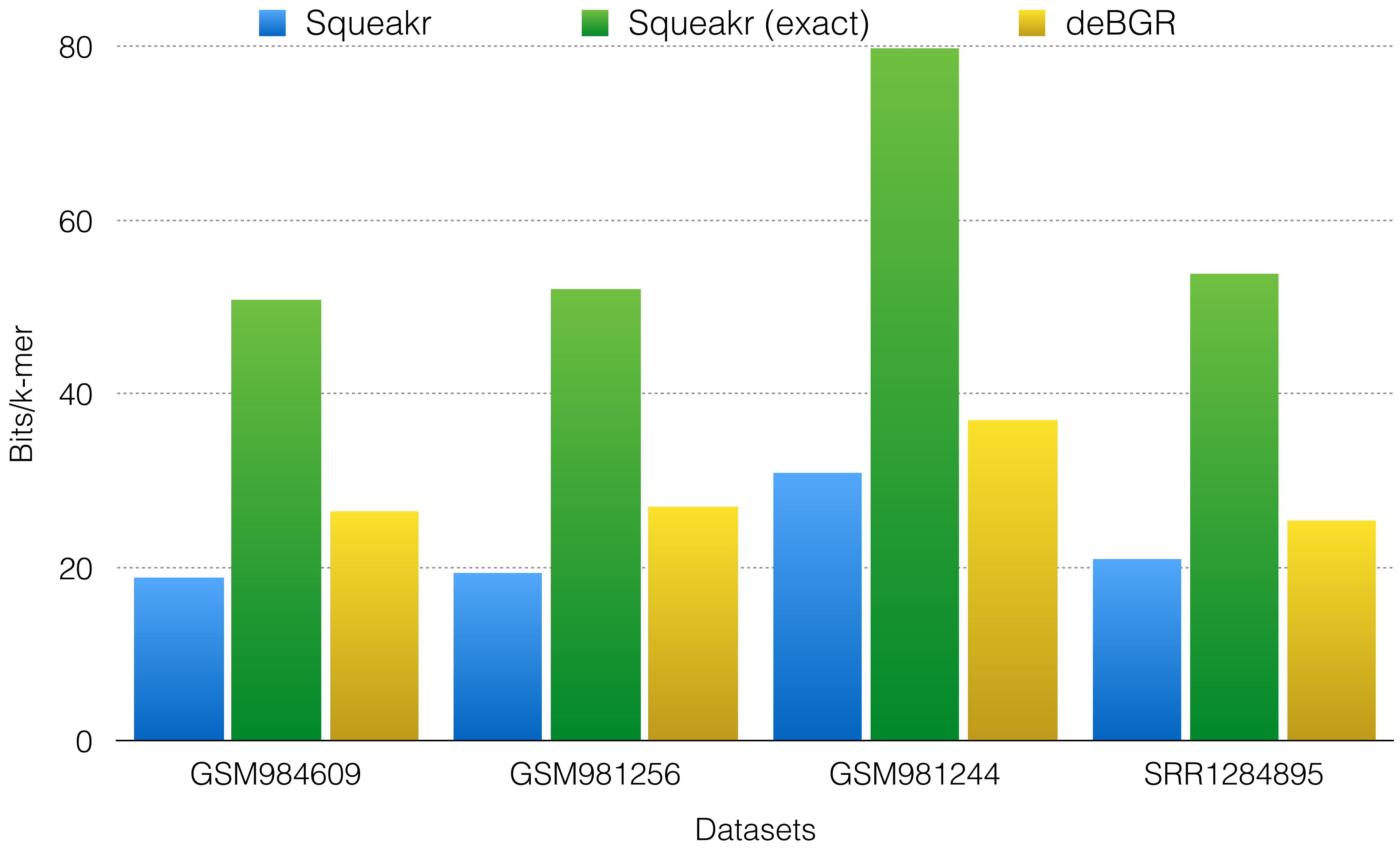
Error correction algorithm

- We use a standard work queue algorithm.
- We bootstrap with a set *C* of edges for which we know the abundance is correct.
- We then expand the set *C* of edges using the weighted de Bruijn graph invariant.
- Please refer to the paper for exact set of rules for error correction.
- Running time: $O(n \cdot \log(n) / \log(1/4\varepsilon))$.

