Subset Seed Automaton

Automate des graines sous-ensemble

Gregory Kucherov\textsuperscript{1}, \textbf{Laurent Noé}\textsuperscript{1}, Mikhail Roytberg\textsuperscript{2}

\textsuperscript{1} LIFL, Université Lille 1 - INRIA (France)
\textsuperscript{2} Institute of Mathematical Problems in Biology, Pushchino (Moscow region)

\textit{Journées algorithmique, combinatoire du texte}
\textit{et applications en bio-informatique}

26, 27 et 28 septembre 2007 - Marne-la-Vallée
My talk
in a few words ...

Motivation : pairwise sequence alignment.
Seeds : filtration to speed-up sequence alignment.
Subset Seeds :
- a new model related to subset matching.
- its associated automaton ...
Sequence Alignment
on a very small DNA example

TTTTGAACGCGACAAAGTCATCGTGCAAATGCAGGAAAGAGAAAAACGCCGAAACGCTTCAGATTCAGCGCAAATGCTCAAGAGGCTCTCGTCGC
TGAGGCACGCTACGGGCCAGCCGAGCCAGTCAT
Sequence Alignment

on a very small DNA example

TTTTGAACGGGACGAAAGTGCATCAGTGCAAATGCGCAAGA

CGCCGAACGCTTCAGATCAGCGCAAATGCTCAAGAGGTCTCGTCGC

TGAGGCACTACGGCCAGCCAGCCAGTCAT
Sequence Alignment
on a very small DNA example

TTTTGAACTGGGACGAAAGTGCATCAGTGCAAATGCGCAAGA
AAAA
CGCCGAACGCTTCAGATCAGCGCAAATGCTCAAGA
GGTCTCGTCGC
TGAGGCACTACGGCCAGCCGAGCCAGTCAT
Sequence Alignment
on a very small DNA example

TTTTGAACTGGGACGAAAGTGCATCAATGCGCAAGAAGAAGA
CGCCGAACGCTTTCAGATCAGCGCAGAAATGCTCAAGAGGTCTCGTCGC
TGAGGCACTACGGCCAGCCGAGCCAGTCAT

ATCAATGCGCAAGA
ATCAGCTCAAGA
Sequence Alignment

on a very small DNA example

TTTTGAACTGGGACGAAAGTGCATCAGTGCAAAATGCGCAAGAAAAA
CGCCGAACGCTTCAGATCAGCGCAAATGCTCAAGAGGTCTCGTCGC
TGAGGCACGTACGGCCAGCCGAGCCAGTCAT

ATCAGTGCAAAATGCGCAAGA

TTTTGAACTGGGACGAAAGTGCATCAGTGCAAAATGCGCAAGA

CGCCGAACGCTTCAGATCAGCGCAAATGCTCAAGAGGTCTCGTCGC

TGAGGCACGTACGGCCAGCCGAGCCAGTCAT

ATCAGTGCAAAATGCGCAAGA

ATCAGCGCAAATGCTCAAGA

ATCAGTGCAAAATGCGCAAGA

ATCAGCGCAAATGCTCAAGA

ATCAGTGCAAAATGCGCAAGA

ATCAGCGCAAATGCTCAAGA
Algorithm: Smith-Waterman algorithm (in $O(n^2)$).

Heuristic: Filtration principle

1. some clues are detected using seeds.
2. these clues are extended by local dynamic programming.
Contiguous Seeds
(Fasta 85, Blast 91, Gapped-Blast 97, ...)

**Principle:** A contiguous seed \( \pi \) detects one alignment motif of size \( k \).

**Notation:** \( \pi \) is represented by a (fixed length) word over alphabet \( \{\#\} \).

(# only accepts the | symbol from an alignment).

**Example**

seed pattern : \( \pi = \#\#\#\#\#\#\# \)

```
ATCAGTGC
CAATGCG
CAAGA
```

```
ATCAGCGCAAATGCTCAAGA
```

```
ATCAGCGCAAATGCTCAAGA
```

### Gregory Kucherov¹, Laurent Noé¹, Mikhail Roytberg²

**Subset Seed Automaton**
Contiguous Seeds

(Fasta 85, Blast 91, Gapped-Blast 97, …)

Principle: A contiguous seed $\pi$ detects one alignment motif of size $k$.

Notation: $\pi$ is represented by a (fixed length) word over alphabet $\{\#\}$.

($\#$ only accepts the $|$ symbol from an alignment).

Example

seed pattern: $\pi = #######$

```
####
ATCAGTGCAAATGCGAAGA
| | | | | | | | | | | | | | | | | | | |
ATCAGCGCAAATGCTCAAGA
```
Contiguous Seeds
(Fasta 85, Blast 91, Gapped-Blast 97, ...)

Principle: A contiguous seed $\pi$ detects one alignment motif of size $k$.
Notation: $\pi$ is represented by a (fixed length) word over alphabet $\{\#\}$. 
(\# only accepts the $|$ symbol from an alignment).

