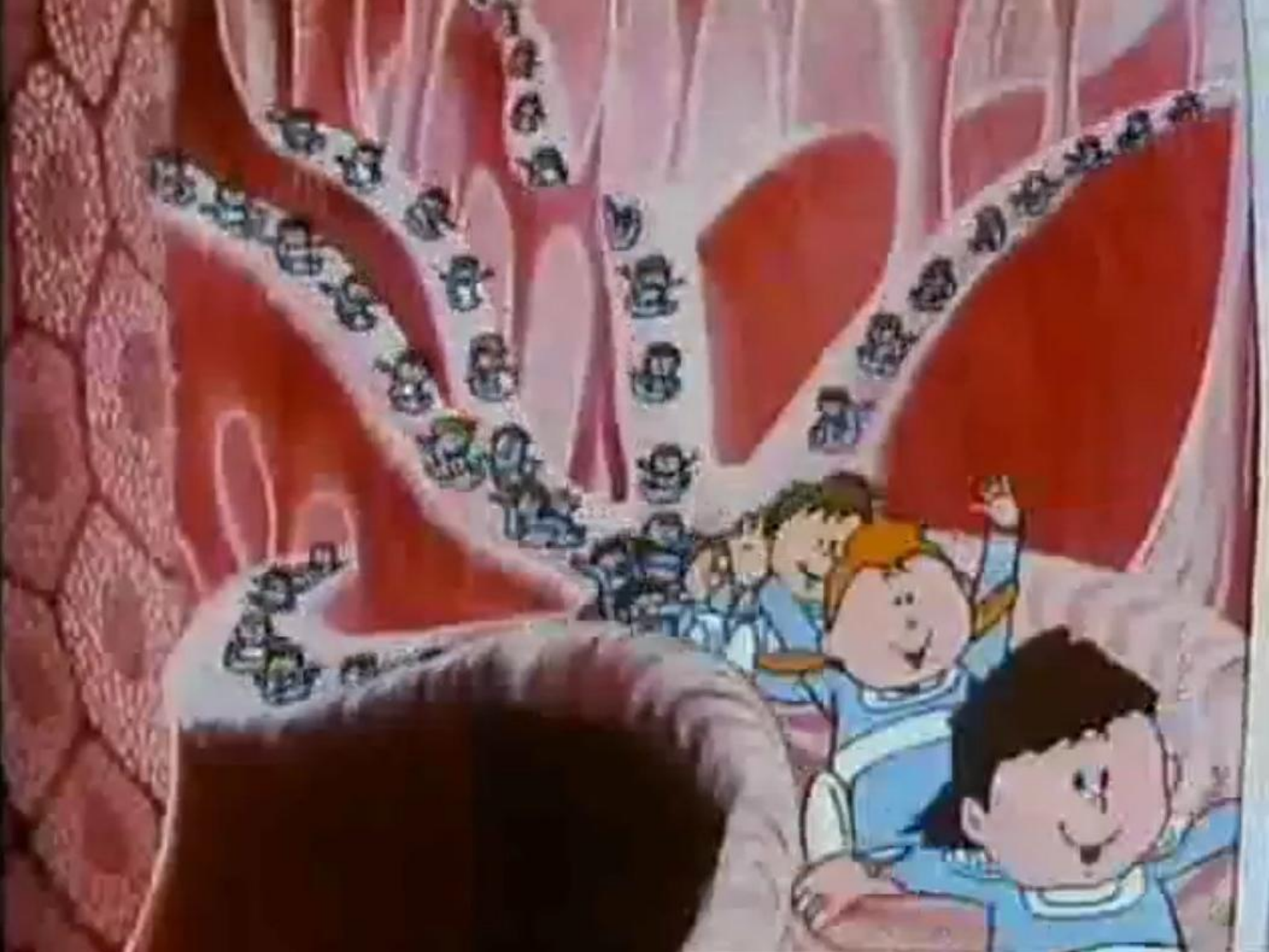
# Détection sans alignement de recombinaisons V(D)J multi-chaînes

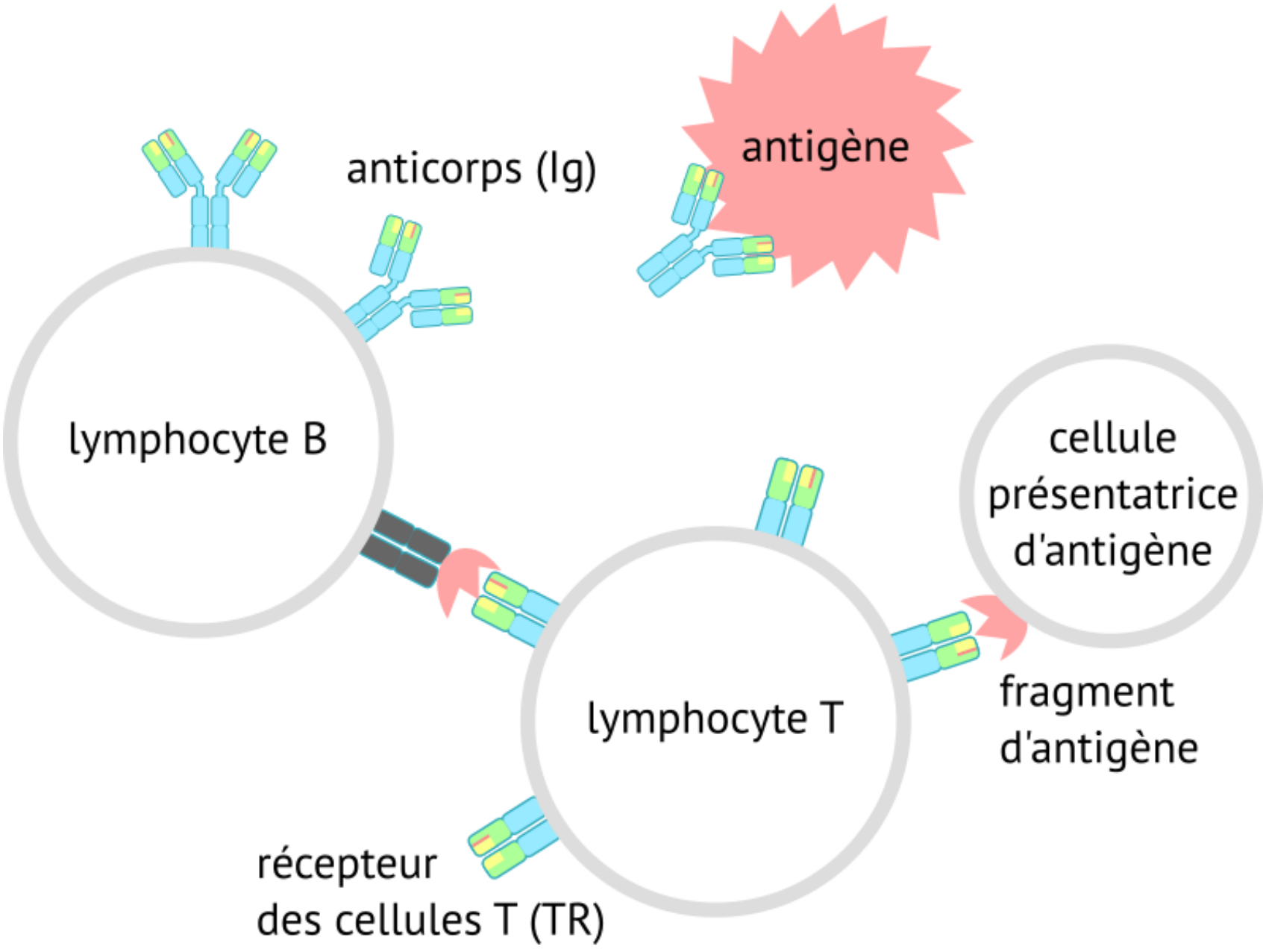
# Alignment-free detection of multi loci V(D)J recombinations

Mathieu Giraud, Mikaël Salson

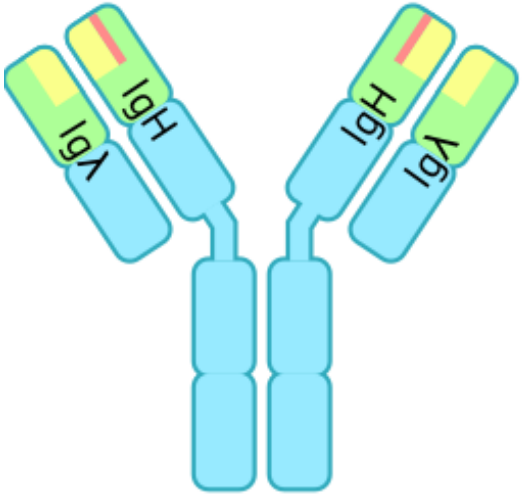
Univ. Lille, CNRS, CRIStAL, Inria

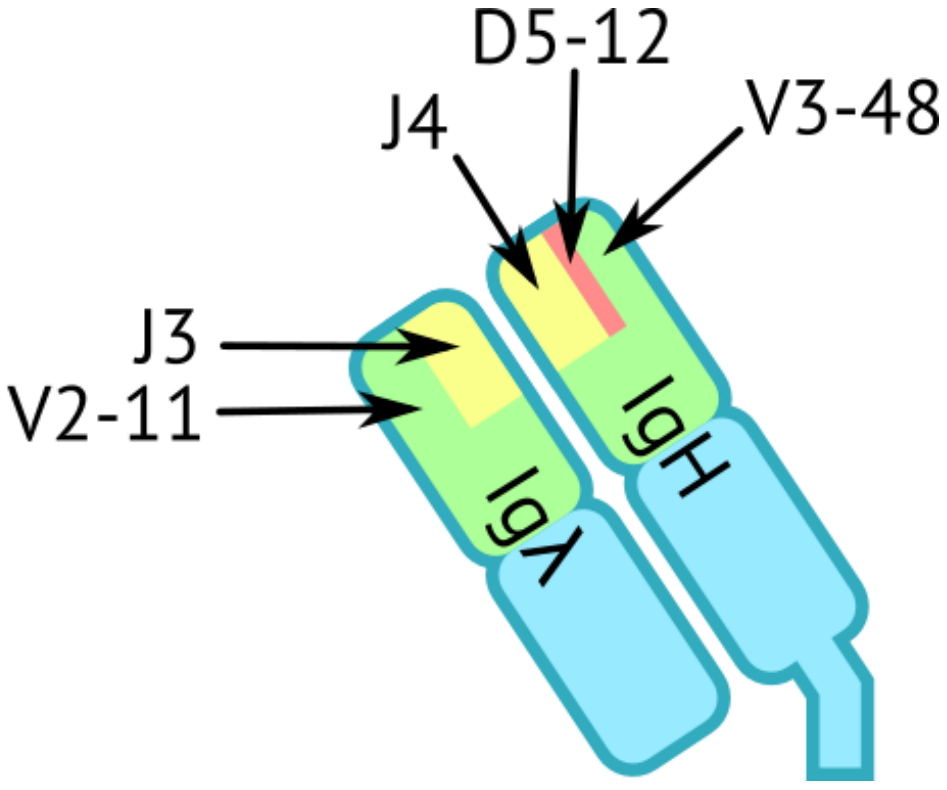


# The Adaptive Immune System



# TCR and Antibody Specificity – V(D)J Recombination

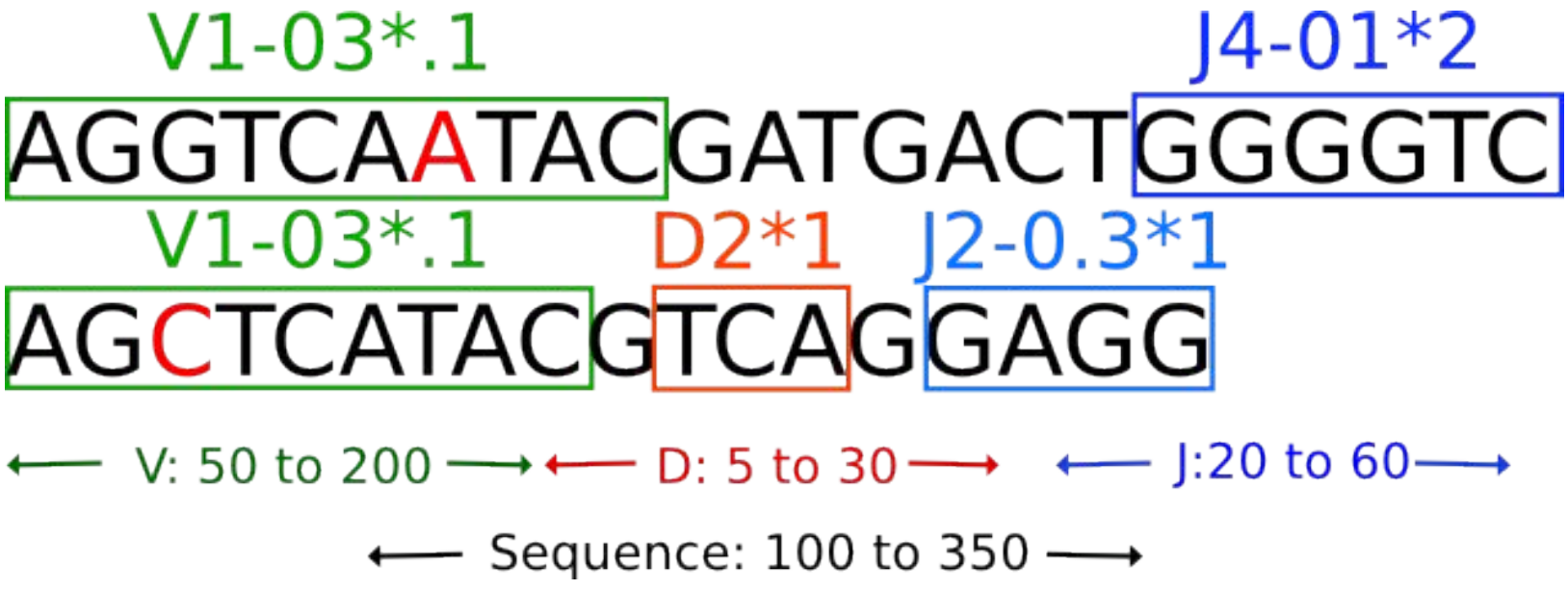


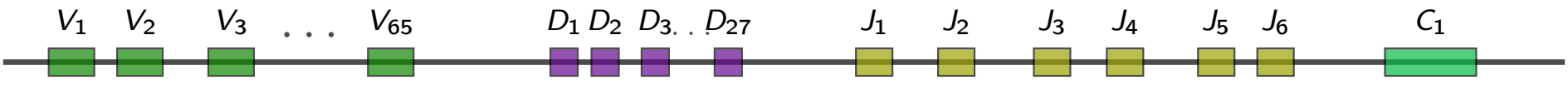

...GGAAGGGCAGAATTA...

V2-11 GGATGGG GAATTA J3

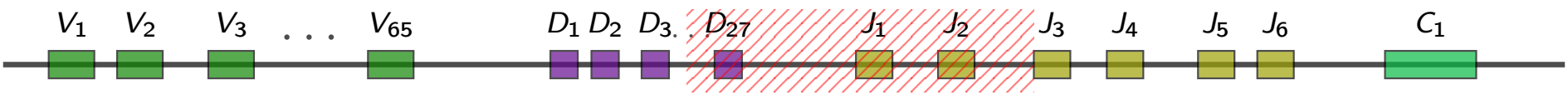
# TCR and Antibody Specificity – V(D)J Recombination



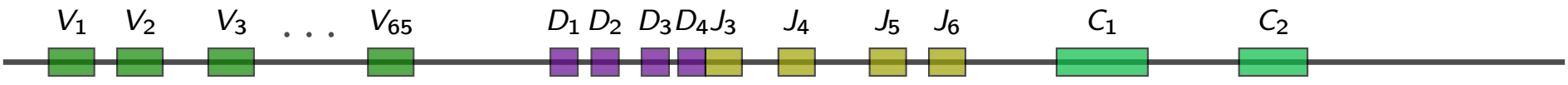
# V(D)J recombinations are responsible for receptor diversity



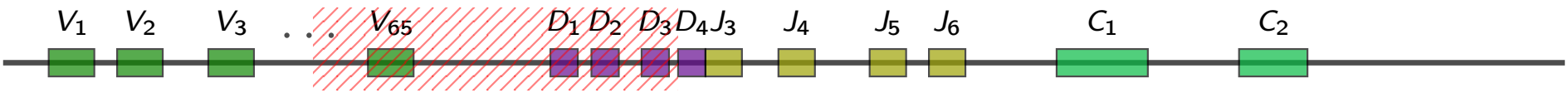
# V(D)J recombinations are responsible for receptor diversity



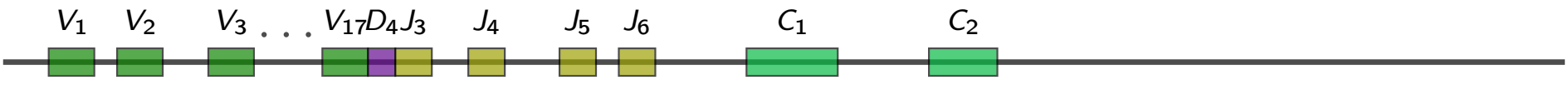
# V(D)J recombinations are responsible for receptor diversity



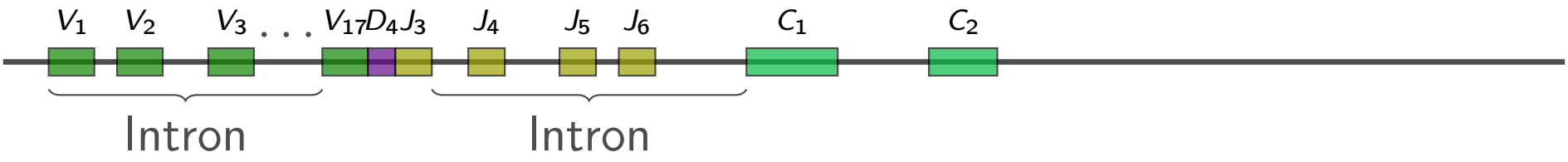
# V(D)J recombinations are responsible for receptor diversity



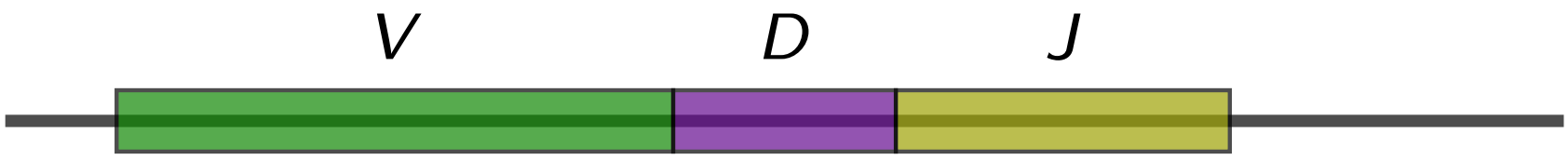
# V(D)J recombinations are responsible for receptor diversity



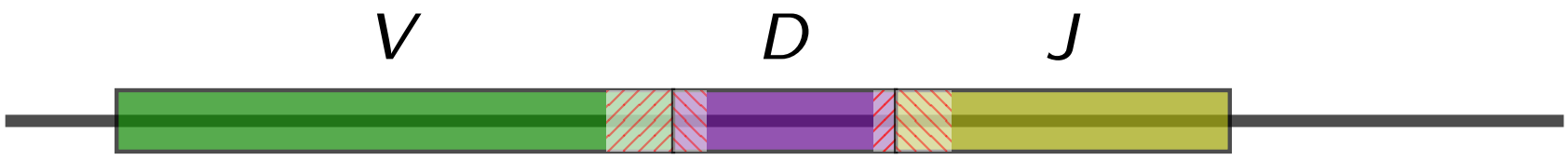
# V(D)J recombinations are responsible for receptor diversity



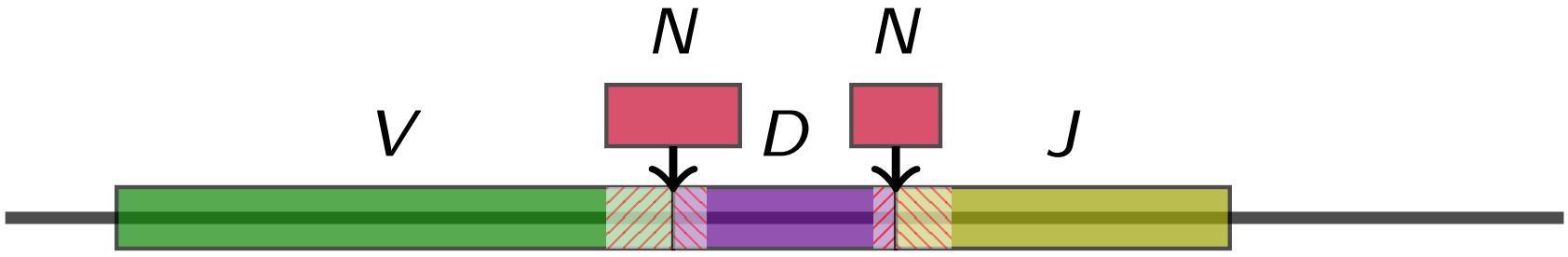
# V(D)J recombinations are responsible for receptor diversity