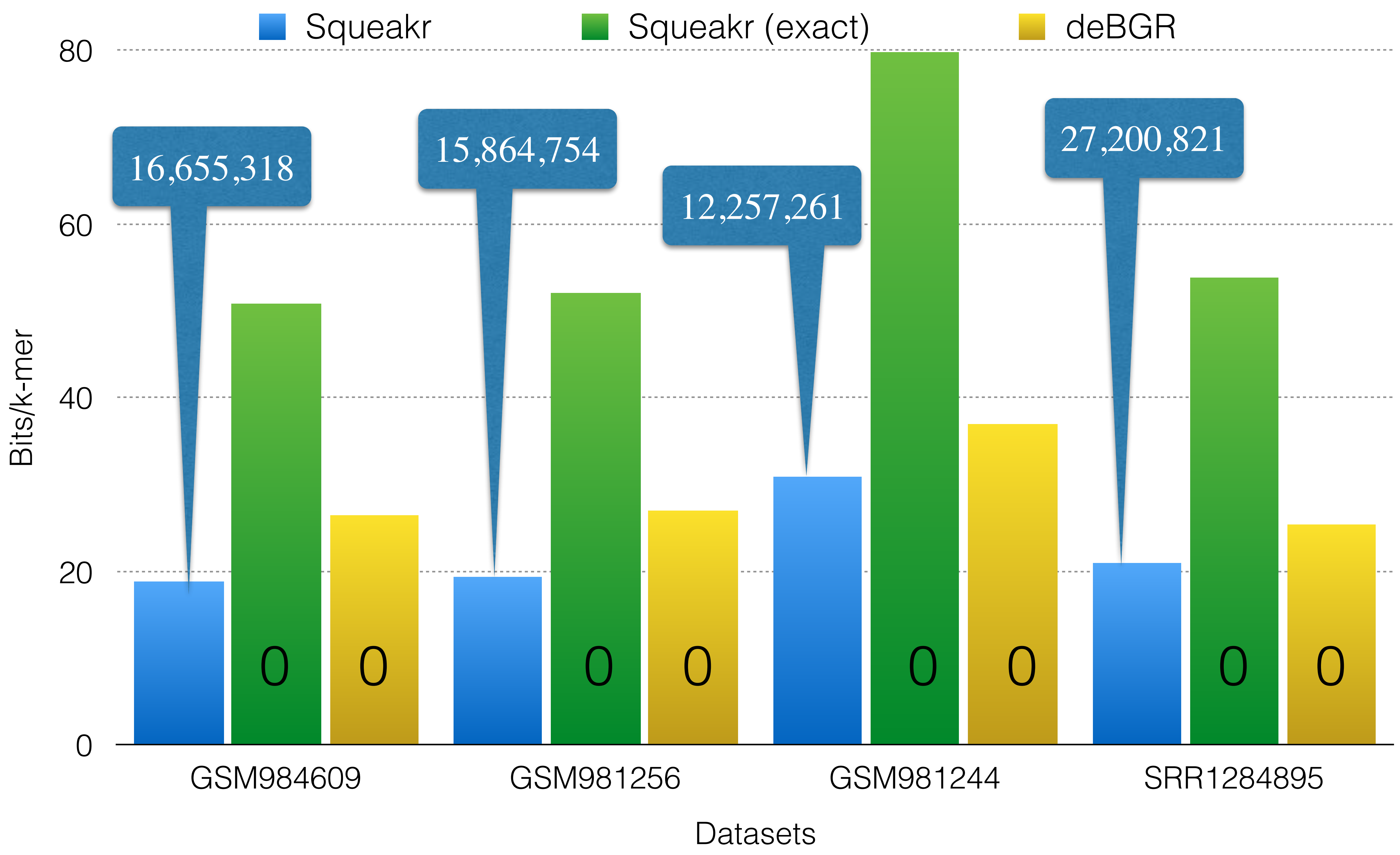
Datasets

Dataset	Size	#k-mer instances	#Distinct k-mers
GSM984609	26 GB	19,662,773,330	1,146,347,598
GSM981256	22 GB	16,470,774,825	1,118,090,824
GSM981244	43 GB	37,897,872,977	1,404,643,983
SRR1284895	33 GB	26,235,129,875	2,079,889,717