Example

seed pattern: $\pi = \#\#\#\#\#\#$

```
#\#\#\#\#\#
ATCAGTGC\#\#\#\#\#\#GCAAGA
\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\:\\\...
**Contiguous Seeds**

*(Fasta 85, Blast 91, Gapped-Blast 97, ...)*

**Principle:** A contiguous seed $\pi$ detects one alignment motif of size $k$.

**Notation:** $\pi$ is represented by a (fixed length) word over alphabet \{#\}. (# only accepts the | symbol from an alignment).

---

**Example**

<table>
<thead>
<tr>
<th>Seed Pattern $\pi$ $=$</th>
<th>######</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATCAG TGCAATGC GCAAGA</td>
<td>######</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ATCAG CGCAATGCT CAAGA</td>
<td></td>
</tr>
</tbody>
</table>
Contiguous Seeds

(\textit{Fasta 85, Blast 91, Gapped-Blast 97, ...})

**Principle:** A contiguous seed $\pi$ detects one alignment motif of size $k$.

**Notation:** $\pi$ is represented by a (fixed length) word over alphabet \{#\}. ($#$ only accepts the $|$ symbol from an alignment).

**Example**

seed pattern: $\pi = \#\#\#\#\#$

```
ATCAGTGCAAATGCGCAAGA
| | | | : | | | | | | | | | | |
ATCAGCGCAAATGCTCAAGA
```
Contiguous Seeds

\textit{(Fasta 85, Blast 91, Gapped-Blast 97, \ldots)}

\textbf{Principle:} A contiguous seed $\pi$ detects one alignment motif of size $k$.

\textbf{Notation:} $\pi$ is represented by a (fixed length) word over alphabet $\{\#\}$.

($\#$ only accepts the $|$ symbol from an alignment).

\textbf{Example}

seed pattern : $\pi = \#\ldots\ldots\ldots\ldots\#$

\begin{verbatim}
   #\ldots\ldots\ldots\ldots\ldots#

   ATCAGTGCAAATGCGCAAGA
   | | | | : | | | | | | | . | | | | |
   ATCAGCGAAATGCTCAAGA
\end{verbatim}
**Definition**

A spaced seed $\pi$ is defined as a binary word over the alphabet \{#, −\} with:
- # : accepts only match symbol |
- − : accepts all alignment symbols (joker).

$s : \text{span (length)}, w : \text{weight (number of #)}$.

**Example**

seed pattern : $\pi = ###-#-##$

ATCAGTGCGAATGCGCAAGA
ATCAGCGCAATGCTCAAGA
Spaced Seeds
(PatternHunter 02, Burkhardt et al. 01, BLASTz 03, YASS 04)

Definition

A spaced seed $\pi$ is defined as a binary word over the alphabet $\{\#, -\}$* with:

- $\#$ : accepts only match symbol $\|$,
- $- \,$ : accepts all alignment symbols (joker).

$s : \text{span (length)}, w : \text{weight (number of $\#$).}

Example

seed pattern : $\pi = \#\#\#-\#-\#\#$

###-#-##
ATCAGTGCAGGATGCGCAAGA
ATCAGCGCAATGCTCAAGA
Definition

A spaced seed $\pi$ is defined as a binary word over the alphabet $\{\#, -\}^\star$ with:
- $\#$: accepts only match symbol $\mid$,
- $-$: accepts all alignment symbols (joker).

$s$ : span (length), $w$ : weight (number of $\#$).

Example

seed pattern : $\pi = \text{###--#--##}$

$\text{###--#--##}$

ATCAGTGCAGATGCGCAAGA

| | | | : | : | | | | | | |

ATCAGCGCAATGCTCAAGA
Definition

A spaced seed $\pi$ is defined as a binary word over the alphabet $\{#, -\}^*$ with:

- $\#$ : accepts only match symbol $\|$,
- $-$ : accepts all alignment symbols (joker).

$s$ : span (length), $w$ : weight (number of $\#$).

Example

seed pattern : $\pi = \#\#\#-\#-\#\#$

\[
\begin{array}{c}
\#\#\#-\#-\#\# \\
ATCAGTGCGAATGCGCAAGA \\
| | | | : | | : | | | | . | | | | \\
ATCAGCGCAATGCTCAAGA
\end{array}
\]
Spaced Seeds
(PatternHunter 02, Burkhardt et al. 01, BLASTz 03, YASS 04)

Definition

A spaced seed $\pi$ is defined as a binary word over the alphabet $\{#, -\}^*$ with:

- $#$: accepts only match symbol $\mid$,
- $-$: accepts all alignment symbols (joker).

$s$: span (length), $w$: weight (number of $#$).

Example

seed pattern: $\pi = \#\#\#-\#-\#\#$

$\#\#\#-\#-\#\#$
ATCAGTGCAGATGCGCAAGA
$\mid\mid\mid\mid:\mid:\mid\mid\mid\mid\mid\mid\mid\mid$'
ATCAGCGCAATGCTCAAGA
Spaced Seeds

(PatternHunter 02, Burkhardt et al. 01, BLASTz 03, YASS 04)

Definition

A spaced seed $\pi$ is defined as a binary word over the alphabet $\{#, -\}^*$ with:
- # : accepts only match symbol $|$ ,
- - : accepts all alignment symbols (joker).

$s : \text{span} \ (\text{length})$, $w : \text{weight} \ (\text{number of #})$.

Example

seed pattern : $\pi = ####-#-##$

$####-#-##$

ATCAGTGCGAATGCACAAGA
ATCAGCGCAAATGCTCAAGA
Spaced Seeds
(PatternHunter 02, Burkhardt et al. 01, BLASTz 03, YASS 04)

**Definition**

A spaced seed $\pi$ is defined as a binary word over the alphabet $\{\#, -\}$* with:

- $\#$: accepts only match symbol $\|$,
- $-$: accepts all alignment symbols (joker).

$s$: span (length), $w$: weight (number of $\#$).

**Example**

seed pattern: $\pi = $$$-##-$$$

|$|$$|$$| $$|$$|$$|$$ |