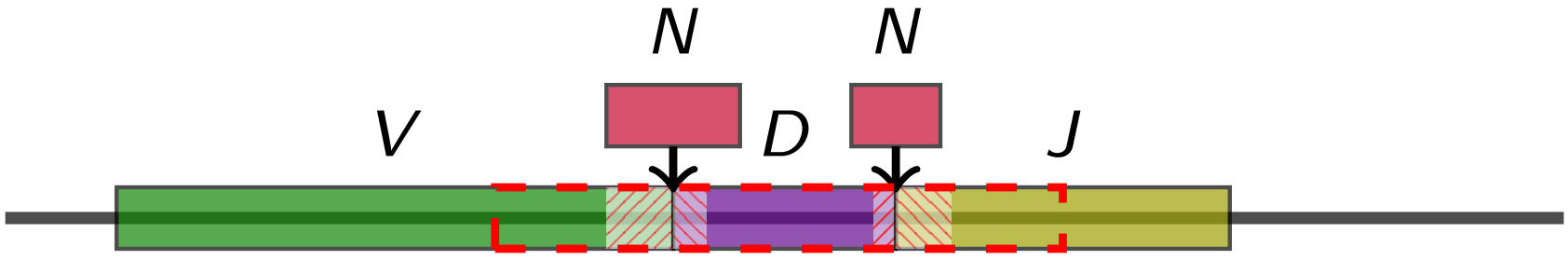
# V(D)J recombinations are responsible for receptor diversity



# V(D)J recombinations are responsible for receptor diversity

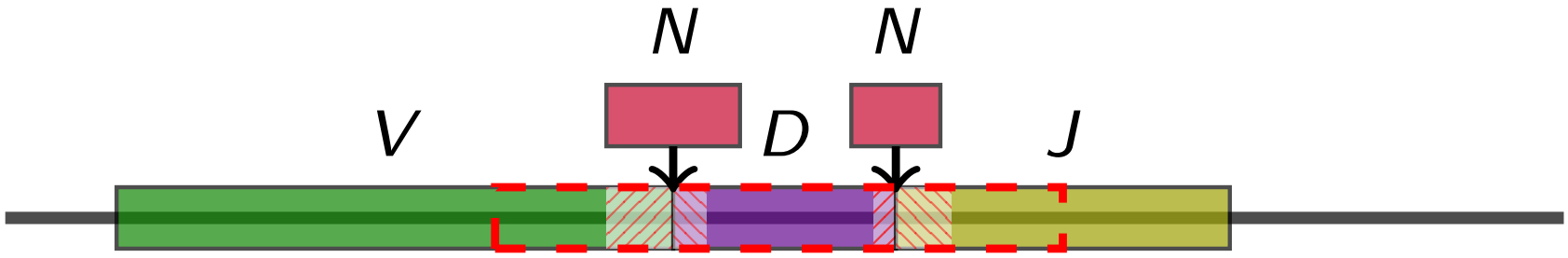


# V(D)J recombinations are responsible for receptor diversity

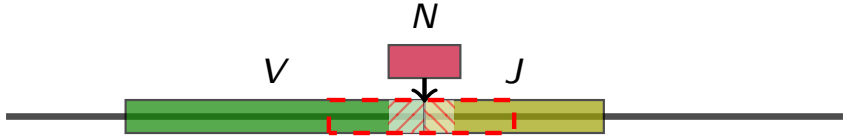



**Diversity region**

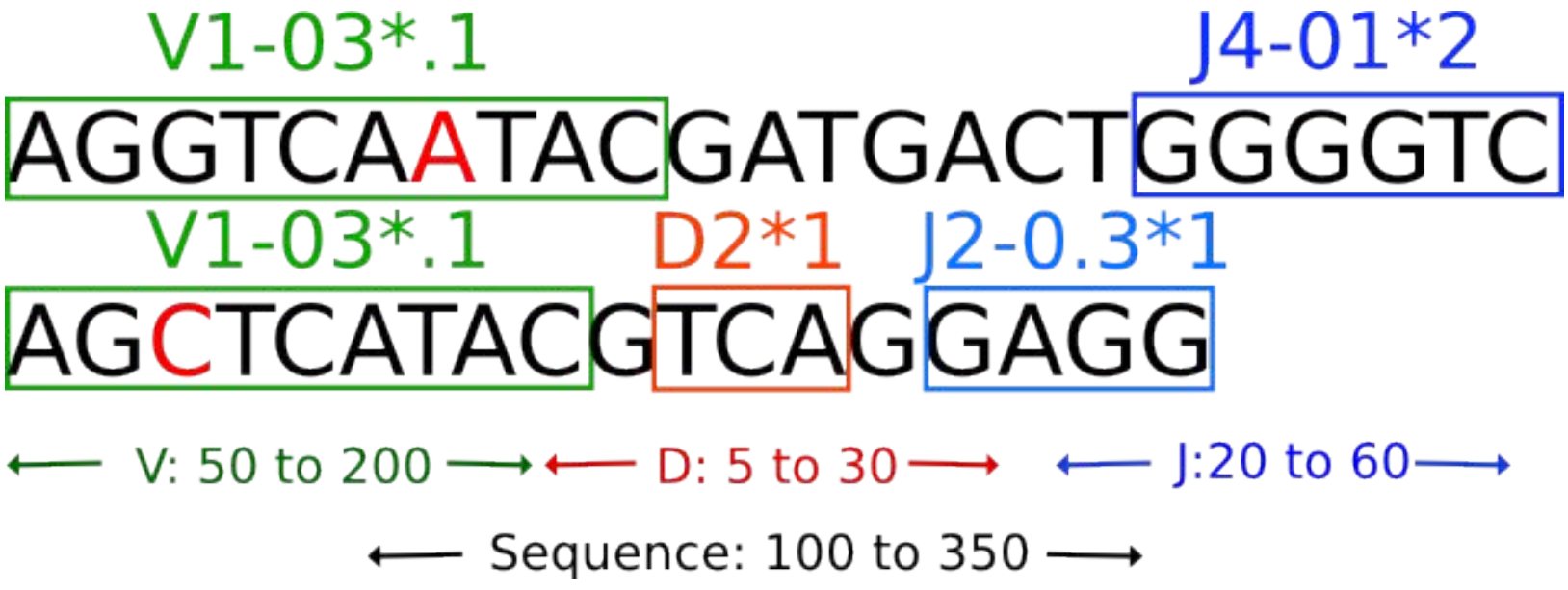
# V(D)J recombinations are responsible for receptor diversity




**Diversity region**

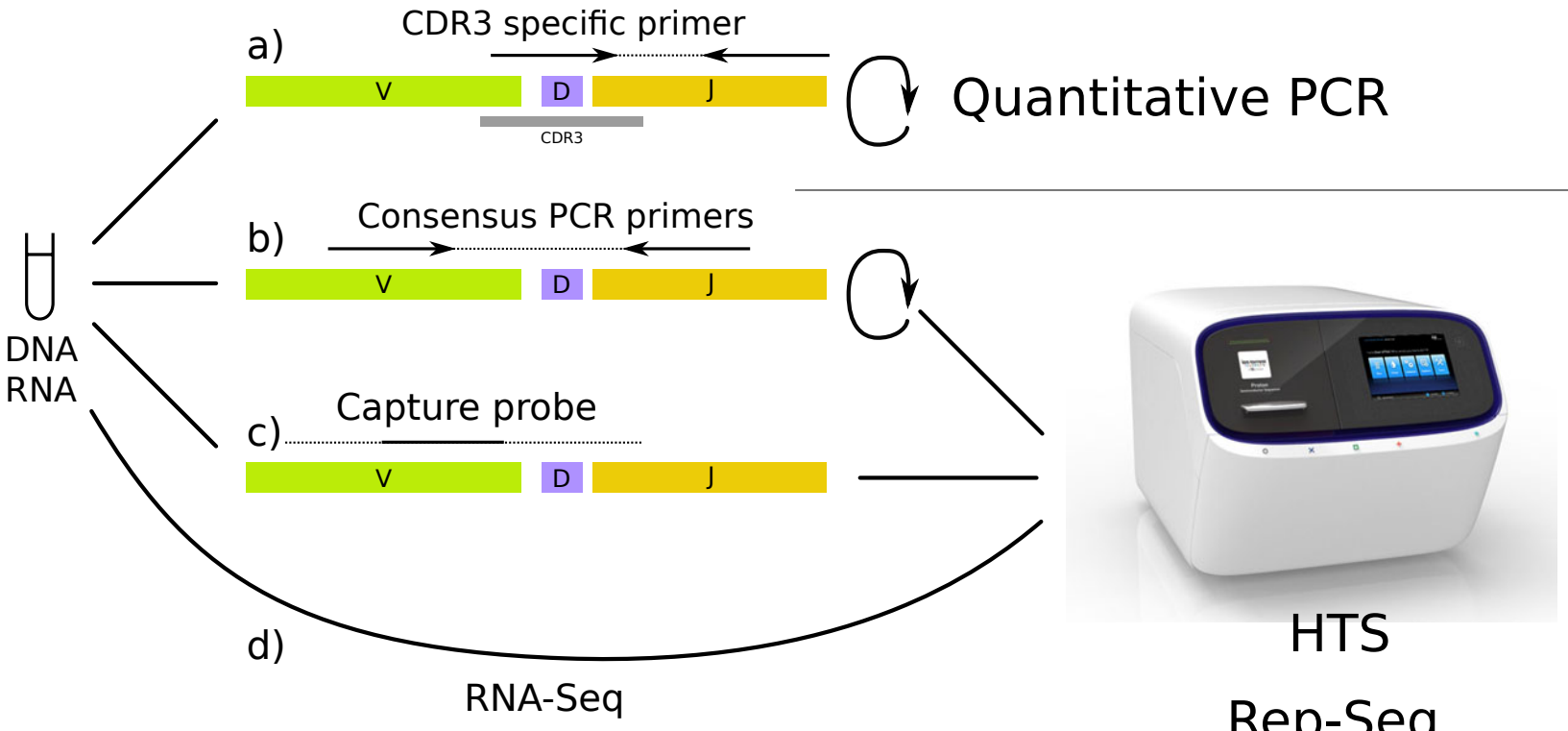


# TCR and Antibody Specificity – V(D)J Recombination



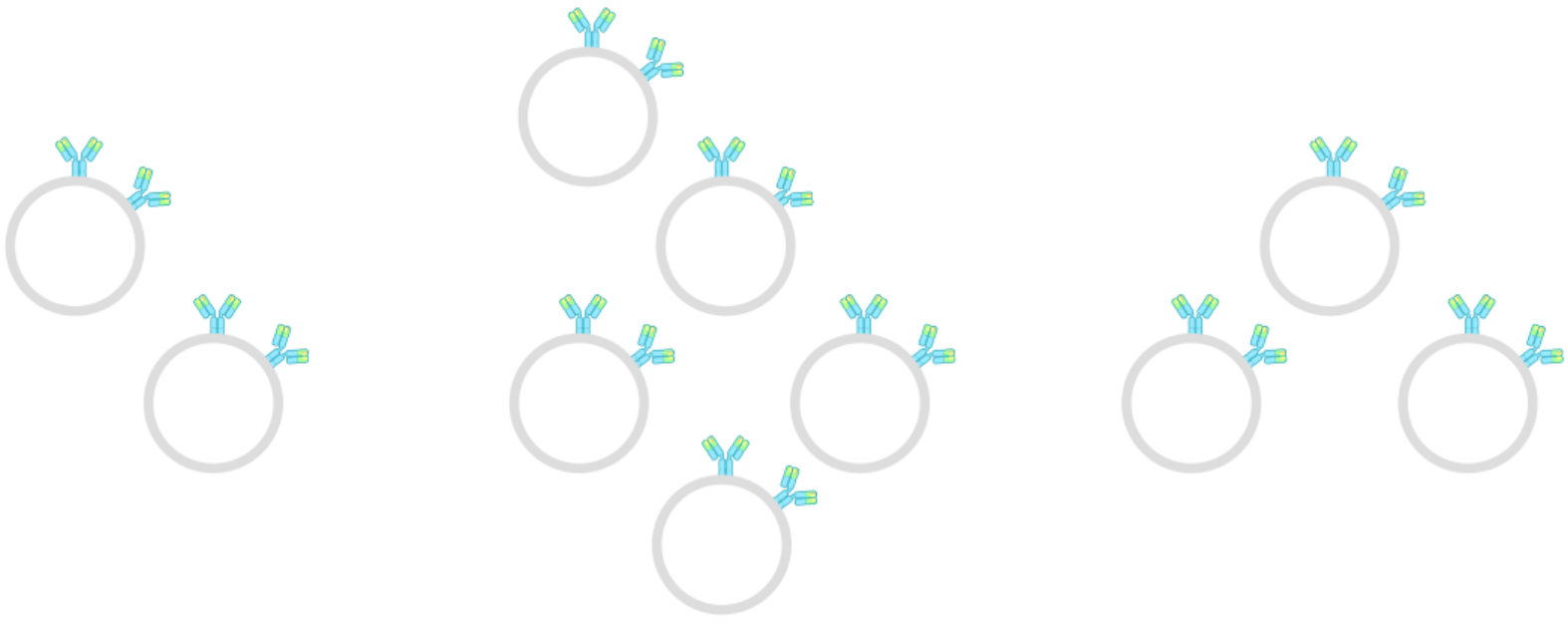
# Immune Repertoire Sequencing (RepSeq)

## Strategies – Sequencing millions of V(D)J recombinations from T-cells or B-cells



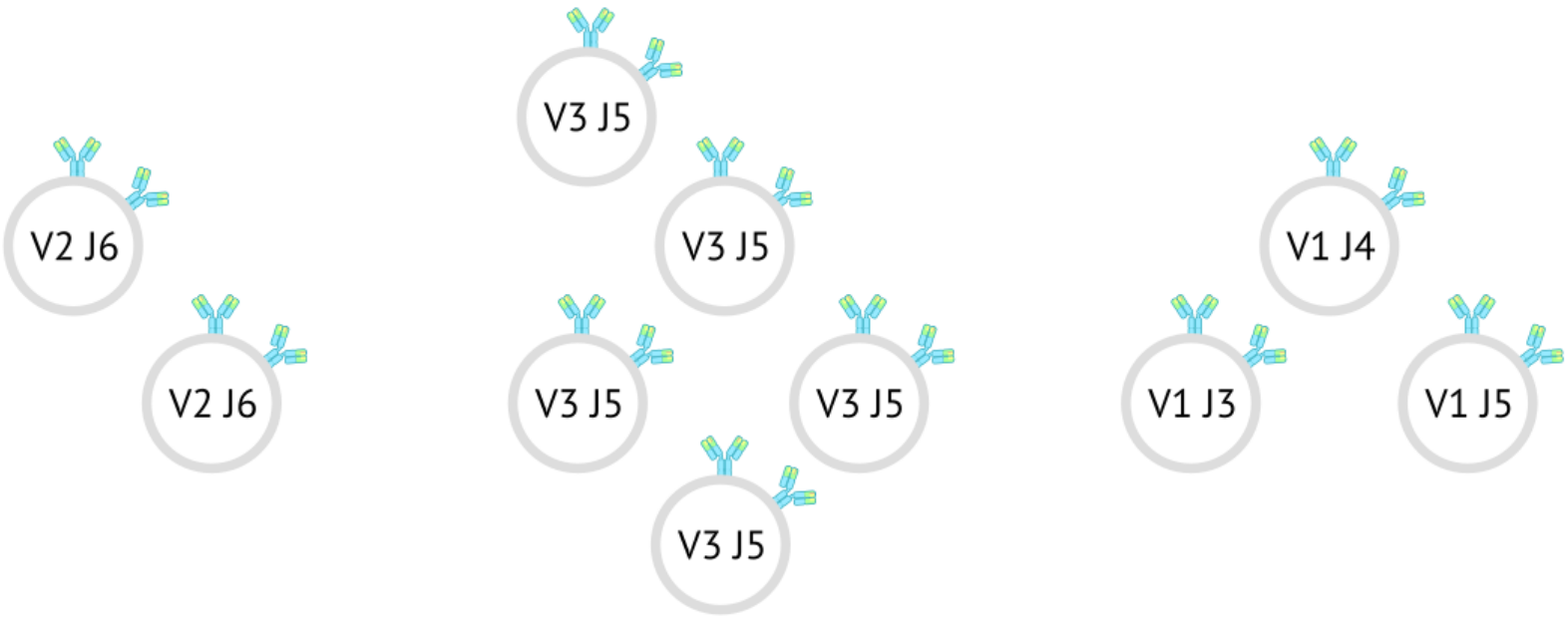
# Immune Repertoire Sequencing (RepSeq)

## Identification of all VDJ recombinations



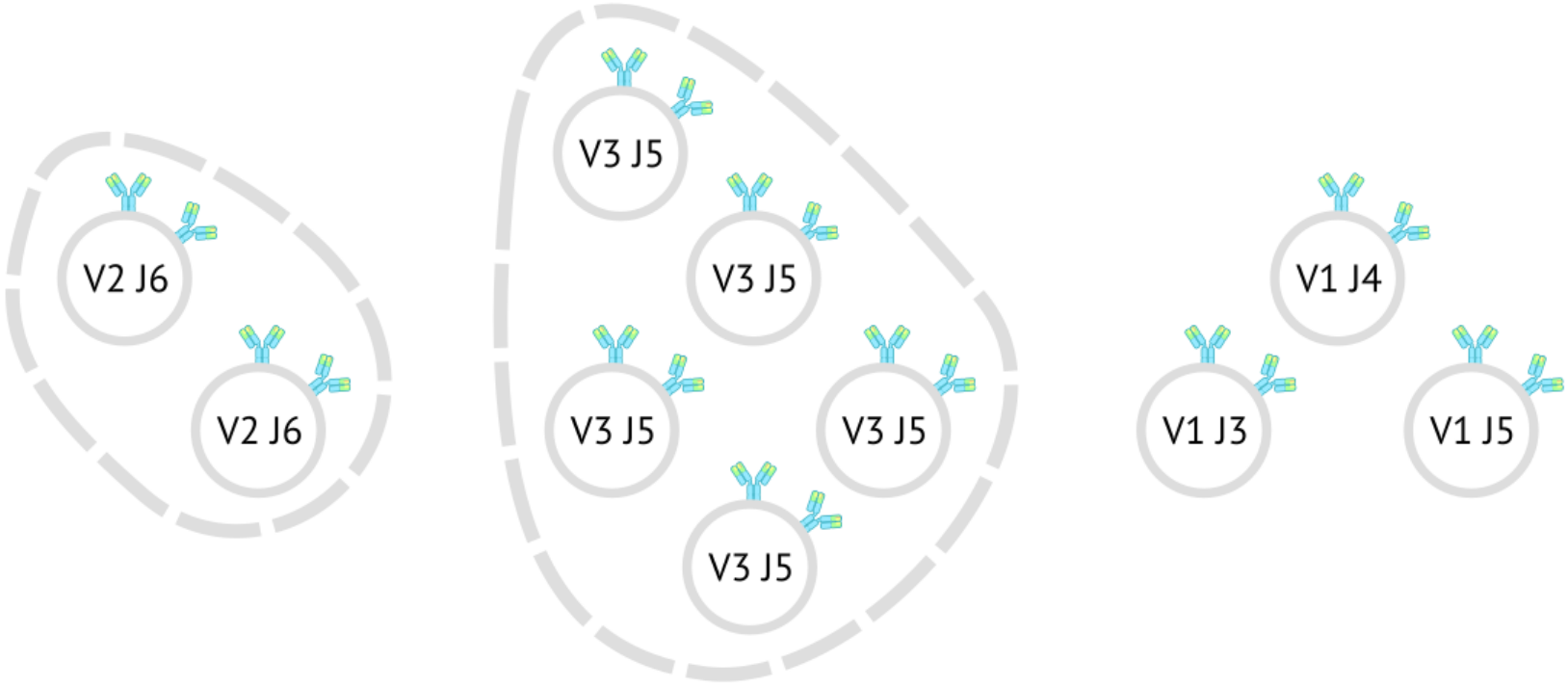
# Immune Repertoire Sequencing (RepSeq)