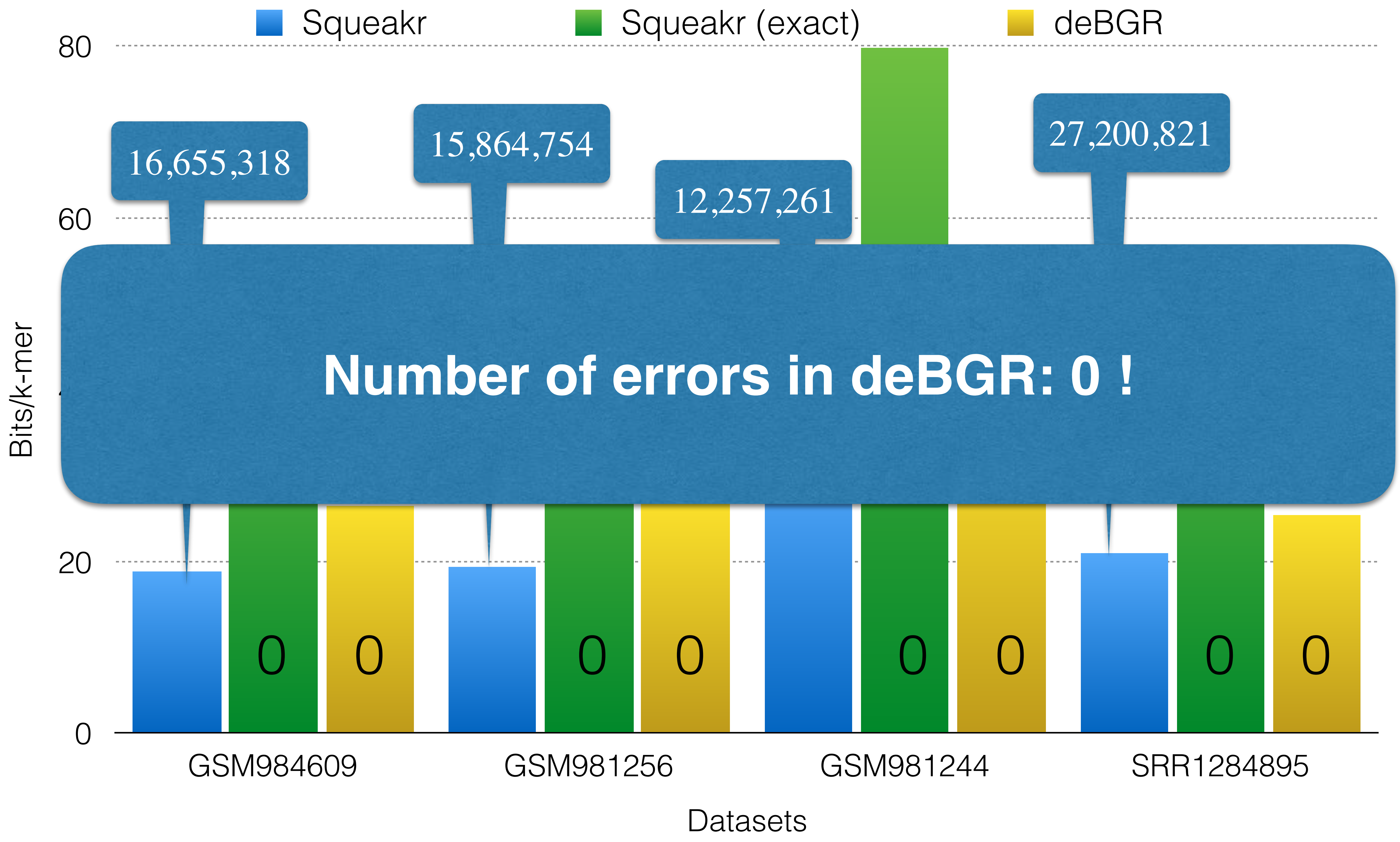
Space vs Accuracy



Space vs Accuracy



Space vs Accuracy



Number of errors in deBGR: 0 !

Conclusion

- Abundance information in important for many data analyses.
- But abundance information is also useful for providing higher de Bruijn graph structural guarantees.
- We show that the abundance information can be used to remove effectively all the errors in an approximate weighted de Bruijn graph representation.

https://github.com/splatlab/debgr