$A$$T$$C$$A$$G$$T$$G$$C$$G$$A$$A$$T$$G$$C$$G$$C$$A$$A$$G$$A$
Spaced Seeds
Research threads

- Burkhardt, Karkkainen, CPM 2001: *spaced seeds for (lossless) approximate pattern matching*
- Ma, Tromp, Li 2002 (*PatternHunter*): *spaced seeds for (lossy) similarity search*
- Califano, Rigoutsos 1993 (*FLASH*), Buhler 2001 (*LSH*)
Spaced Seeds
Research threads (cont.)

- **Extended seed models:** BLASTZ 2003, Brejova et al 2003, Chen&Sung 2003, Noe&Kucherov 2004 (YASS), Sun&Buhler 2006, Mak et al 2006
- **Statistical foundations:** Choi&Zhang 2004, Zhang 2005, Kong 2007
- **Efficient implementation of spaced seeds:** Csuros 2004, Csuros&Ma 2004
- **Multiple spaced seeds:** Li et al 2004 (PatternHunter II), Sun&Buhler 2004, Kong 2007
- **Designing (multiple) seeds:** Xu et al 2004, Brown 2004
- **Surveys:** Brown&Li&Ma 2005, Brown 2007
The main question in (most of) these papers: how to choose the best seed ...
The main question in (most of) these papers: how to choose the best seed ...

**Sensitivity**: defined as the *probability* to have at least one *hit* (seed occurrence) inside an alignment.
The main question in (most of) these papers: how to choose the best seed ...

**Sensitivity**: defined as the *probability* to have at least one *hit* (seed occurrence) inside an alignment.

**Best Seed**: defined as the one that maximize the sensitivity (among the seeds of a given class).
Need to determine the *language* recognized by a given seed $\pi$. 
Need to determine the *language* recognized by a given seed $\pi$.

**Alignment:** word on the binary match/mismatch alphabet
Need to determine the \textit{language} recognized by a given seed $\pi$.

\textbf{Alignment} : word on the binary match/mismatch alphabet
\textbf{Notation} : $\{1, 0\}$
Estimating Seed Sensitivity
using the Aho-Corasick automaton (Buhler et al 2003)

seed pattern : #−#−#
Estimating Seed Sensitivity
using the Aho-Corasick automaton (Buhler et al 2003)

seed pattern: #-#-#

words matched: \{10101, 10111, 11101, 11111\}
Estimating Seed Sensitivity using the Aho-Corasick automaton (Buhler et al 2003)

seed pattern: #−#−#

words matched: \{10101, 10111, 11101, 11111\}
Estimating Seed Sensitivity
using the Aho-Corasick automaton (Buhler et al 2003)

seed pattern: #--#--#

words matched: {10101, 10111, 11101, 11111}
Two kinds of mismatches: transitions and transversions

Definition

Transitions are substitutions between purins \((A \leftrightarrow G)\) or between pyrimidins \((T \leftrightarrow C)\). Transitions are usually overrepresented mutations ...

- : is a transition symbol (noted \(h\)).
- . is a transversion symbol (noted \(0\)).

Example

\[
\begin{align*}
\text{ATCAGTGC} & \text{G} \text{AATGCGCAAGA} \\
\text{ATCAGCG} & \text{CAATGCTCAAGA} \\
\end{align*}
\]

\[
\begin{align*}
\text{ATCAG} & \text{TGC} \text{AATGCGCAAGA} \\
\text{ATCAGCG} & \text{CAATGCTCAAGA} \\
\end{align*}
\]

\[
\begin{align*}
\text{11111h11h11111011111} \\
\end{align*}
\]
A transition constrained seed $\pi$ is defined as a ternary word over the alphabet $\{#, @, -\}^*$ with:

- $#$: accepts only match symbol $\mid$,
- $-$: accepts all alignment symbols (joker),
- $@$: accepts match symbol $\mid$ or transition mismatch symbol :,

Example seed pattern: $\pi = ##@#-@##$

ATCAGTGC GAATGCGCAAGA
$| | | | : | | : | | | | . | | | |$
ATCAGCGCAATGCTCAAGA
A transition constrained seed $\pi$ is defined as a ternary word over the alphabet $\{#, @, -\}^*$ with:

- $#$: accepts only match symbol $\mid$,
- $-$: accepts all alignment symbols (joker),
- $@$: accepts match symbol $\mid$ or transition mismatch symbol $:\$,

Example

seed pattern: $\pi = $$$@$$-@@$"
Transition Constrained Seed \((YASS \ 04)\)

**Definition**

A transition constrained seed \(\pi\) is defined as a ternary word over the alphabet \(#, @, -\)* with:
- \(#\) : accepts only match symbol \(|\)
- \(-\) : accepts all alignment symbols (joker),
- \(@\) : accepts match symbol \(|\) or transition mismatch symbol \(:\)

**Example**

seed pattern : \(\pi = ##@#-@##\)

\[
\begin{array}{c}
\text{ATCAGTGCGAATGC}G\text{CAAGA} \\
\text{| | | | | : | | : | | | | | . | | | | |} \\
\text{ATCAGCGCAATGCTCAAGA}
\end{array}
\]
Transition Constrained Seed (YASS 04)

**Definition**

A transition constrained seed $\pi$ is defined as a ternary word over the alphabet $\{#, @, -\}^*$ with:

- # : accepts only match symbol $|$,
- - : accepts all alignment symbols (joker),
- @ : accepts match symbol $|$ or transition mismatch symbol :,

**Example**

seed pattern : $\pi = ##@#-@##$