## Identification of all VDJ recombinations



# Immune Repertoire Sequencing (RepSeq)

## Identification of all VDJ recombinations




20% 50% 30%

# Vidjil

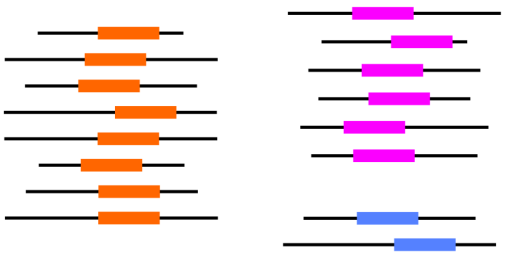
High-throughput Repertoire Sequencing (RepSeq) analysis

**Web Application**

Patient database

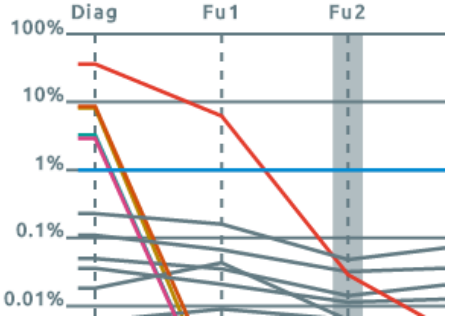
Server

**Vidjil-algo**



C++

Client




Javascript, d3.js

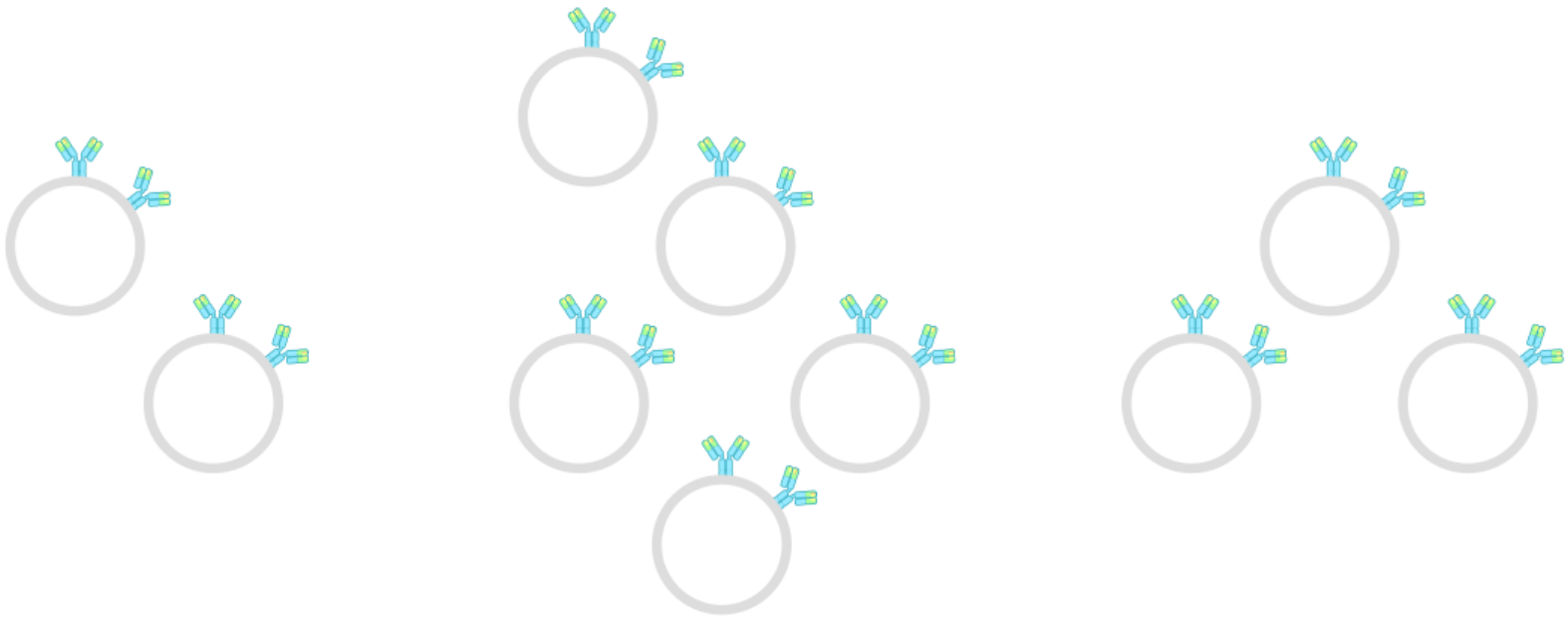
Python, web2py, AJAX

- code on `http://git.vidjil.org/`
- open-source (GPL v3), public issue tracker (Gitlab)
- continuous integration, $> 2,000$ unit and functional tests

Duez et al., PLOS One, 2016

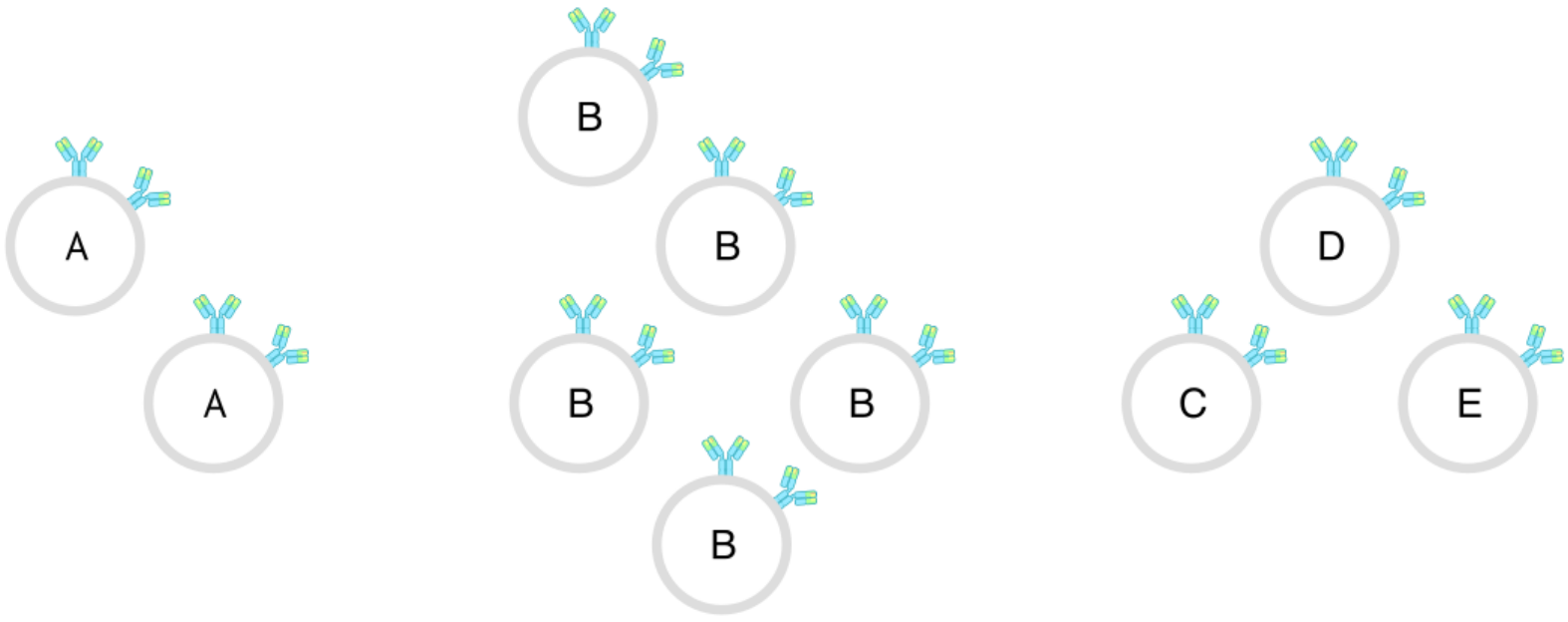
# Immune Repertoire Sequencing (RepSeq)

## Clone clustering



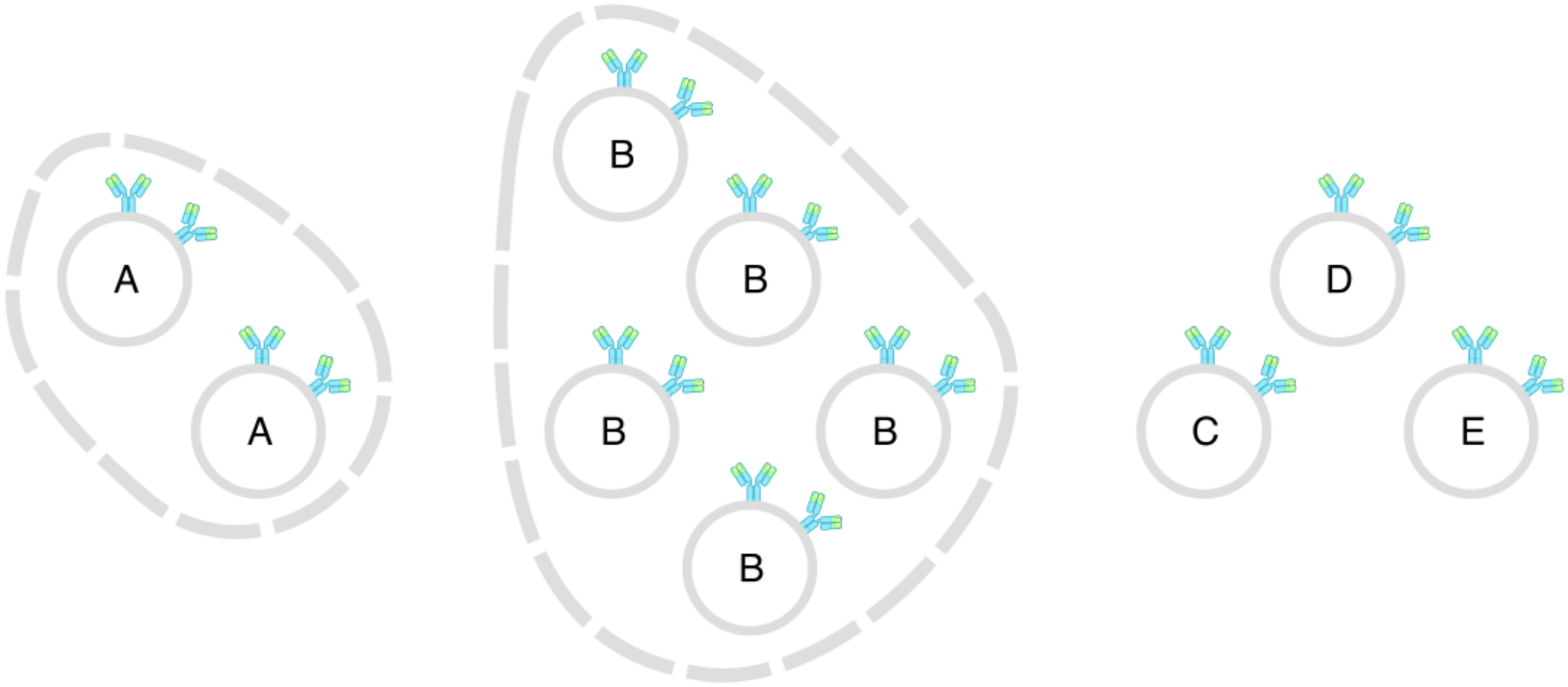
# Immune Repertoire Sequencing (RepSeq)

## Clone clustering



# Immune Repertoire Sequencing (RepSeq)

## Clone clustering



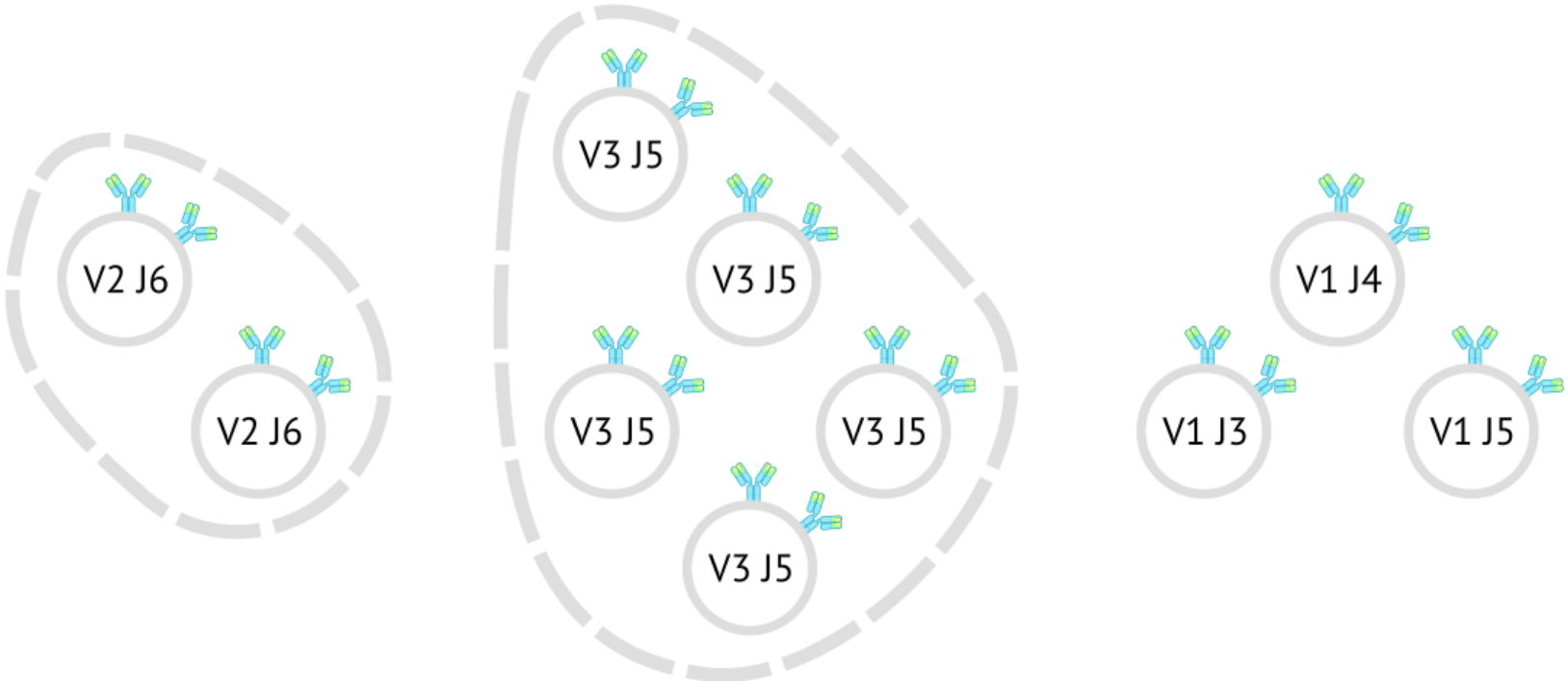

20% 50% 30%

1 000 000 VDJ = 100 s

Giraud, Salson et al., BMC Genomics, 2014

# Immune Repertoire Sequencing (RepSeq)

## Clone clustering




20% 50% 30%

1 000 000 VDJ = 100 s

Giraud, Salson et al., BMC Genomics, 2014

# Fast identification of a window centered on the CDR3

## Clone clustering

parts of V genes

```
ACAC CACG ACGG CGGC GGCC
GCCG TCTT CTTC TTCC TCCA
CCAA CAAC AACC ACCT CCTT
CTTG TTGG TGGA ACTT  ...
```

parts of J genes

```
ATAC TACT ACTT CCAG CAGC
AGCA GCAC TGGG GGGC GGCA
GCAA CAAG AAGA AGAG GAGT
AGTT GTTG TTGG  ...
```

```
ACACGGCCGTGTATTACTGTGCGAGAGAGCTGAATACTTCCAGCACTGGGGCC
```

# Fast identification of a window centered on the CDR3

## Clone clustering

parts of V genes

```
ACAC CACG ACGG CGGC GGCC
GCCG TCTT CTTC TTCC TCCA
CCAA CAAC AACC ACCT CCTT
CTTG TTGG TGGA ACTT  ...
```

parts of J genes

```
ATAC TACT ACTT CCAG CAGC
AGCA GCAC TGGG GGGC GGCA
GCAA CAAG AAGA AGAG GAGT
AGTT GTTG TTGG  ...
```

# Fast identification of a window centered on the CDR3

## Clone clustering

parts of V genes

```
ACAC CACG ACGG CGGC GGCC
GCCG TCTT CTTC TTCC TCCA
CCAA CAAC AACC ACCT CCTT
CTTG TTGG TGGA ACTT  ...
```

parts of J genes

```
ATAC TACT ACTT CCAG CAGC
AGCA GCAC TGGG GGGC GGCA
GCAA CAAG AAGA AGAG GAGT
AGTT GTTG TTGG  ...
```


$O(n)$ alignment-free V(D)J detection algorithm

Giraud, Salson *et al*, BMC Genomics, 2014

# Fast identification of a window centered on the CDR3

## Clone clustering

parts of V genes

```
ACAC CACG ACGG CGGC GGCC
GCCG TCTT CTTC TTCC TCCA
CCAA CAAC AACC ACCT CCTT
CTTG TTGG TGGA ACTT  ...
```

parts of J genes

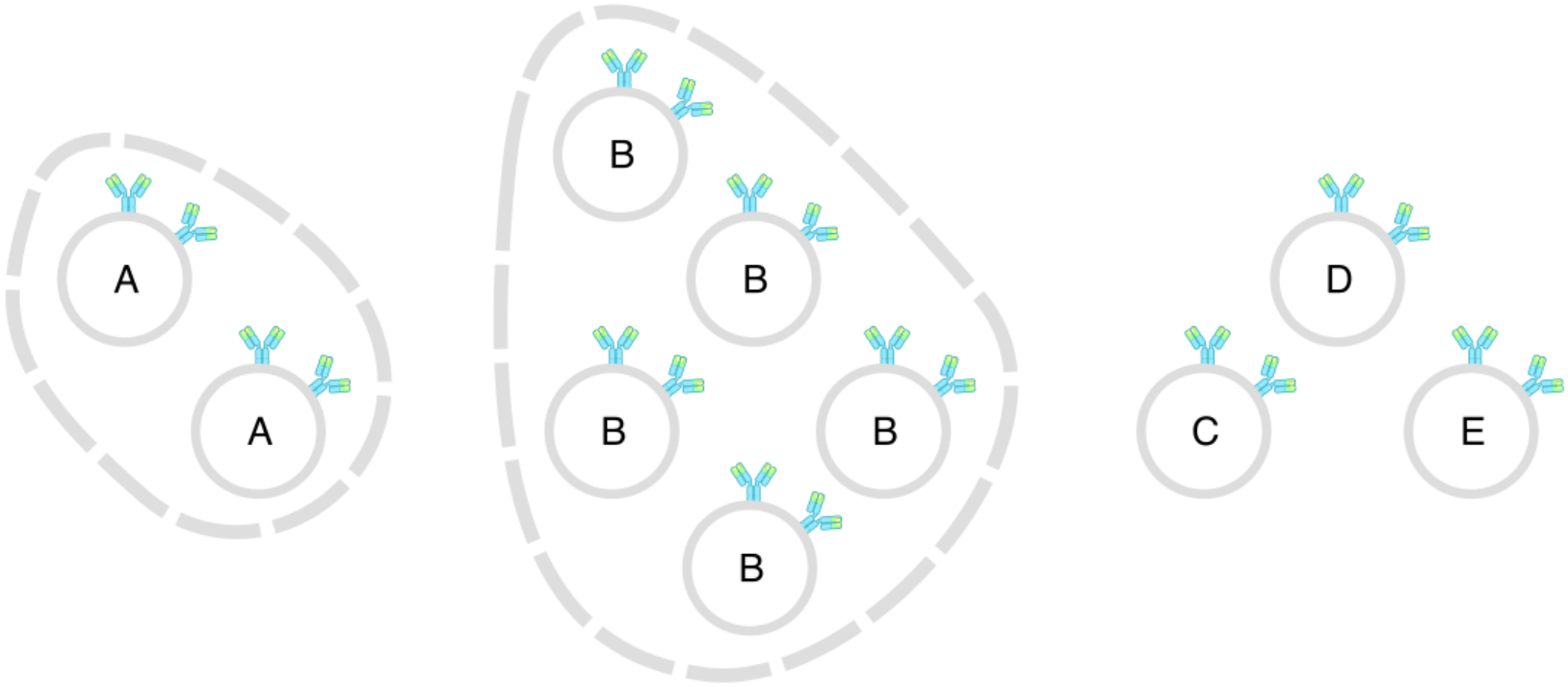
```
ATAC TACT ACTT CCAG CAGC
AGCA GCAC TGGG GGGC GGCA
GCAA CAAG AAGA AGAG GAGT
AGTT GTTG TTGG  ...
```