```
##@#-@##
ATCAGTGCAGATGCGCAAGA
ATCAGCGCAAATGCTCAAGA
```
Transition Constrained Seed *(YASS 04)*

**Definition**

A transition constrained seed $\pi$ is defined as a ternary word over the alphabet $\{#, @, -\}^*$ with:

- $#$: accepts only match symbol $\mid$,
- $-$: accepts all alignment symbols (*joker*),
- $@$: accepts match symbol $\mid$ or transition mismatch symbol :.

**Example**

seed pattern: $\pi = ##@#-@##$

```
##@#-@##
ATCAGTGCAGAATGCGCAAGA
ATCAGCGCAATGCTCAAGA
```
Transition Constrained Seed (YASS 04)

Definition

A transition constrained seed $\pi$ is defined as a ternary word over the alphabet $\{#, @, -\}$ with:

- $#$: accepts only match symbol $\|$,
- $-$: accepts all alignment symbols (joker),
- $@$: accepts match symbol $\|$ or transition mismatch symbol $:$,

Example

seed pattern: $\pi = ##@#-@##$

```
##@#-@##
ATCAGTGCGAATGCGCAAGA
ATCAGCGCAATGCCTCAAGA
```
A transition constrained seed $\pi$ is defined as a ternary word over the alphabet $\{\#, @, -\}^*$ with:
- $\#$ : accepts only match symbol $|$,
- $-$ : accepts all alignment symbols (joker),
- $@$ : accepts match symbol $|$ or transition mismatch symbol $:\$.

Example

seed pattern: $\pi = \#\#@$#$-@#$

---

Gregory Kucherov$^1$, Laurent Noé$^1$, Mikhail Roytberg$^2$
Subset seeds
a general model for transition constrained seeds

Definition

- **Alignment alphabet**: \( \mathcal{A} = \{1, \ldots \} \)
- **Seed alphabet**: \( \mathcal{B} = \{1\} \cup 2^\mathcal{A} \)

Example

- \( \mathcal{A} = \{1, h, 0\} \) where \( h \) is a transition mismatch, \( 0 \) a transversion mismatch.
- \( \mathcal{B} = \{\# , @, -\} \) with
  
  \[
  \begin{align*}
  \# &= \{1\} \\
  @ &= \{1, h\} \\
  - &= \{1, h, 0\} 
  \end{align*}
  \]
seed pattern: `#@-#`
Subset seeds
automaton with the Aho-Corasick approach

seed pattern : #@–#

words matched : \{1h01, 1hh1, 1h11, 1101, 11h1, 1111\}
seed pattern : #@-#  
words matched : \{1h01,  
1hh1,  
1h11,  
1101,  
11h1,  
1111\}  

- The resulting automaton size is $O(w|A|^{s-w})$, where $A$ is the alignment alphabet, $s$ the span of $\pi$, and $w$ the number of # in $\pi$.  
- ...
**Knuth-Morris-Pratt idea**: After reading a prefix of an alignment, what are the prefixes of seed $\pi$ that can be potentially extended into a match?
Subset seeds

how to build an efficient (i.e. small) subset seed automaton?

- **Knuth-Morris-Pratt idea**: After reading a prefix of an alignment, what are the prefixes of seed $\pi$ that can be potentially extended into a match?
- Since 1 is matched by any seed letter, 1s at the end don’t need to be considered
Subset seeds
how to build an efficient (i.e. small) subset seed automaton?

- **Knuth-Morris-Pratt idea**: After reading a prefix of an alignment, what are the prefixes of seed $\pi$ that can be potentially extended into a match?

- Since 1 is matched by any seed letter, 1s at the end don’t need to be considered

$\Rightarrow$ focus on alignments that don’t end with 1.
Subset seeds

how to build an efficient (i.e. small) subset seed automaton?

- **Knuth-Morris-Pratt idea**: After reading a prefix of an alignment, what are the prefixes of seed $\pi$ that can be potentially extended into a match?

- Since 1 is matched by any seed letter, 1s at the end don’t need to be considered

  $\longrightarrow$ focus on alignments that don’t end with 1.

  $\Longrightarrow$ focus on seed prefixes $R_\pi$ that don’t end with #.
Subset seeds
how to build an efficient (i.e. small) subset seed automaton?

\[ \pi : \#@-@-## \]

- **Knuth-Morris-Pratt idea:** After reading a prefix of an alignment, what are the prefixes of seed \( \pi \) that can be potentially extended into a match?

- Since 1 is matched by any seed letter, 1s at the end don’t need to be considered

  \[ \rightarrow \] focus on alignments that don’t end with 1.

  \[ \Rightarrow \] focus on seed prefixes \( R_\pi \) that don’t end with #.
Subset seeds
how to build an efficient (i.e. small) subset seed automaton?

- **Knuth-Morris-Pratt idea**: After reading a prefix of an alignment, what are the prefixes of seed $\pi$ that can be potentially extended into a match?

- Since 1 is matched by any seed letter, 1s at the end don’t need to be considered
  - focus on alignments that don’t end with 1.

 .focus on seed prefixes $R_\pi$ that don’t end with #.

$\pi : \#@-#@-##$

101h111h11

Gregory Kucherov, Laurent Noé, Mikhail Roytberg

Subset Seed Automaton
Subset seeds
how to build an efficient (i.e. small) subset seed automaton?

• *Knuth-Morris-Pratt idea:* After reading a prefix of an alignment, what are the prefixes of seed $\pi$ that can be potentially extended into a match?

• Since 1 is matched by any seed letter, 1s at the end don’t need to be considered

--- focus on alignments that don’t end with 1.

--- focus on seed prefixes $R_\pi$ that don’t end with #.
Subset seeds

how to build an efficient (i.e. small) subset seed automaton?

- **Knuth-Morris-Pratt idea:** After reading a prefix of an alignment, what are the prefixes of seed $\pi$ that can be potentially extended into a match?
- Since 1 is matched by any seed letter, 1s at the end don’t need to be considered
  - focus on alignments that don’t end with 1.
  - focus on seed prefixes $R_\pi$ that don’t end with #.

$\pi : \#\@-\#\@-##$

101h111h11

$\#\@-\#\@-##$

$\#\@-\#\@-\#$

$\#\@-\#\@$

$\#\@-\#$

$\#\@$

$\#$
**Knuth-Morris-Pratt idea:** After reading a prefix of an alignment, what are the prefixes of seed $\pi$ that can be potentially extended into a match?

- Since 1 is matched by any seed letter, 1s at the end don’t need to be considered
- focus on alignments that don’t end with 1.
- focus on seed prefixes $R_\pi$ that don’t end with #.
Subset seeds
how to build an efficient (i.e. small) subset seed automaton?

Knuth-Morris-Pratt idea: After reading a prefix of an alignment, what are the prefixes of seed \( \pi \) that can be potentially extended into a match?

- Since 1 is matched by any seed letter, 1s at the end don’t need to be considered
- focus on alignments that don’t end with 1.
- focus on seed prefixes \( R_\pi \) that don’t end with #.

\[
\begin{align*}
\pi & : \#@@-@@-### \\
101h111h1 & \\
#@@-@@-## & \\
#@@-@@-# & \\
#@@-@ & \\
#@-# & \\
#@ & \\
# & \\
\end{align*}
\]

\[
R_\pi = \{2, 3, 5, 6\} 
\]
Subset seeds
how to build an efficient (i.e. small) subset seed automaton?

- **Knuth-Morris-Pratt idea**: After reading a prefix of an alignment, what are the prefixes of seed $\pi$ that can be potentially extended into a match?
- Since 1 is matched by any seed letter, 1s at the end don’t need to be considered
  - focus on alignments that don’t end with 1.
  - focus on seed prefixes $R_\pi$ that don’t end with #.

\[\pi : \#@-##@-##@\]

101h111h11
#@-#@-##
#@-#@-##
#@-#@-##
#@-#@-##
#@-#@-##
#@
#

\[R_\pi = \{2, 3, 5, 6\}\]
Subset seeds
how to build an efficient (i.e. small) subset seed automaton?

- **Knuth-Morris-Pratt idea**: After reading a prefix of an alignment, what are the prefixes of seed $\pi$ that can be potentially extended into a match?