$O(n)$ alignment-free V(D)J detection algorithm

Giraud, Salson *et al*, BMC Genomics, 2014

# Immune Repertoire Sequencing (RepSeq)

## Clone clustering




20% 50% 30%

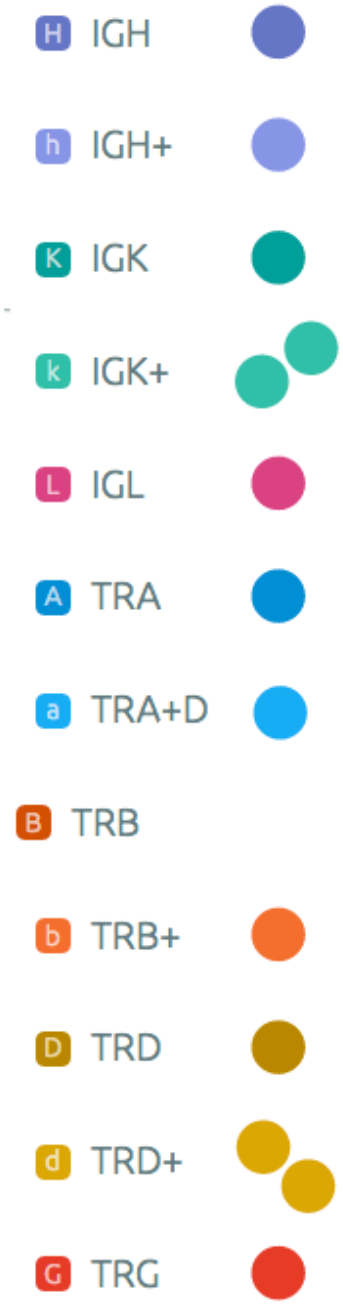
1 000 000 VDJ = 100 s

Giraud, Salson et al., BMC Genomics, 2014

# Vidjil-algo

## analyses recombinations on all human TR/Ig locus

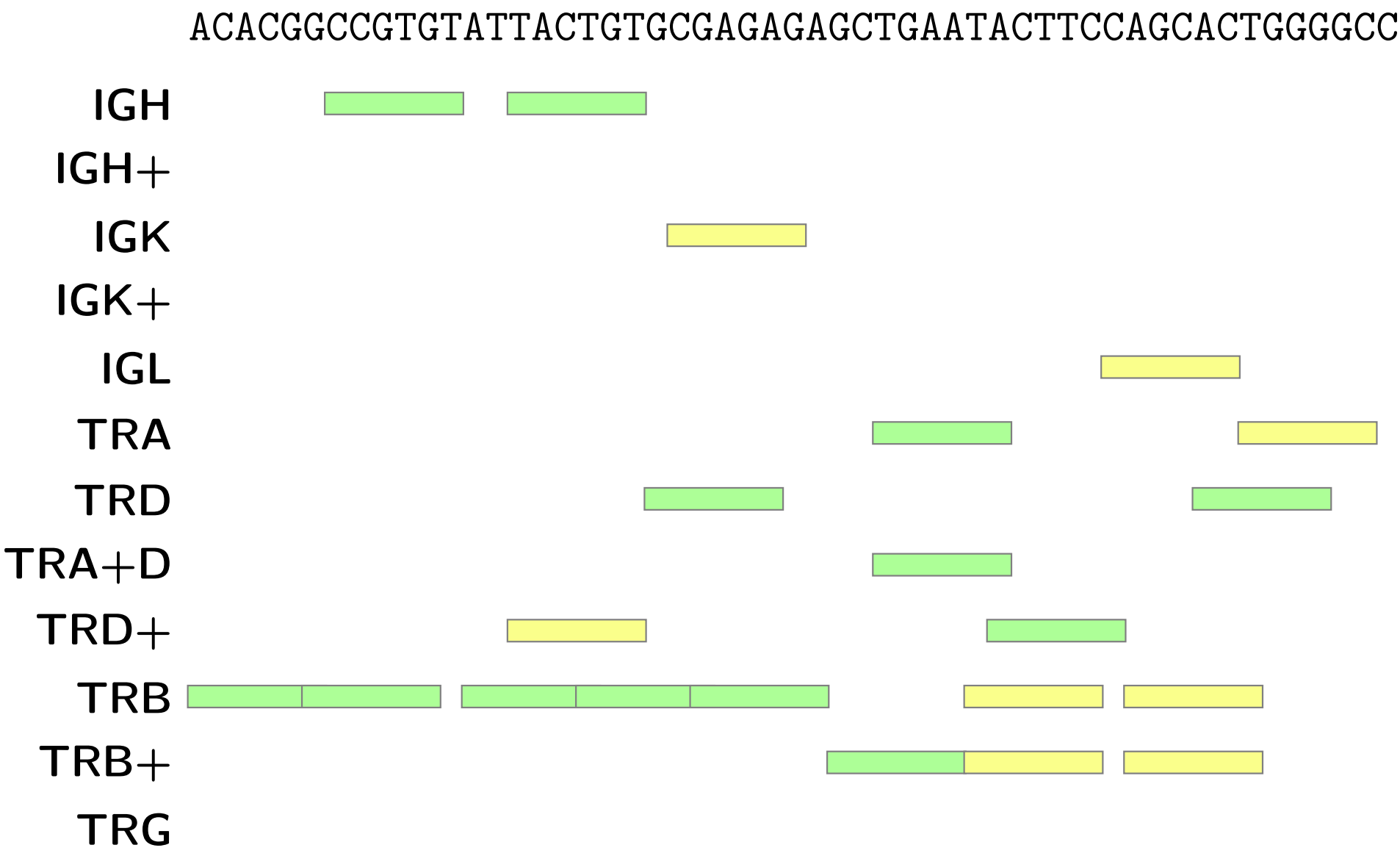
| complete recombinations | | incomplete/special recombinations | |
|---|---|---|---|
| **TRA** | Va-Ja | | |
| **TRB** | Vb-(Db)-Jb | **TRB+** | Db-Jb |
| **TRD** | Vd-(Dd)-Jd | **TRD+** | Vd-Dd3, Dd2-(Dd)-Jd, Dd2-Dd3 |
| | | **TRA+D** | Vd-(Dd)-Ja, Dd-Ja |
| **TRG** | Vg-Jg | | |
| **IGH** | Vh-(Dh)-Jh | **IGH+** | Dh-Jh |
| **IGL** | Vl-Jl | | |
| **IGK** | Vk-Jk | **IGK+** | Vk-KDE, INTRON-KDE |



# One pass for each recombination system

ACACGGCCGTGTATTACTGTGCGAGAGAGCTGAATACTTCCAGCACTGGGGCC

# One pass for each recombination system



How could we find
a V(D)J recombination (if any)
in a single pass?

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

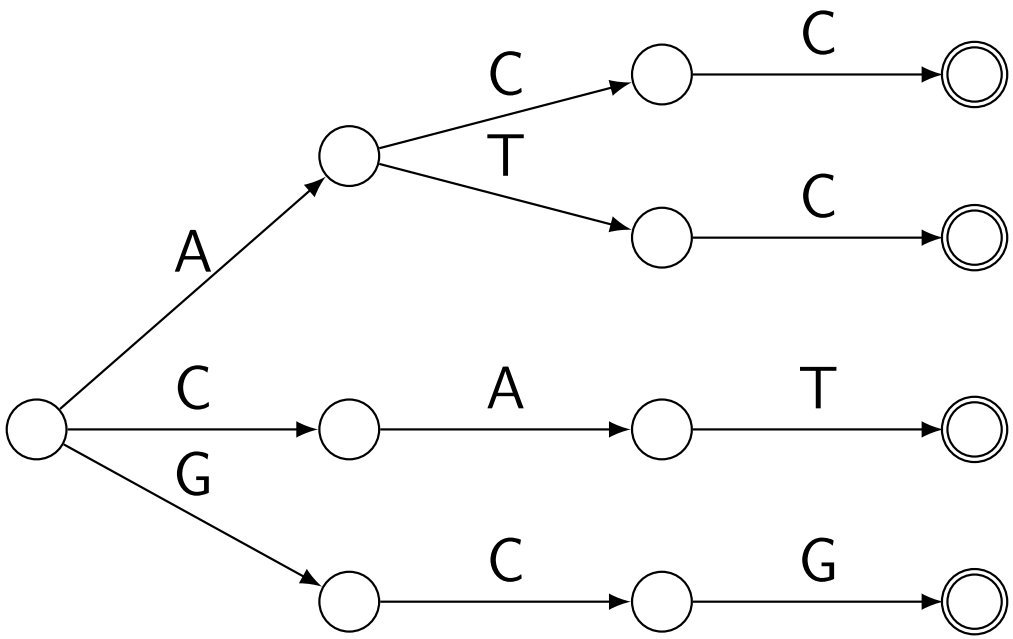
Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$