- Since 1 is matched by any seed letter, 1s at the end don’t need to be considered
  - focus on alignments that don’t end with 1.
  - focus on seed prefixes $R_\pi$ that don’t end with #.

\[ \pi : \text{#@-#@-##} \]
\[ 10111111 \]
\[ \text{#@-#@-##} \]
\[ \text{#@-#@-##} \]
\[ \text{#@-#@-##} \]
\[ \text{#@-#@-##} \]
\[ \text{#@-#@-##} \]
\[ \text{#@-#@-##} \]
\[ \text{#@-#@-##} \]
\[ \text{#@-@} \]
\[ \text{#@} \]
\[ \text{#} \]

\[ R_\pi = \{2, 3, 5, 6\} \]
State definition:

$q$ is defined as a pair $⟨X, t⟩$:

- $X \subseteq R_\pi$ : a subset of the prefixes of $\pi$ that don’t end with #.
- $t \in [1..s]$ : length of the last run of 1.

Final states:

states $⟨X, t⟩$ such that $\max\{X\} + t \geq s$

$\pi : \#@-#@-##$

$R_\pi = \{2, 3, 5, 6\}$
State definition:

$q$ is defined as a pair $\langle X, t \rangle$:

- $X \subseteq R_\pi$: a subset of the prefixes of $\pi$ that don’t end with $\#$.
- $t \in [1..s]$: length of the last run of $1$.

Final states:

states $\langle X, t \rangle$ such that $\max\{X\} + t \geq s$
State definition:

$q$ is defined as a pair $\langle X, t \rangle$:

- $X \subseteq R_{\pi}$: a subset of the prefixes of $\pi$ that don’t end with #.
- $t \in [1..s]$: length of the last run of 1.

Final states:

states $\langle X, t \rangle$ such that $\max \{X\} + t \geq s$

$\pi : \#01@\#@\#@\#$

$R_{\pi} = \{2, 3, 5, 6\}$

$q : \begin{cases} X & = \{2, 3, 6\} \\
         t & = \ 2 \end{cases}$
Transition function:

\( \psi(q, a) \) (\( q : \langle X, t \rangle \in Q, a \in A \)) is defined as:

\[
\text{if } a = 1 \text{ then } \\
\psi(\langle X, t \rangle, 1) = \langle X, t + 1 \rangle \quad (1) \\
\text{else} \\
\psi(\langle X, t \rangle, a) = \langle Y, 0 \rangle \quad (2)
\]
Transition function:

\[ \psi(q, a) \ (q : \langle X, t \rangle \in Q, a \in A) \] is defined as:

\[
\text{if } a = 1 \text{ then } \\
\psi(\langle X, t \rangle, 1) = \langle X, t + 1 \rangle \quad (1)
\]

\[
\text{else } \\
\psi(\langle X, t \rangle, a) = \langle Y, 0 \rangle \quad (2)
\]

(1) Prefix \(1^t\) is incremented

(2) \(Y\) updates the prefixes of \(X\) after reading \(1^t \cdot a\)
Transition function:

\[ \psi(q, a) \ (q : \langle X, t \rangle \in Q, a \in A) \] is defined as:

1. If \( a = 1 \):
   \[ \psi(\langle X, t \rangle, 1) = \langle X, t + 1 \rangle \] (1)

2. Else:
   \[ \psi(\langle X, t \rangle, a) = \langle Y, 0 \rangle \] (2)

(1) Prefix 1\(^t\) is incremented

(2) Y updates the prefixes of X after reading 1\(^t\) \cdot a
Transition function:

\( \psi(q, a) \) (\( q : \langle X, t \rangle \in Q, a \in A \)) is defined as:

\[
\begin{align*}
\text{if } a = 1 \text{ then } \\
\psi(\langle X, t \rangle, 1) &= \langle X, t + 1 \rangle \quad (1) \\
\text{else } \\
\psi(\langle X, t \rangle, a) &= \langle Y, 0 \rangle \quad (2)
\end{align*}
\]

(1) Prefix \( 1^t \) is incremented

(2) \( Y \) updates the prefixes of \( X \) after reading \( 1^t \cdot a \)

\[
q : \begin{cases} 
X = \{2, 3, 6\} \\
t = 0
\end{cases}
\]
Transition function:

\( \psi(q, a) \) \((q : \langle X, t \rangle \in Q, a \in A)\) is defined as:

\[
\text{if } a = 1 \text{ then }
\psi(\langle X, t \rangle, 1) = \langle X, t + 1 \rangle \quad (1)
\]

\[
\text{else }
\psi(\langle X, t \rangle, a) = \langle Y, 0 \rangle \quad (2)
\]

(1) Prefix \(1^t\) is incremented

(2) \(Y\) updates the prefixes of \(X\) after reading \(1^t \cdot a\)
Transition function:

\[ \psi(q, a) \ (q : \langle X, t \rangle \in Q, a \in \mathcal{A}) \text{ is defined as:} \]