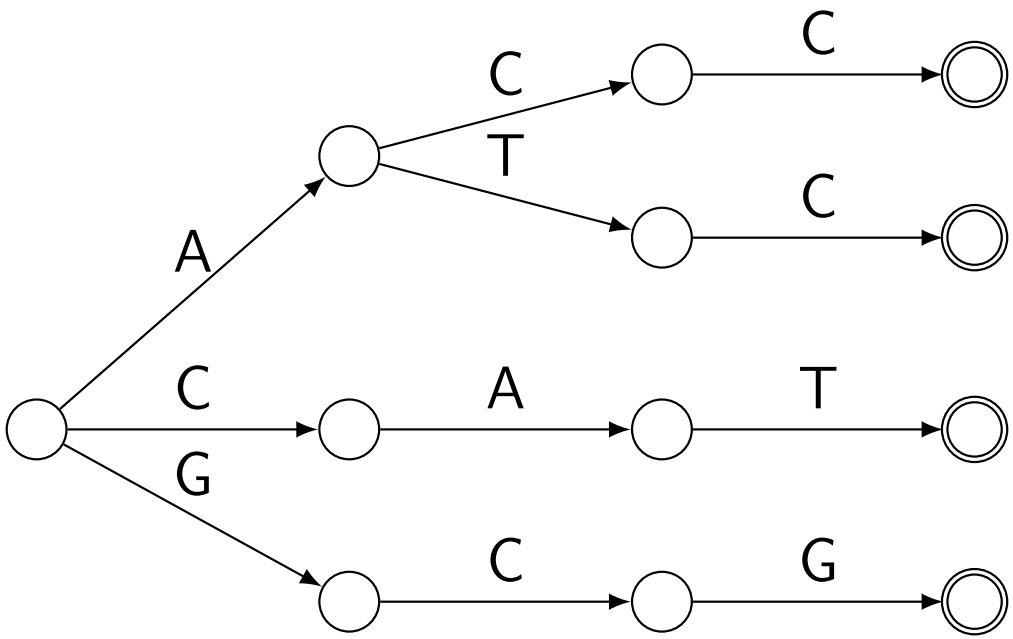


# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$



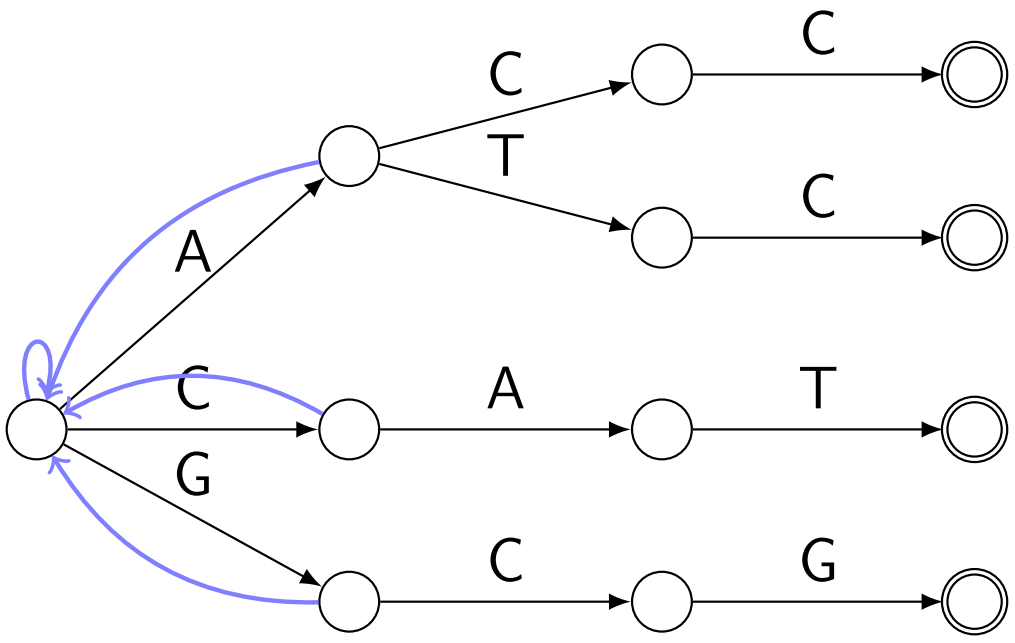

**Failure function**: returns the longest proper suffix accessible from the initial state

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$



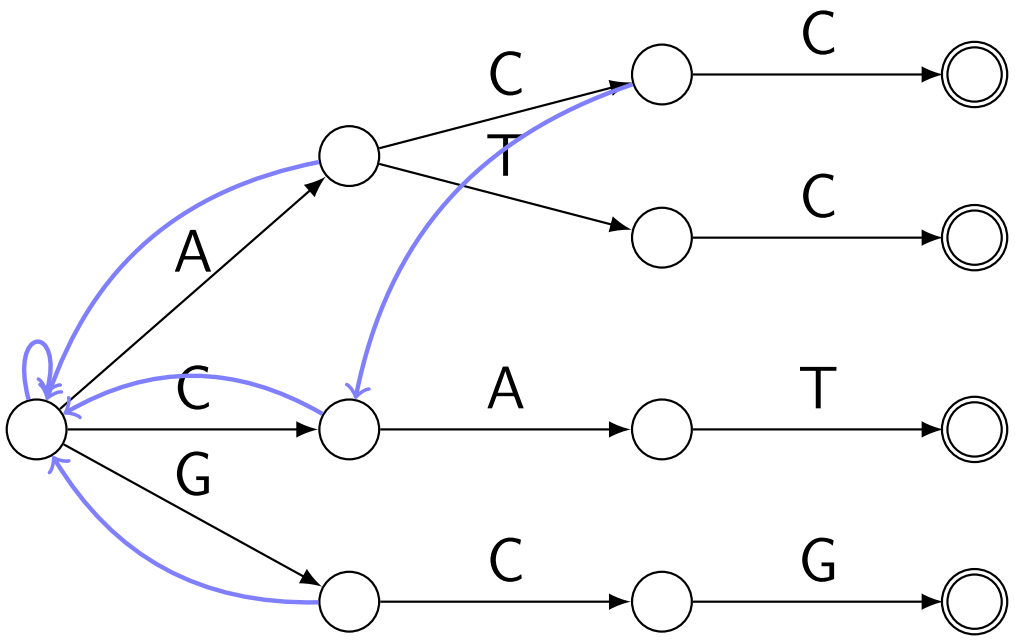

**Failure function**: returns the longest proper suffix accessible from the initial state

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$



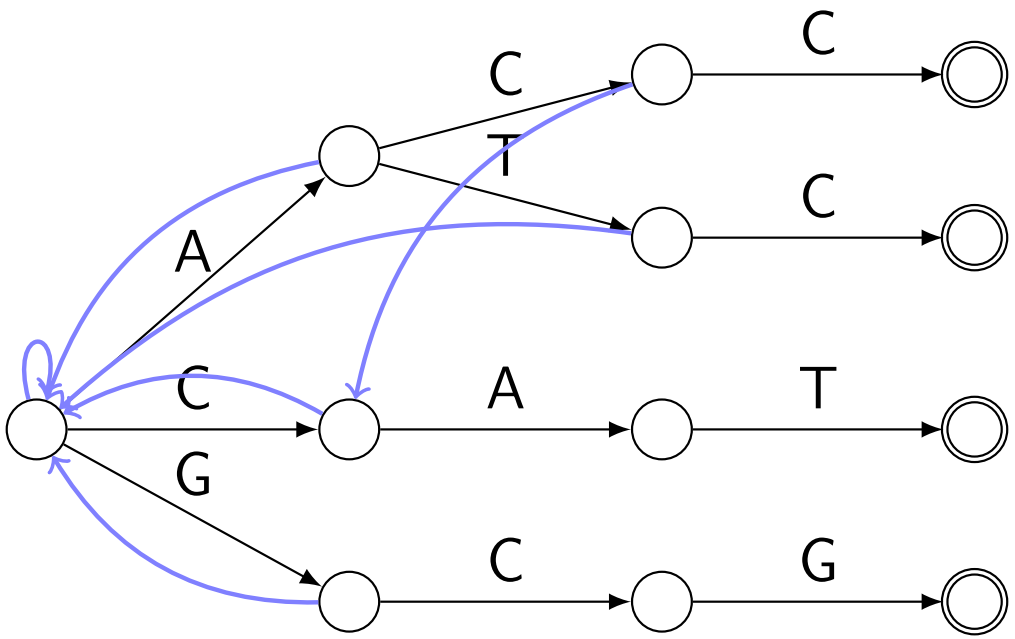

**Failure function**: returns the longest proper suffix accessible from the initial state

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$



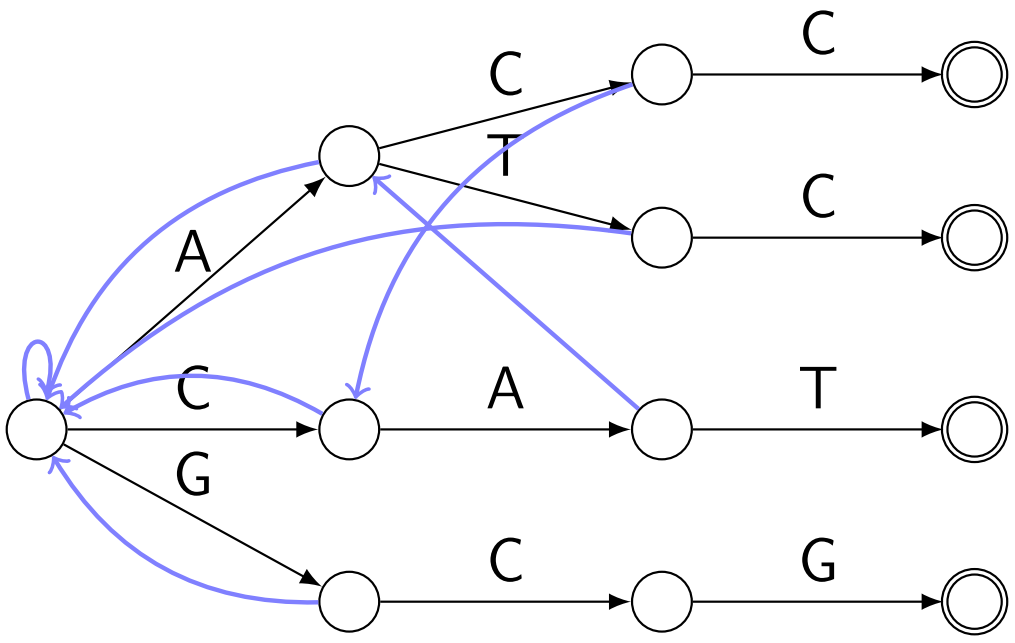

**Failure function**: returns the longest proper suffix accessible from the initial state

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$



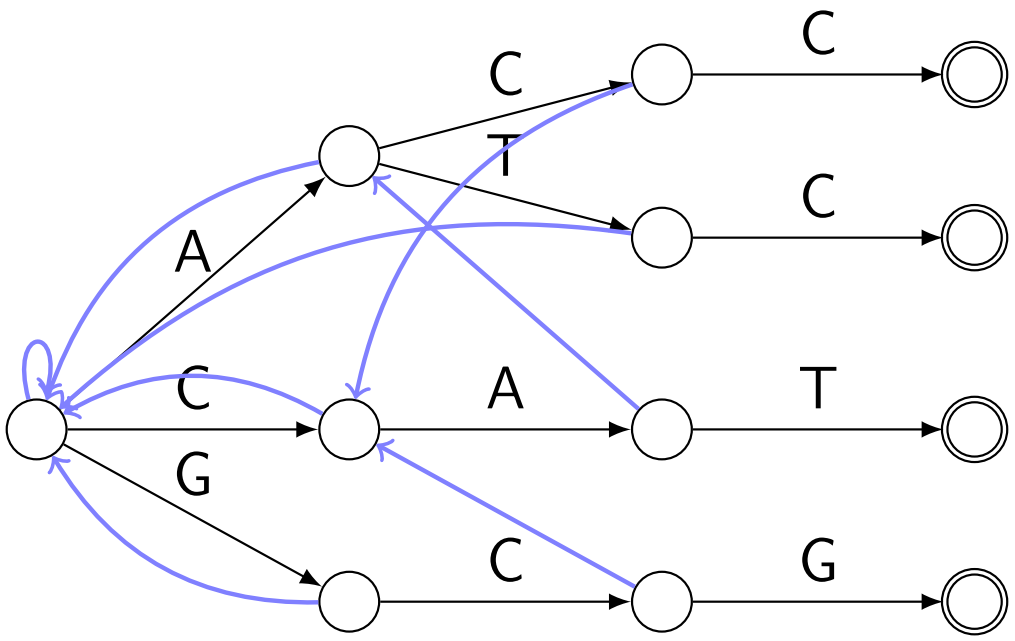

**Failure function**: returns the longest proper suffix accessible from the initial state

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$



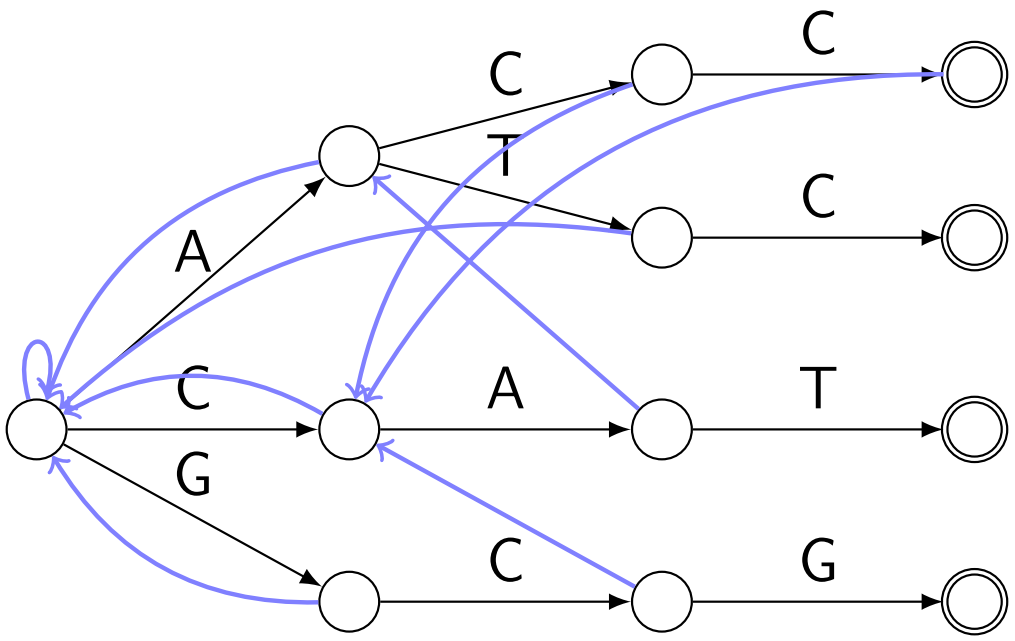

**Failure function**: returns the longest proper suffix accessible from the initial state

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$



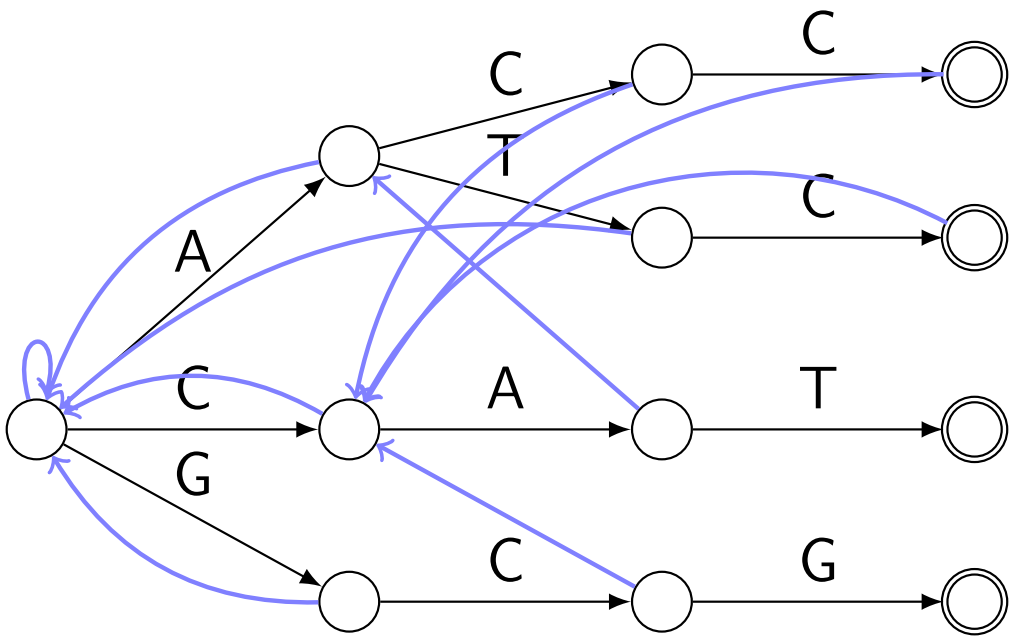

**Failure function**: returns the longest proper suffix accessible from the initial state

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$



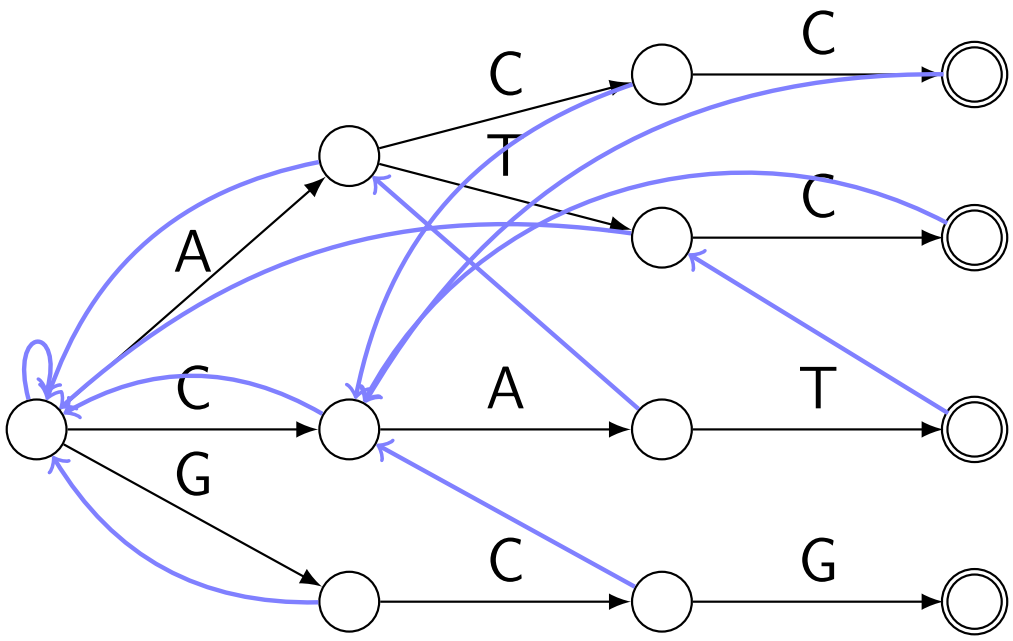

**Failure function**: returns the longest proper suffix accessible from the initial state

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$



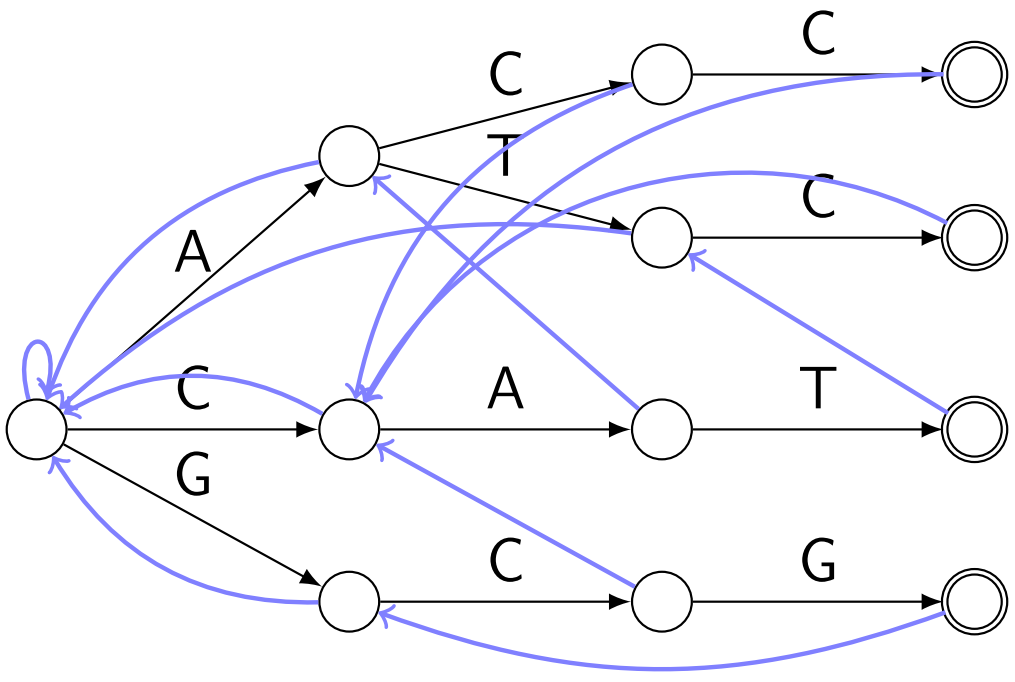

**Failure function**: returns the longest proper suffix accessible from the initial state

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$



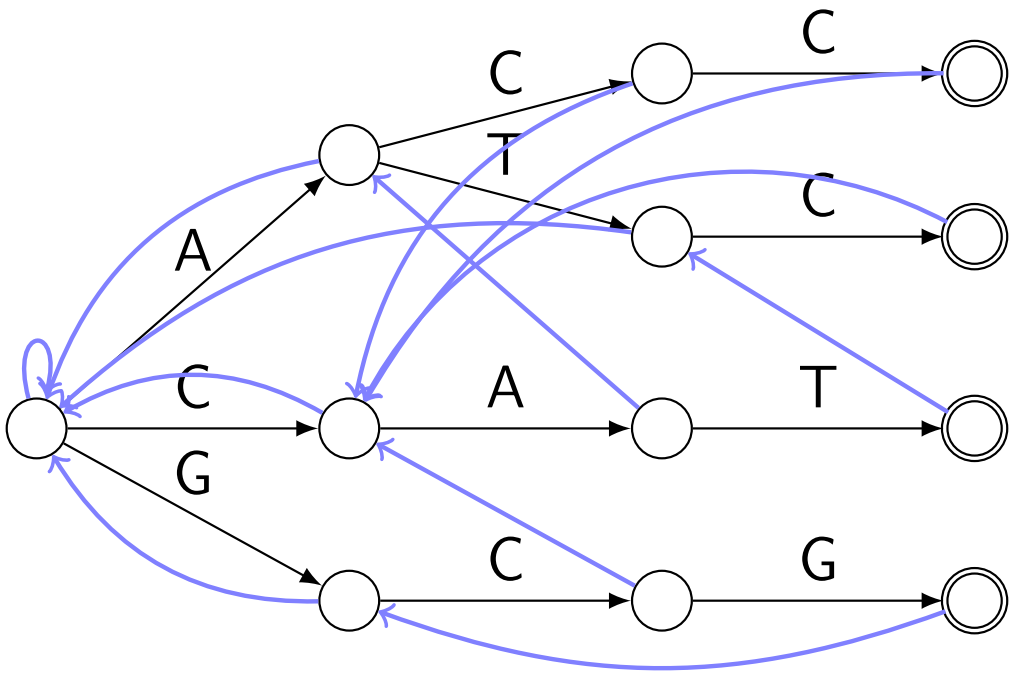

**Failure function**: returns the longest proper suffix accessible from the initial state

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

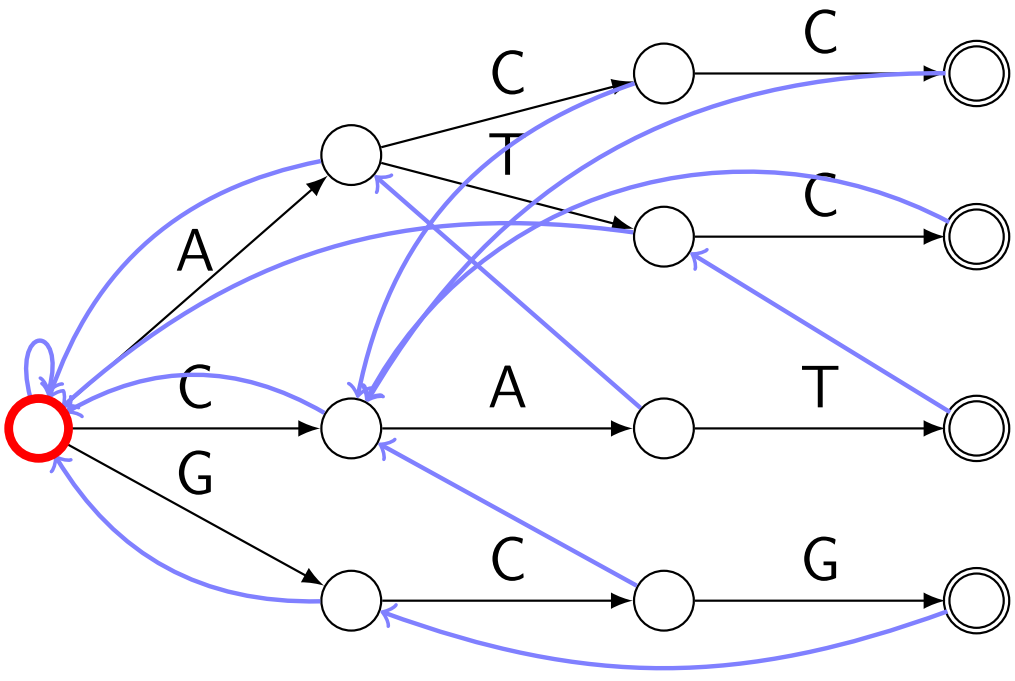
Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

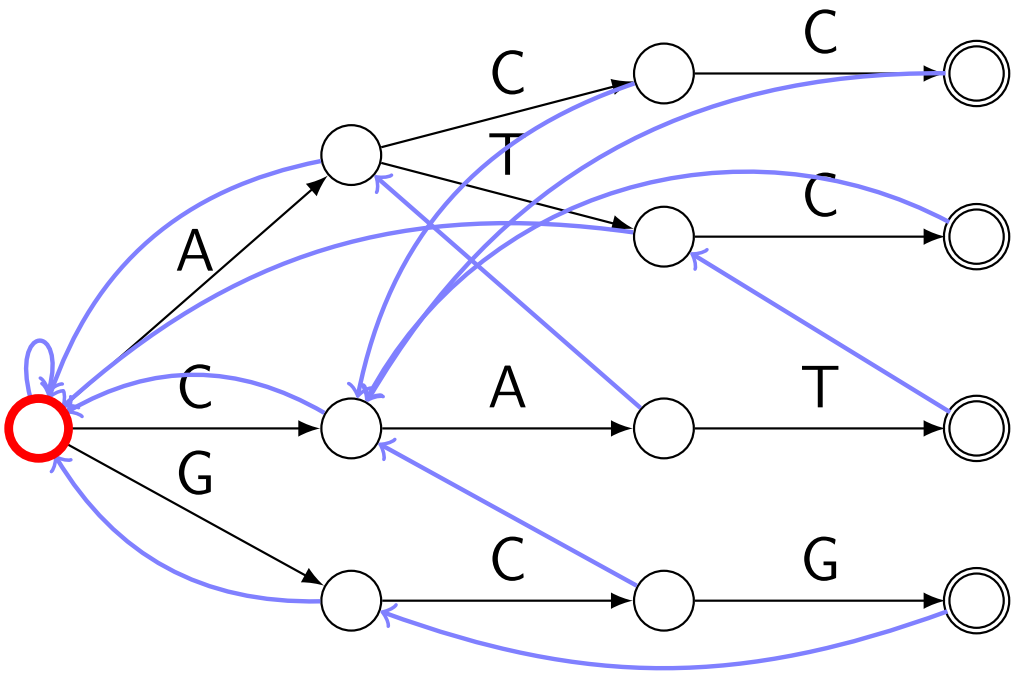
Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

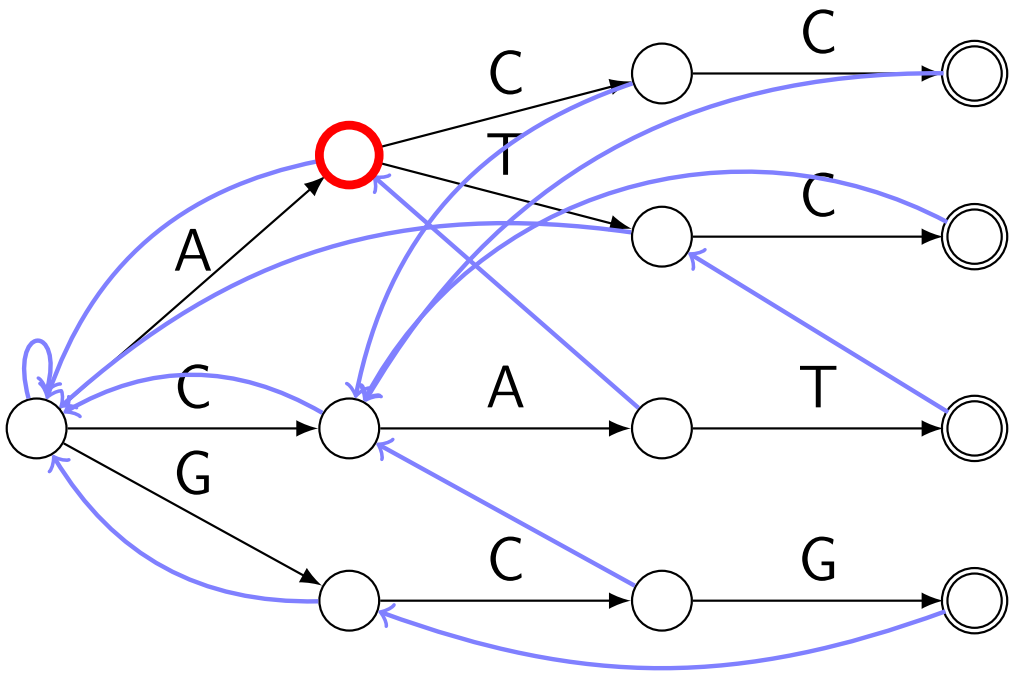
Searching $P$ in $T$ = ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

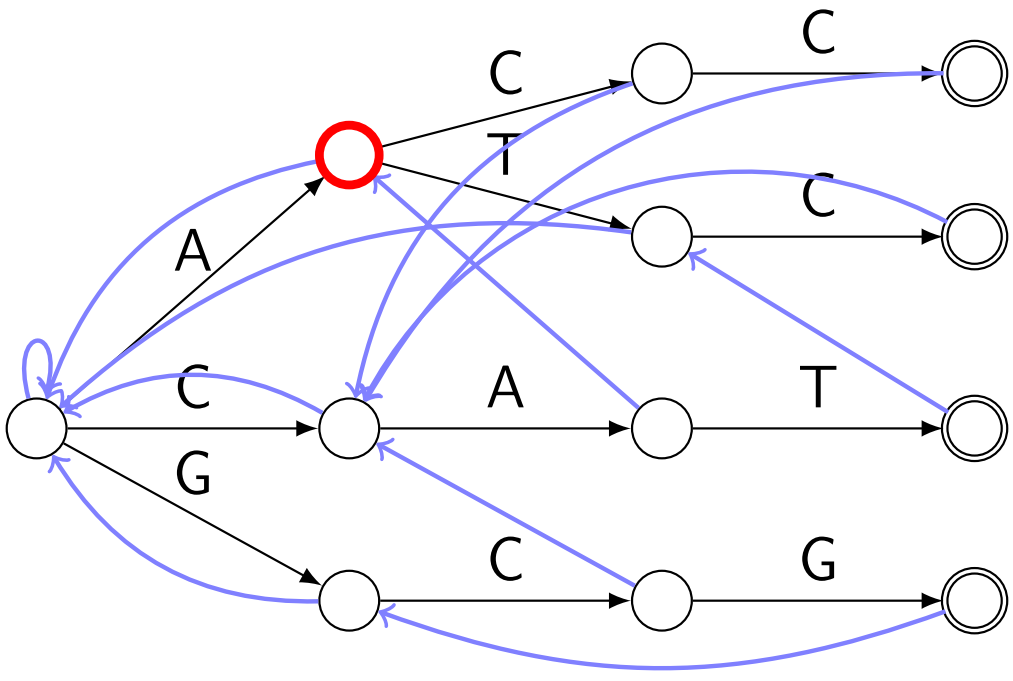
Searching $P$ in $T$ = ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

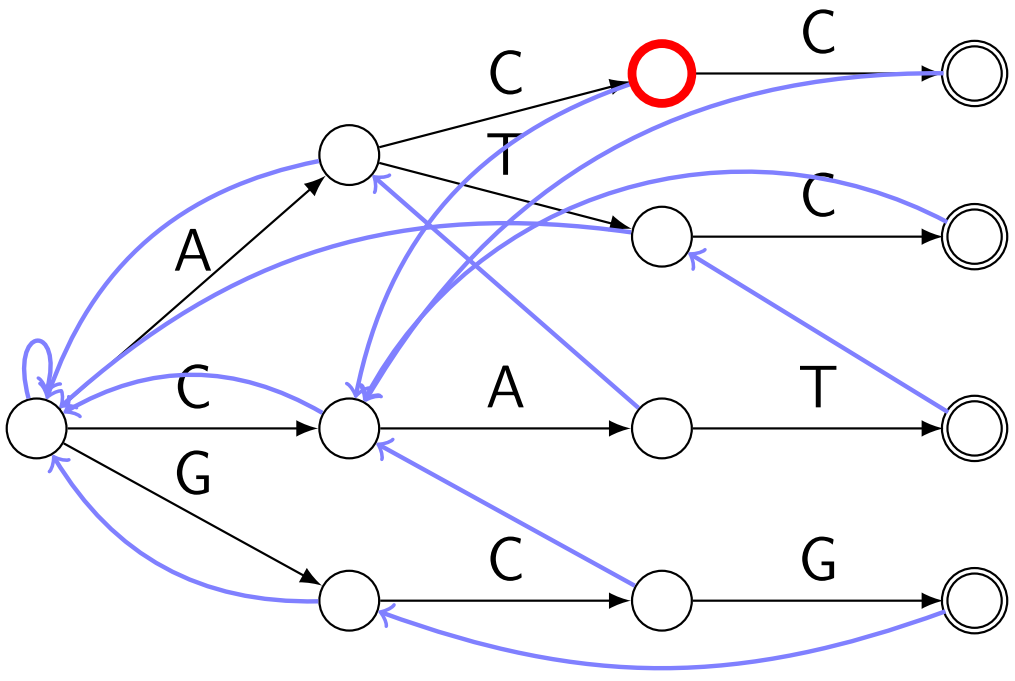
Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

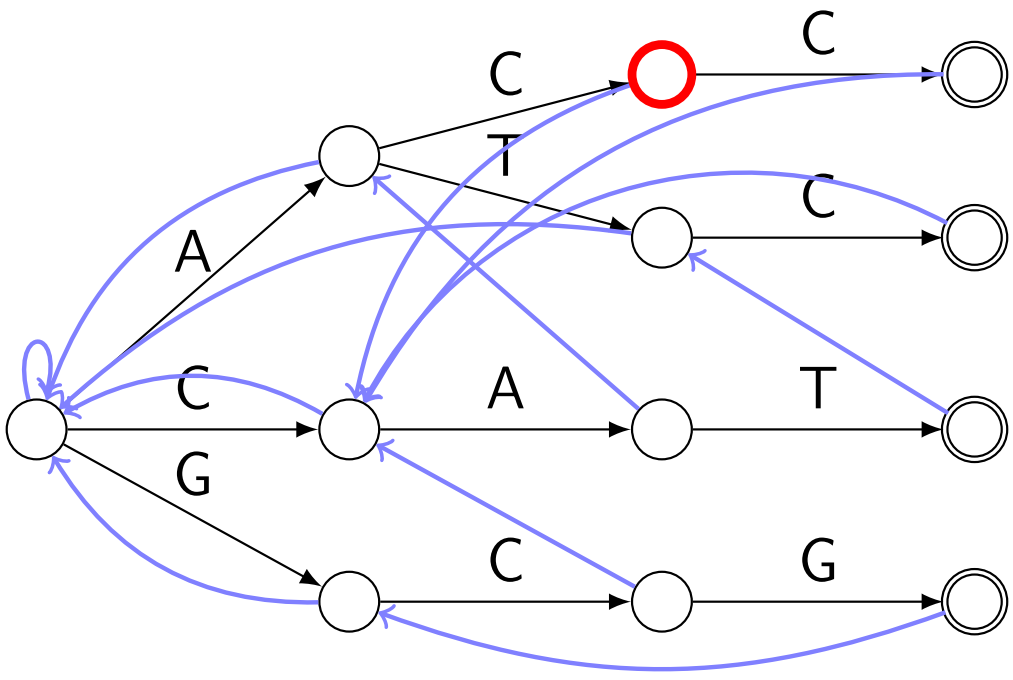
Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

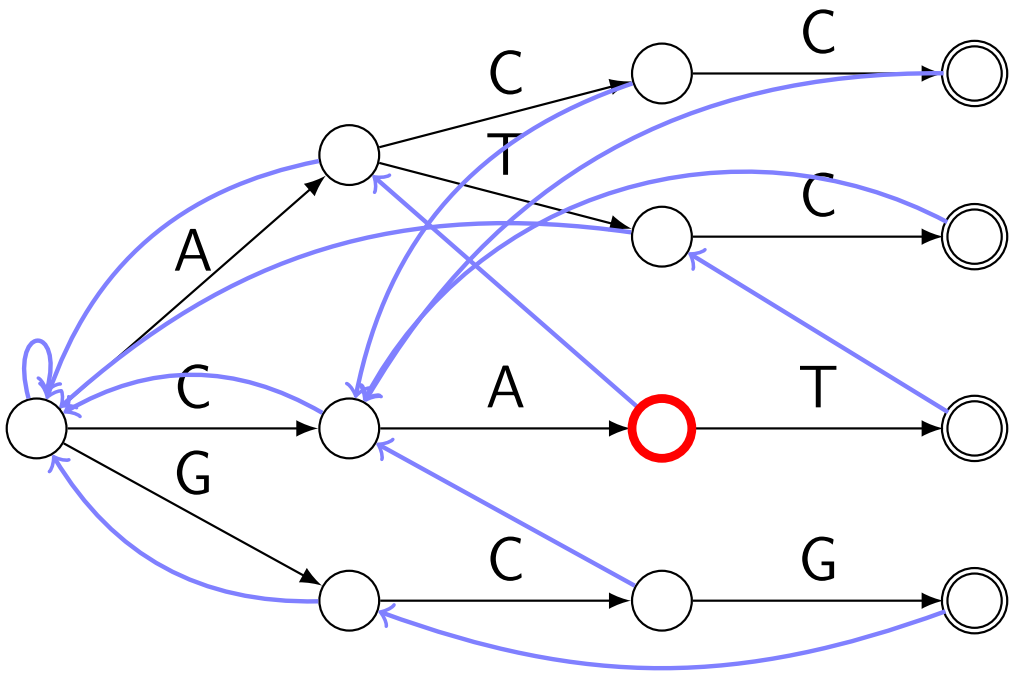
Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

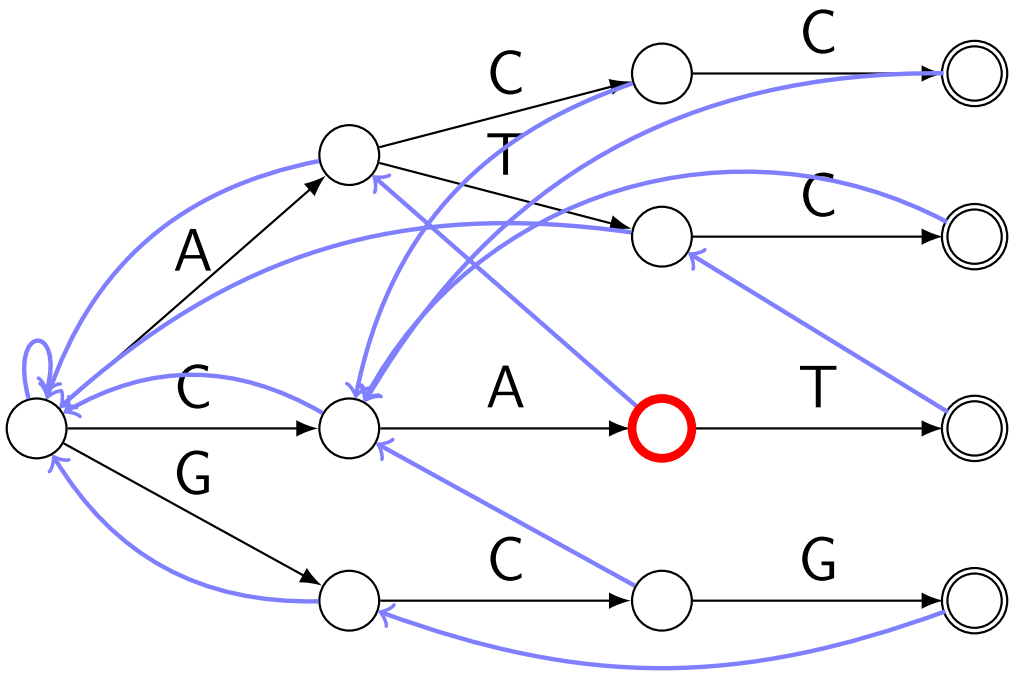
Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

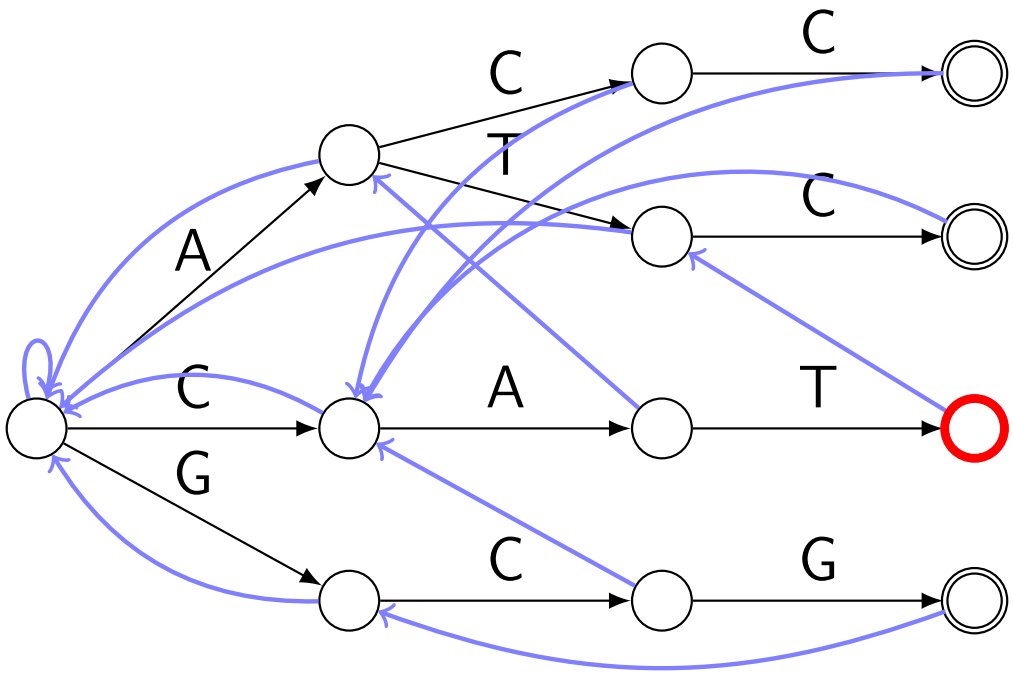
Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

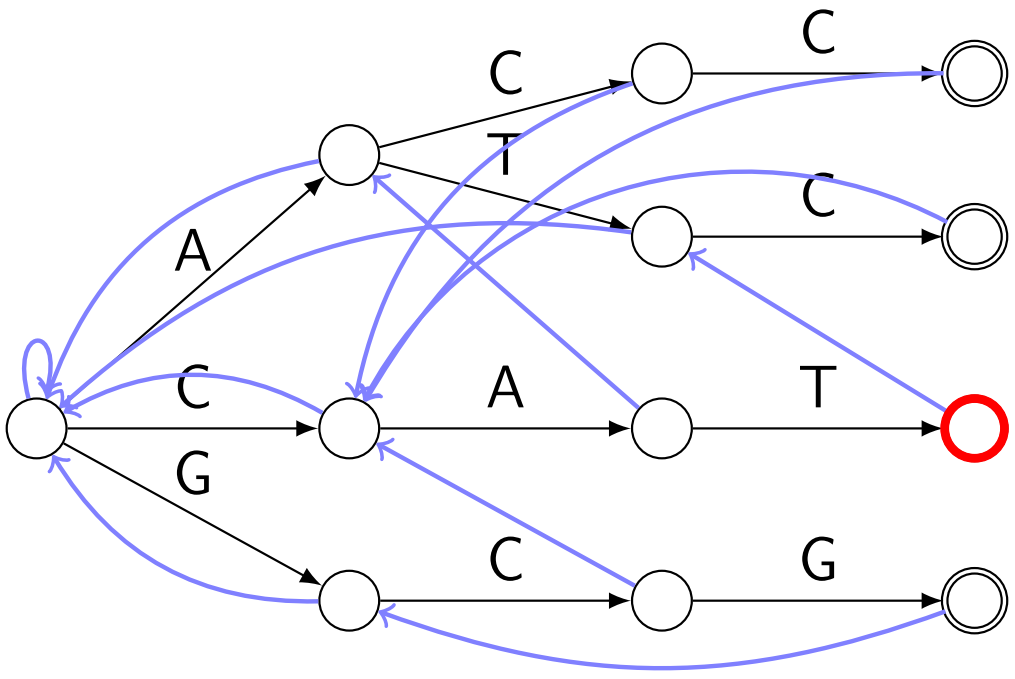
Searching $P$ in $T$ =ACATCG CAT found!

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

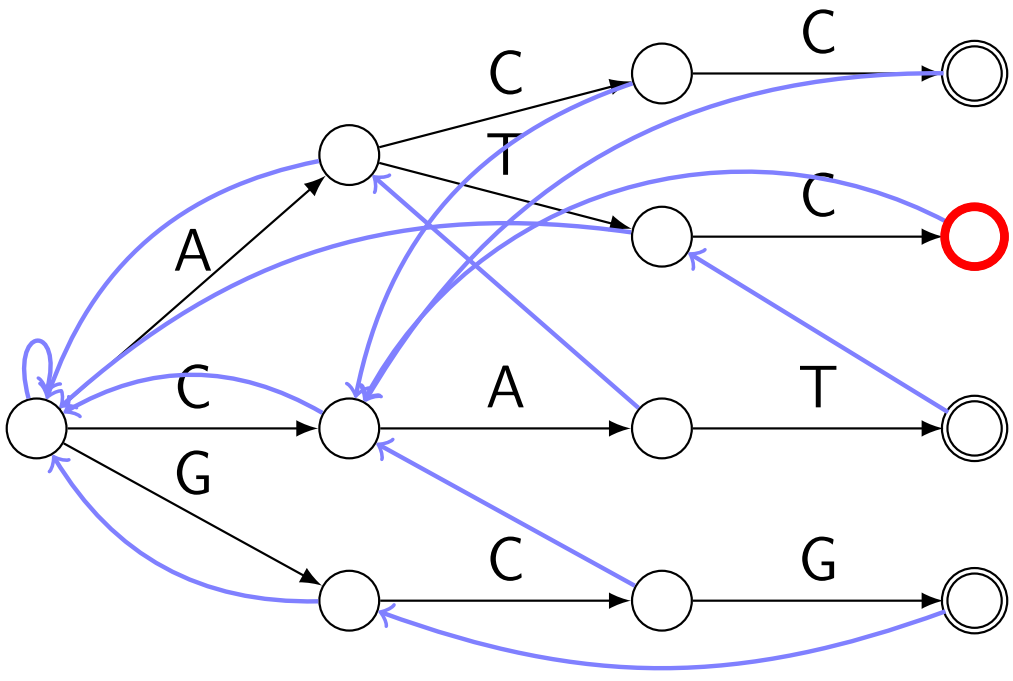
Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

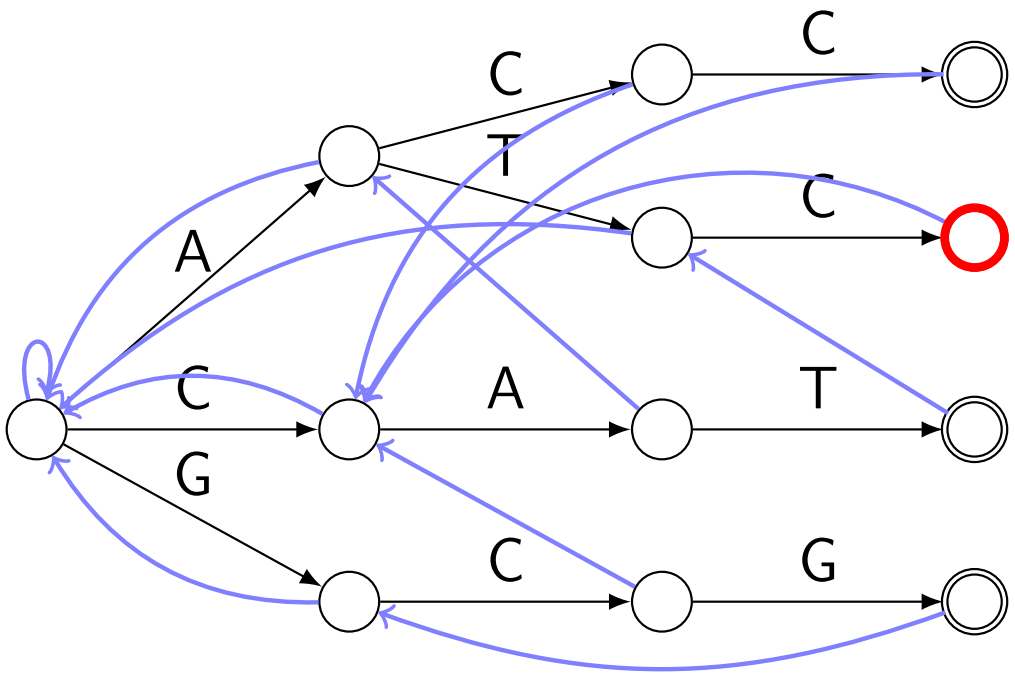
Searching $P$ in $T$ =ACATCG ATC found!

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

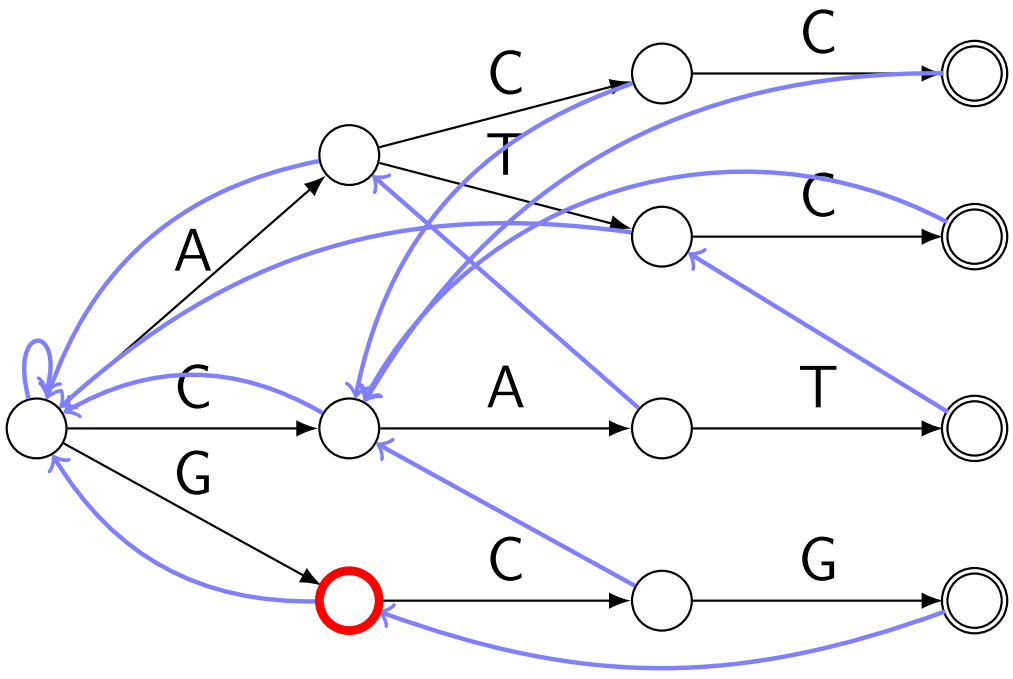
Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

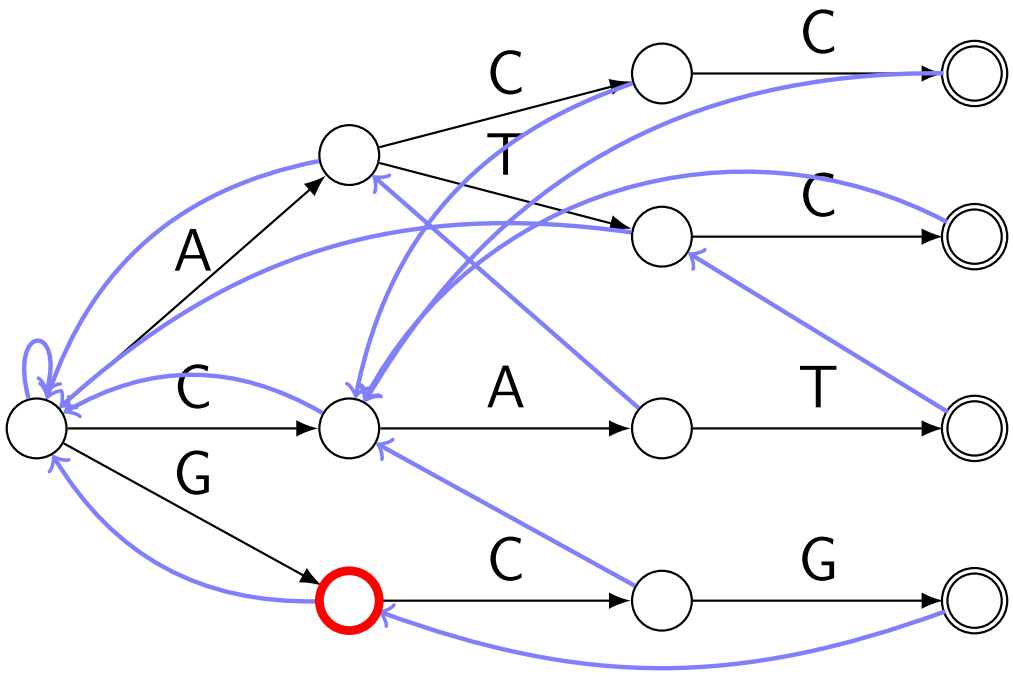
Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton: searches patterns in linear time

Introduced by Alfred Aho and Margaret Corasick in 1975

Searches a **set of patterns** $P$ in a text $T$ in time $O(|T|)$

$$P = \{\mathrm{ACC}, \mathrm{ATC}, \mathrm{CAT}, \mathrm{GCG}\}$$




**Failure function**: returns the longest proper suffix accessible from the initial state

Searching $P$ in $T$ =ACATCG

# Aho-Corasick automaton for V(D)J detection

# Aho-Corasick automaton for V(D)J detection

**What are the patterns?**

# Aho-Corasick automaton for V(D)J detection

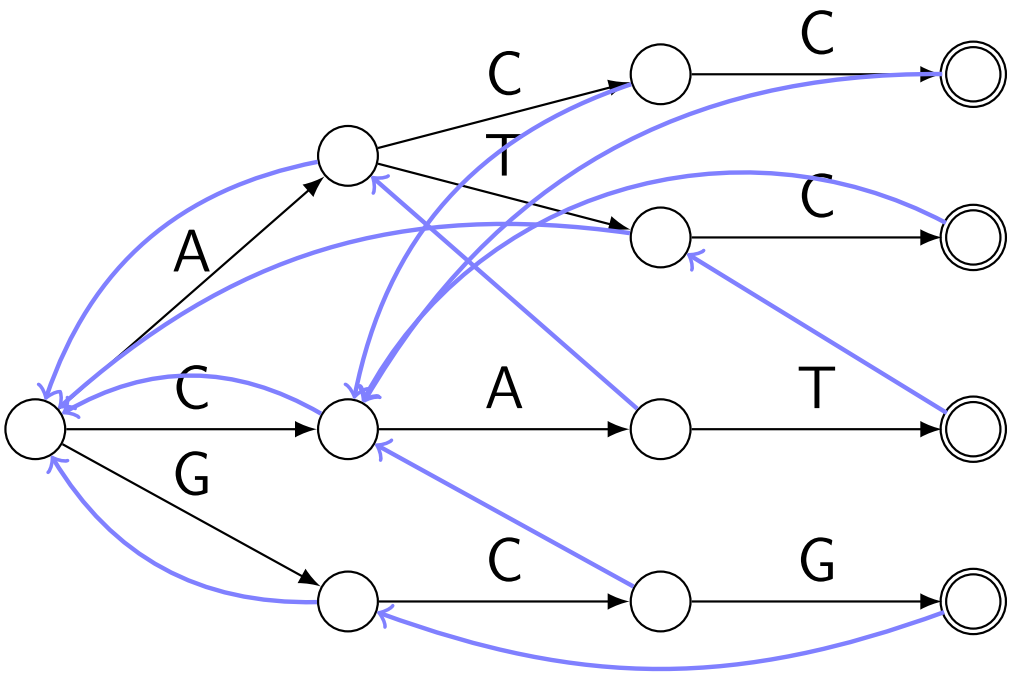
## What are the patterns?

(spaced) k-mers from V and J genes

# Aho-Corasick automaton for V(D)J detection

## What are the patterns?

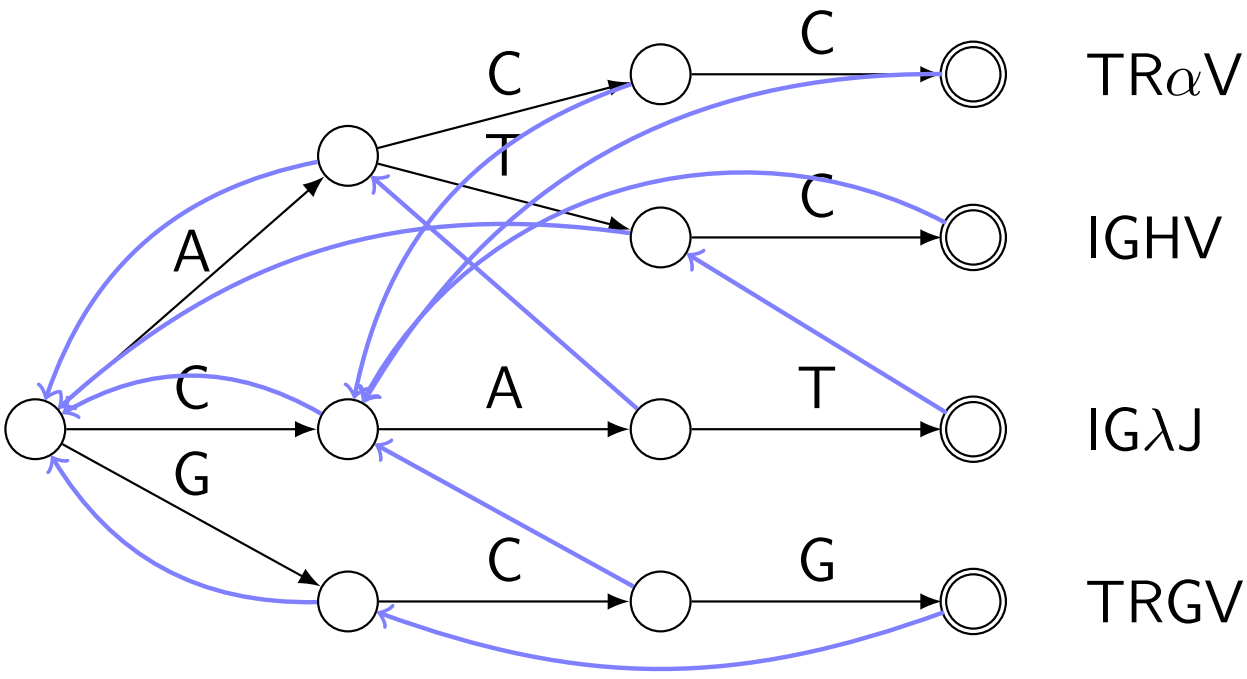
(spaced) k-mers from V and J genes



# Aho-Corasick automaton for V(D)J detection

## What are the patterns?

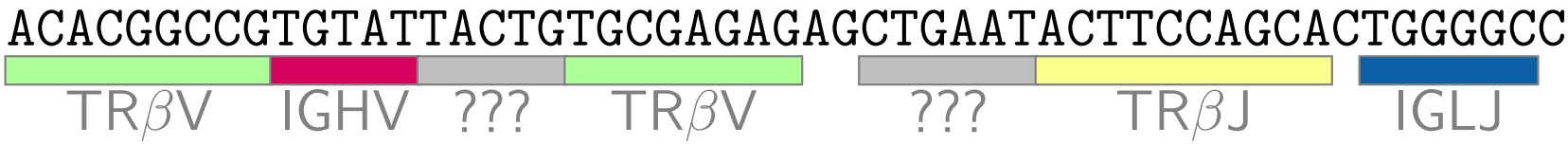
(spaced) k-mers from V and J genes



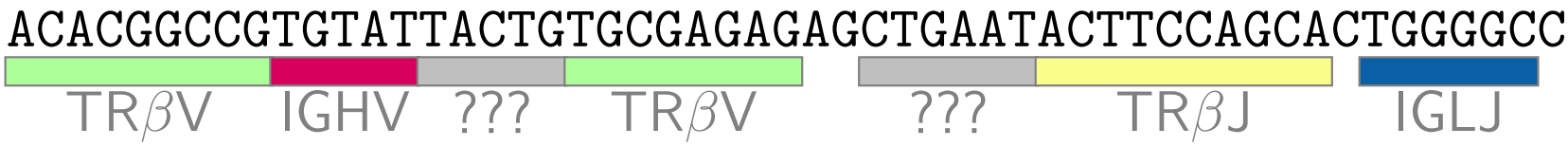
# Analysing all recombinations in a single pass

ACACGGCCGTGTATTACTGTGCGAGAGAGCTGAATACTTCCAGCACTGGGGCC

# Analysing all recombinations in a single pass



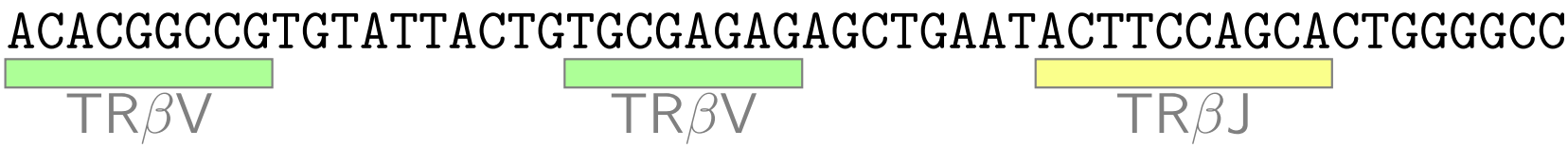
# Analysing all recombinations in a single pass




Keep the two most abundant annotations

Here TR$\beta$V and TR$\beta$J

# Analysing all recombinations in a single pass




Keep the two most abundant annotations

Here TR$\beta$V and TR$\beta$J

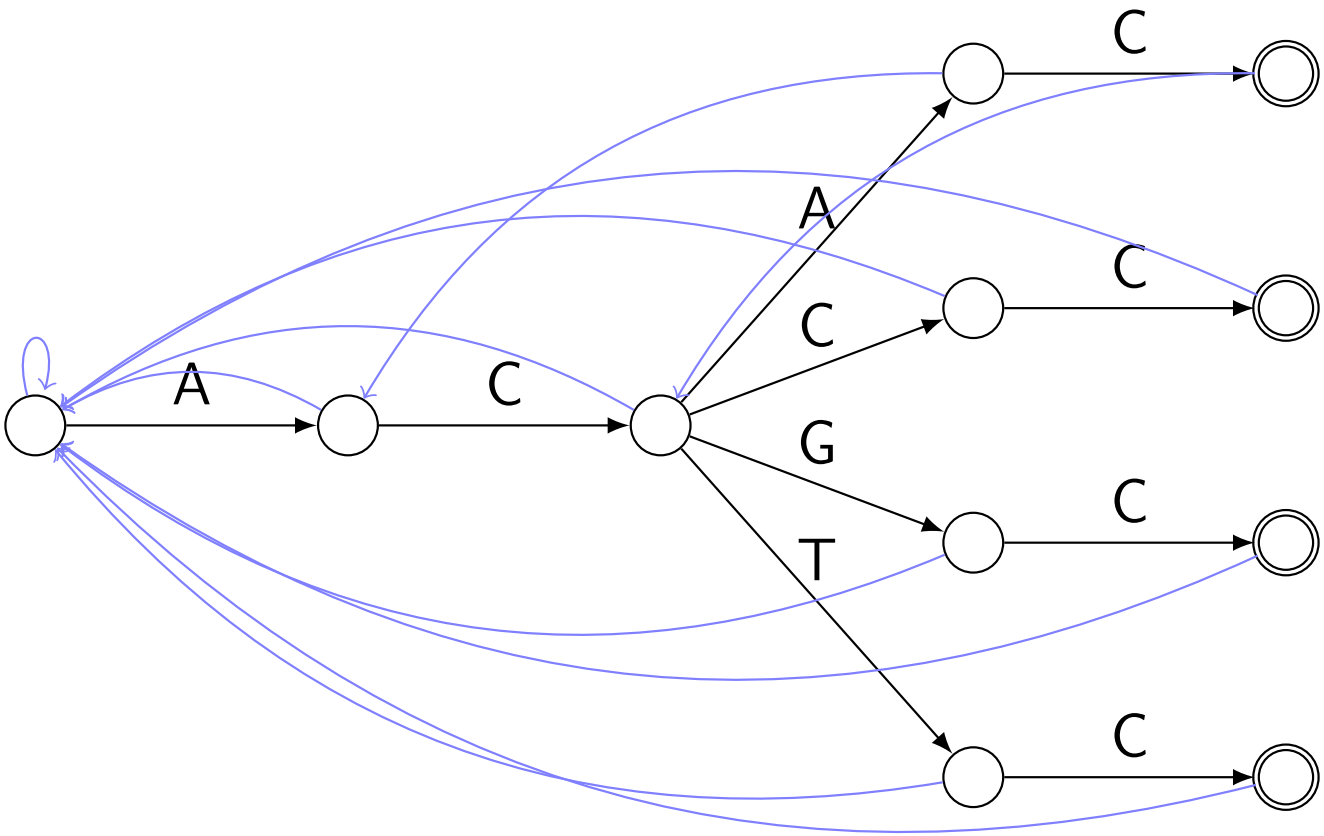
# How to include spaced seeds in the AC automaton?

# How to include spaced seeds in the AC automaton?

Not in a very smart way: add all possible paths

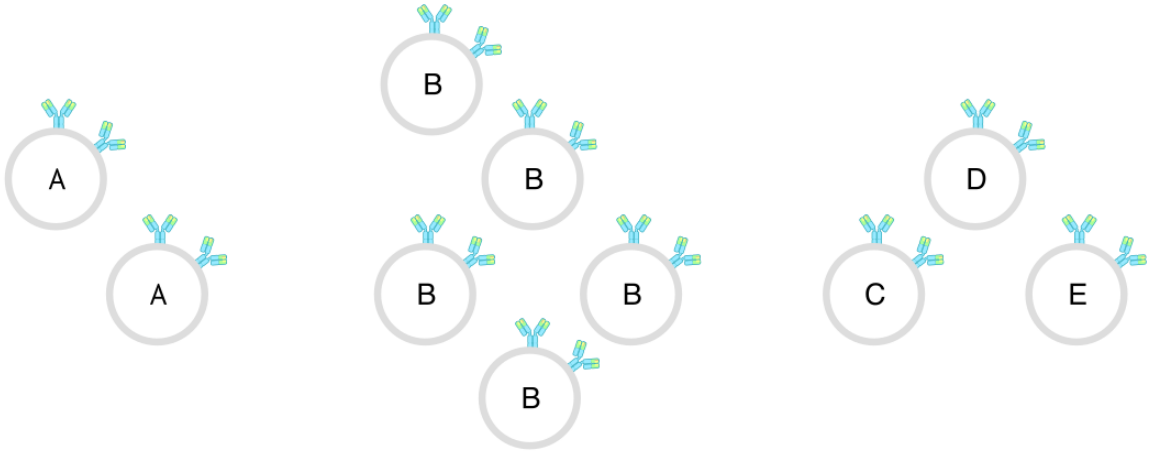
# How to include spaced seeds in the AC automaton?

Not in a very smart way: add all possible paths
Indexing AC–C

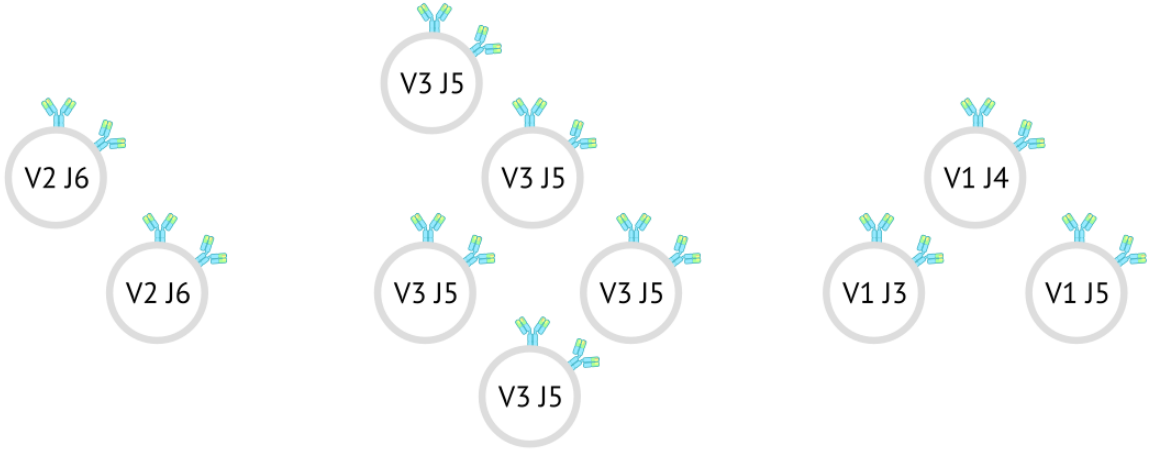


# V(D)J detection or V(D)J assignment?

## V(D)J detection




## V(D)J assignment



# Comparison with other software

| | |
|---|---|
| MiXCR | V(D)J-assign all reads (Bolotin *et al*, 2015) |
| IgReC | V(D)J-assign all reads (Shlemov *et al*, 2016) |
| Vidjil-algo (old) | V(D)J-detect all reads and assign most abundant clusters |
| Vidjil-algo (new) | V(D)J-detect all reads and assign most abundant clusters |

# Comparison with other software

MiXCR V(D)J-assign all reads (Bolotin *et al*, 2015)

IgReC V(D)J-assign all reads (Shlemov *et al*, 2016)

Vidjil-algo (old) V(D)J-detect all reads
and assign most abundant clusters

Vidjil-algo (new) V(D)J-detect all reads
and assign most abundant clusters

Thus the comparison is unfair

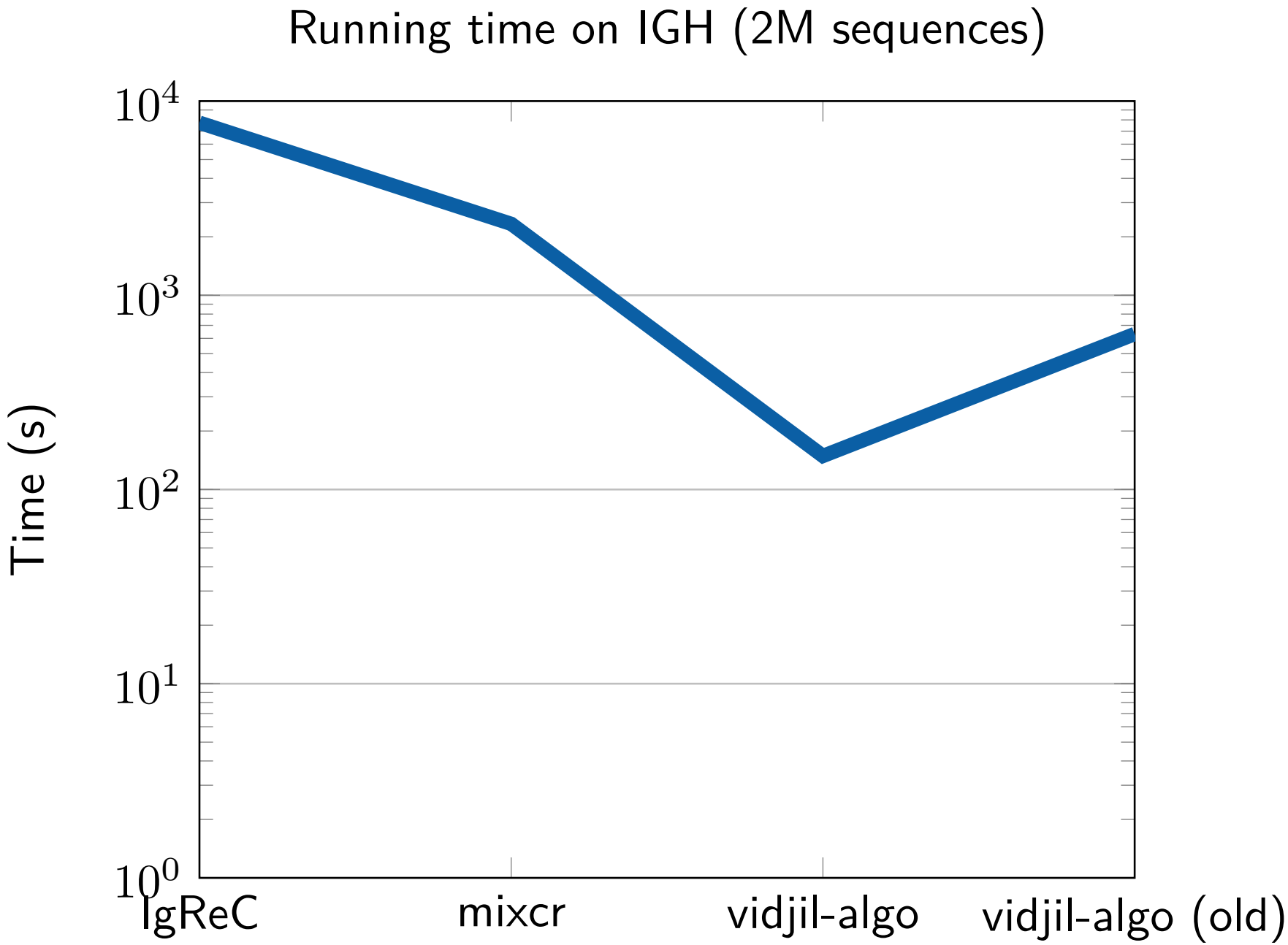
but that's the only one we can do

# Benchmark datasets

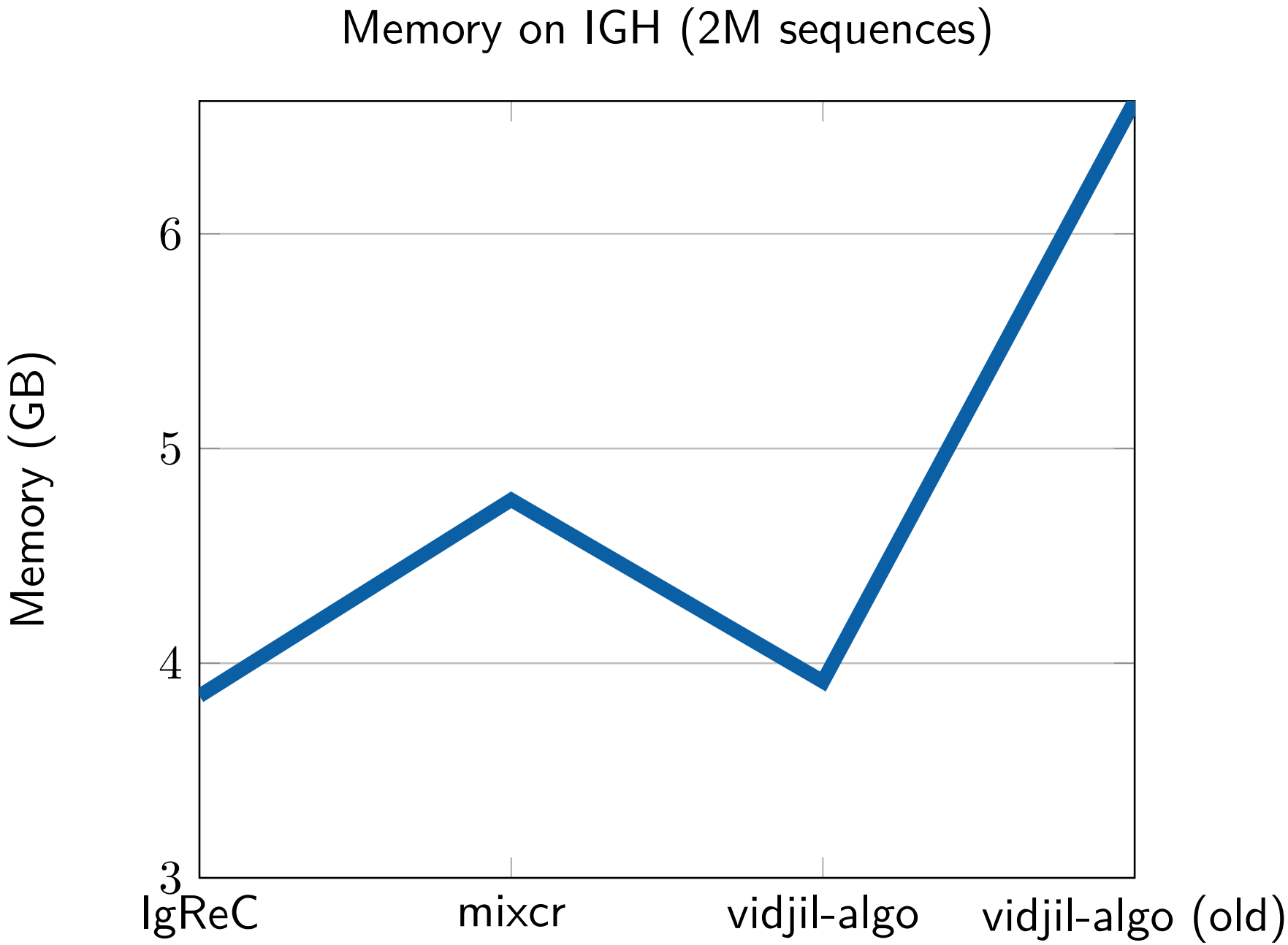
**True dataset** All V(D)J recombinations, with random indels at junctions and 2% differences

**False dataset** Random DNA sequences of length 350–450

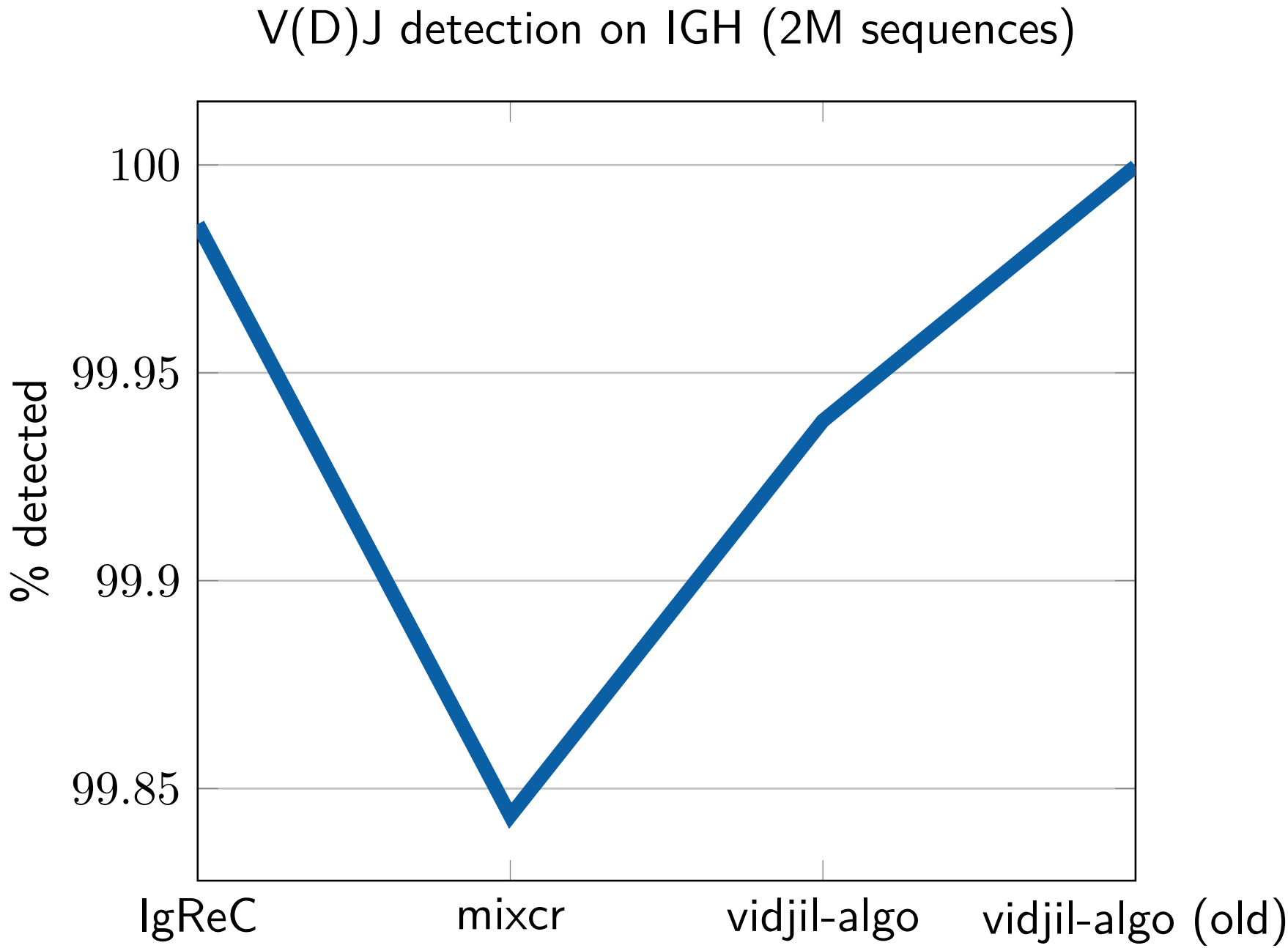
# A precise and quicker heuristic



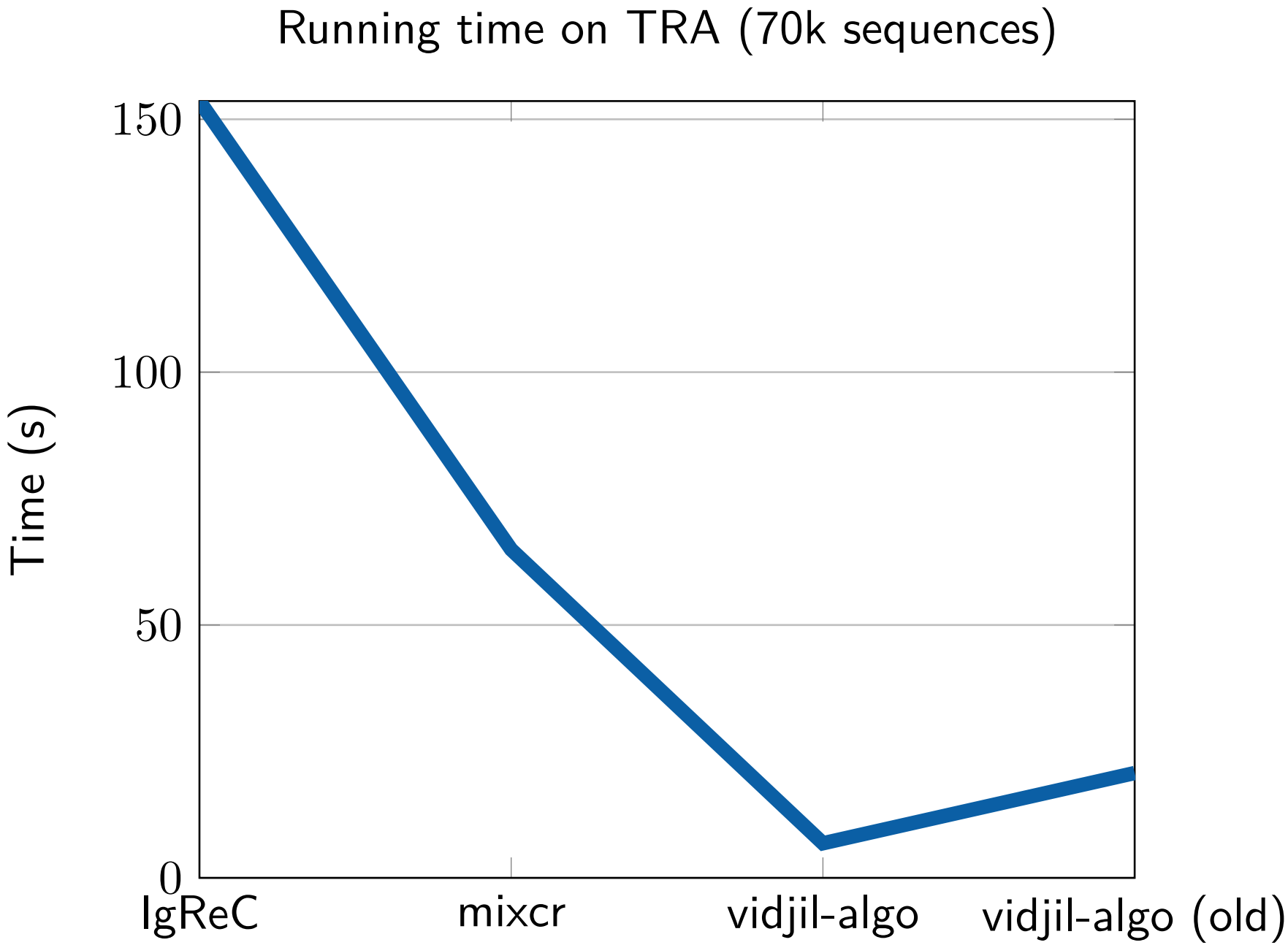
# A precise and quicker heuristic



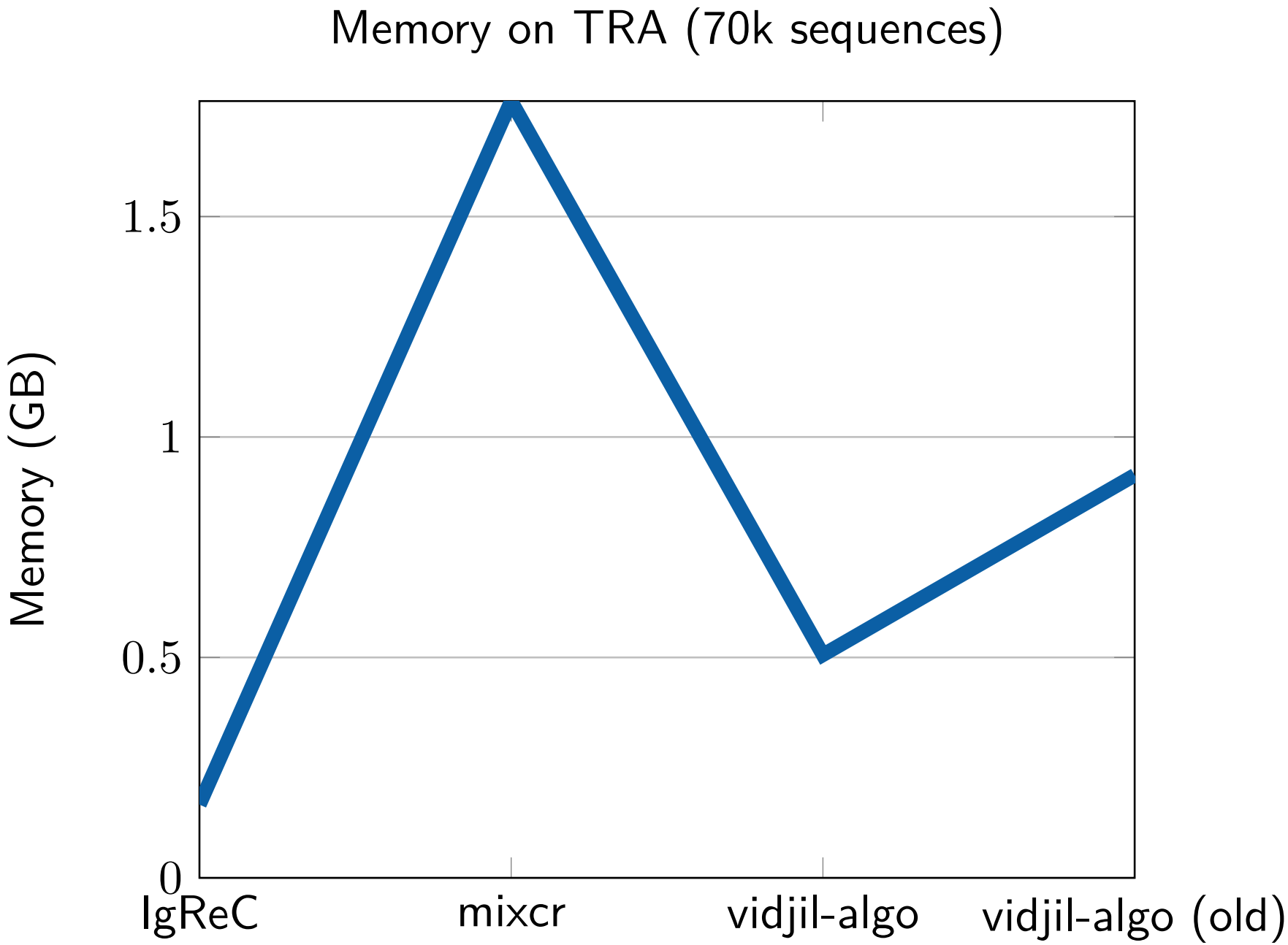
# A precise and quicker heuristic



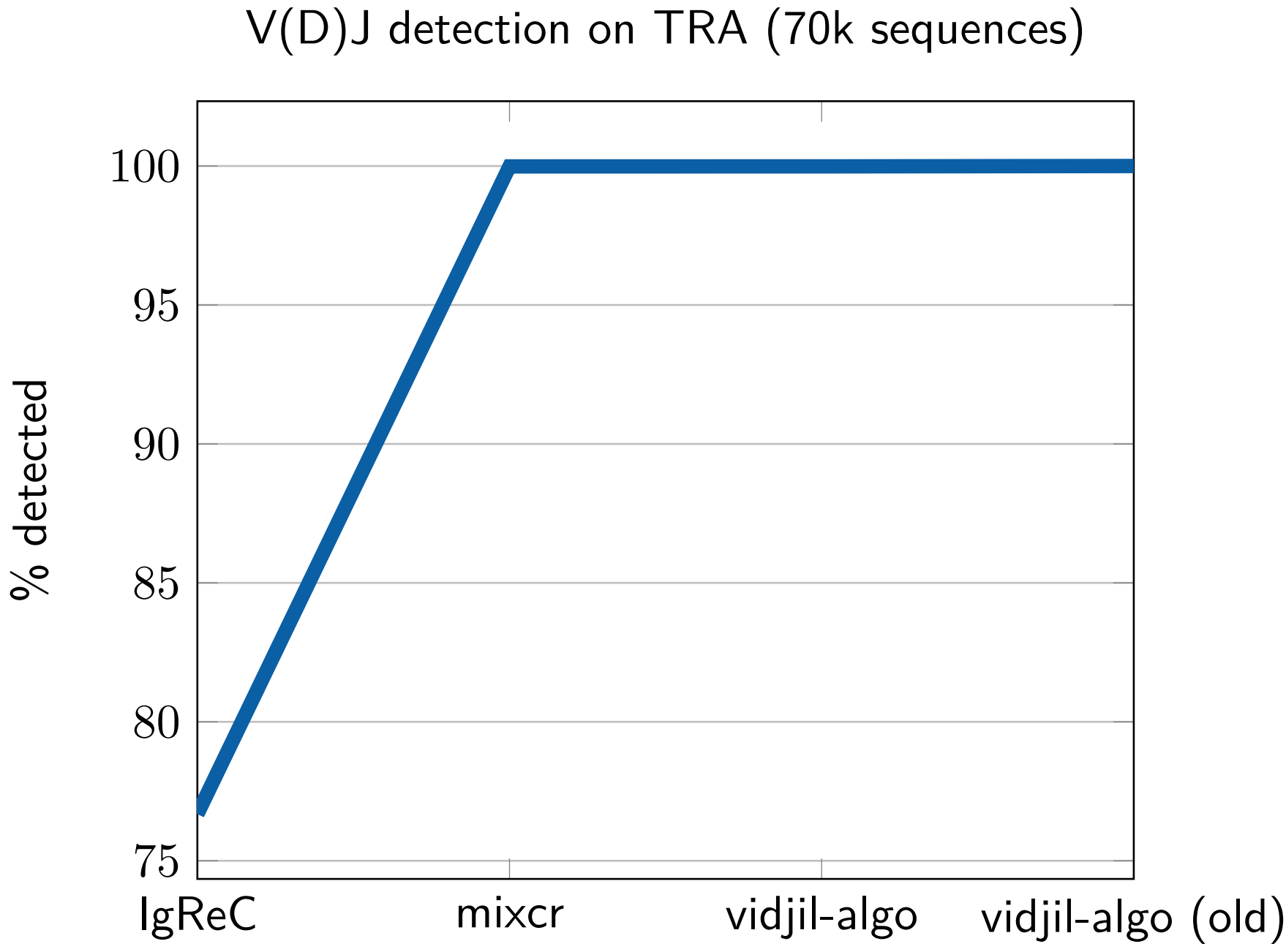
# A precise and quicker heuristic



# A precise and quicker heuristic



# A precise and quicker heuristic



# Conclusions

A linear-time alignment-free V(D)J detection

Much quicker, about as precise as before

In the future:
Consider several results per state
Optimize spaced seeds for each recombination system

Integrate to the Vidjil platform (50 samples/day)