if \( a = 1 \) then

\[ \psi(\langle X, t \rangle, 1) = \langle X, t + 1 \rangle \quad (1) \]

else

\[ \psi(\langle X, t \rangle, a) = \langle Y, 0 \rangle \quad (2) \]

(1) Prefix \( 1^t \) is incremented

(2) \( Y \) updates the prefixes of \( X \) after reading \( 1^t \cdot a \)

\[
\pi : \begin{array}{c}
\# @ - \# @ -
\# @ - \# @ - \# \\
# @ - \# @ - \\
# @ - \\
\end{array}
\]

\[
101 h111 h1
\]

\[
\begin{array}{c}
\# @ - \# @ - \# \\
# @ - \# @ - \\
# @ - \\
\end{array}
\]

\[
\begin{array}{c}
\# @ - \# @ - \\
# @ - \\
\# @ -
\end{array}
\]

\[
\begin{array}{c}
\# @ - \\
\# @ -
\end{array}
\]

\[
\begin{array}{c}
\# @ -
\end{array}
\]

\[
\begin{array}{c}
\#
\end{array}
\]

\[
q : \begin{cases}
X = \{2, 3, 6\} \\
t = 1
\end{cases}
\]
Subset seed automaton

Transition function:

\[ \psi(q, a) \ (q : \langle X, t \rangle \in Q, a \in A) \] is defined as:

\[ \text{if } a = 1 \text{ then} \]
\[ \psi(\langle X, t \rangle, 1) = \langle X, t + 1 \rangle \quad (1) \]

\[ \text{else} \]
\[ \psi(\langle X, t \rangle, a) = \langle Y, 0 \rangle \quad (2) \]

(1) Prefix $1^t$ is incremented

(2) $Y$ updates the prefixes of $X$ after reading $1^t \cdot a$

\[ q : \begin{cases} 
X = \{2, 3, 6\} \\
t = 1 
\end{cases} \]

\[ \pi : \#\@-\#\@-## \]

101h111h1h
#\@-#\@-##
#\@-#\@-##
#\@-#\@-##
#\@-#\@-##
Transition function:

\[ \psi(q, a) \ (q : \langle X, t \rangle \in Q, \ a \in A) \text{ is defined as:} \]

\[
\begin{align*}
\text{if} \quad a &= 1 \quad \text{then} \\
\psi(\langle X, t \rangle, 1) &= \langle X, t + 1 \rangle \\ 
\text{else} \\
\psi(\langle X, t \rangle, a) &= \langle Y, 0 \rangle
\end{align*}
\]

(1) Prefix \( 1^t \) is incremented

(2) \( Y \) updates the prefixes of \( X \) after reading \( 1^t \cdot a \)

\[ q : \begin{cases} 
X &= \{2, 3, 6\} \\
t &= 1
\end{cases} \]
Transition function:

\[ \psi(q, a) \ (q : \langle X, t \rangle \in Q, \ a \in A) \text{ is defined as:} \]

\[
\begin{align*}
\text{if } a &= 1 \text{ then} \\
\psi(\langle X, t \rangle, 1) &= \langle X, t + 1 \rangle \quad (1) \\
\text{else} \\
\psi(\langle X, t \rangle, a) &= \langle Y, 0 \rangle \quad (2)
\end{align*}
\]

(1) Prefix \(1^t\) is incremented

(2) \(Y\) updates the prefixes of \(X\) after reading \(1^t \cdot a\)

\[ q : \left\{ \begin{array}{c} X = \{2, 3, 6\} \\ t = 1 \end{array} \right. \]
Transition function:

\[ \psi(q, a) \ (q : \langle X, t \rangle \in Q, a \in A) \text{ is defined as:} \]

\[
\begin{align*}
\text{if} \quad a &= 1 \quad \text{then} \\
\psi(\langle X, t \rangle, 1) &= \langle X, t + 1 \rangle \quad (1)
\end{align*}
\]

\[
\begin{align*}
\text{else} \\
\psi(\langle X, t \rangle, a) &= \langle Y, 0 \rangle \quad (2)
\end{align*}
\]

(1) Prefix \( 1^t \) is incremented

(2) \( Y \) updates the prefixes of \( X \) after reading \( 1^t \cdot a \)

\[ q : \begin{cases} 
X &= \begin{cases} 2, 3, 6 \end{cases} \\
t &= 1
\end{cases} \]
Subset seed automaton

Transition function:

\[ \psi(q, a) \ (q : \langle X, t \rangle \in Q, \ a \in A) \text{ is defined as:} \]

\[
\begin{align*}
\text{if } a = 1 & \text{ then} \\
\psi(\langle X, t \rangle, 1) &= \langle X, t + 1 \rangle \quad (1) \\
\text{else} & \\
\psi(\langle X, t \rangle, a) &= \langle Y, 0 \rangle \quad (2)
\end{align*}
\]

(1) Prefix 1\(^t\) is incremented
(2) Y updates the prefixes of X after reading 1\(^t\) \cdot a
Subset seed automaton

transition function

Transition function:

\( \psi(q, a) \) \( (q : \langle X, t \rangle \in Q, a \in A) \) is defined as:

```latex
\begin{align*}
\text{if } a = 1 \text{ then } \\
\psi(\langle X, t \rangle, 1) &= \langle X, t + 1 \rangle \quad (1) \\
\text{else} \\
\psi(\langle X, t \rangle, a) &= \langle Y, 0 \rangle \quad (2)
\end{align*}
```

(1) Prefix \( 1^t \) is incremented

(2) \( Y \) updates the prefixes of \( X \) after reading \( 1^t \cdot a \)

\[
q : \begin{cases} 
Y &= \{2, 5\} \\
t &= 0
\end{cases}
\]
Automaton construction

- incremental construction of states (breadth-first)
Automaton construction

- incremental construction of states (breadth-first)
- **difficulty**: keeping track and retrieving already created states

Reachable states computed in constant time

Ideas behind:

- **Fail** function (≈ Aho-Corasick):
  \[ \text{Fail}(⟨X, t⟩) = ⟨X', t⟩ \]

- **RevMaxFail**:
  \[ \text{RevMaxFail}(⟨X', t⟩) = \text{last created state} ⟨X, t⟩ \]

\[ X' = X \setminus \max \{X\} \]
Automaton construction

- incremental construction of states (breadth-first)
- **difficulty**: keeping track and retrieving already created states

Reachable states computed in constant time

Ideas behind:

- **Fail** function (≈ Aho-Corasick):

  \[
  \text{Fail}(\langle X, t \rangle) = \langle X', t \rangle \text{ with } X' = X \setminus \max\{X\}
  \]

- **Fail^{-1}**:

  \[
  \text{RevMaxFail}(\langle X', t \rangle) = \text{last created state } \langle X, t \rangle \text{ with } X' = X \setminus \max\{X\}
  \]
Subset seed automaton
reachable states computed constant time

\[ q : \begin{cases} 
X & = \{2, 3, 6\} \\
t & = 1 
\end{cases} \quad \xrightarrow{a} \quad 
q_r : \begin{cases} 
X & = \{2, 5, 8\} \\
t & = 0 
\end{cases} \]
Subset seed automaton

reachable states computed constant time

\[
q : \begin{cases} \quad X = \{2, 3, 6\} \\
\quad t = 1 \end{cases} \rightarrow_a \quad q_r : \begin{cases} \quad X = \{2, 5, 8\} \\
\quad t = 0 \end{cases}
\]

\[
q' : \begin{cases} \quad X = \{2, 3\} \\
\quad t = 1 \end{cases} \rightarrow_a \quad q'_r : \begin{cases} \quad X = \{2, 5\} \\
\quad t = 0 \end{cases}
\]

Gregory Kucherov\textsuperscript{1}, Laurent No\textsuperscript{e}\textsuperscript{1}, Mikhail Roytberg\textsuperscript{2}

\textbf{Subset Seed Automaton}
The number of states is $O(w2^{s-w})$, where $s$ the span of $\pi$, and $w$ the number of # in $\pi$. The base of the exponent does not depend on $A$. The automaton is strictly smaller than the AC automaton as there is a morphism from states of $S_\pi \rightarrow AC_\pi$. In practice, it is much smaller for $|A| > 2$ and smaller even for $|A| = 2$ (ordinary spaced seeds). The automaton admits an efficient incremental construction such that each transition is computed in constant time.
The number of states is $O(w2^{s-w})$, where $s$ the span of $\pi$, and $w$ the number of # in $\pi$.

The base of the exponent does not depend on $A$. 

In practice, it is much smaller for $|A| > 2$ and smaller even for $|A| = 2$ (ordinary spaced seeds).

The automaton admits an efficient incremental construction such that each transition is computed in constant time.
The number of states is $O(w2^{s-w})$, where $s$ the span of $\pi$, and $w$ the number of $#$ in $\pi$.

The base of the exponent does not depend on $A$.

The automaton is strictly smaller than the AC automaton as there is a morphism from states of $S_\pi \rightarrow AC_\pi$. In practice, it is much smaller for $|A| > 2$ and smaller even for $|A| = 2$ (ordinary spaced seeds).
The number of states is $O(w2^{s-w})$, where $s$ the span of $\pi$, and $w$ the number of # in $\pi$.

The base of the exponent does not depend on $\mathcal{A}$.

The automaton is strictly smaller than the AC automaton as there is a morphism from states of $S_\pi \rightarrow AC_\pi$.

In practice, it is much smaller for $|\mathcal{A}| > 2$ and smaller even for $|\mathcal{A}| = 2$ (ordinary spaced seeds).
• The number of states is $O(w2^{s-w})$, where $s$ the span of $\pi$, and $w$ the number of # in $\pi$.

• The base of the exponent does not depend on $A$.

• The automaton is strictly smaller than the AC automaton as there is a morphism from states of $S_\pi \to AC_\pi$.

• In practice, it is much smaller for $|A| > 2$ and smaller even for $|A| = 2$ (ordinary spaced seeds).

• The automaton admits an efficient incremental construction such that each transition is computed in constant time.
The number of states is $O(w^{2^{s-w}})$, where $s$ is the span of $\pi$, and $w$ the number of $\#$ in $\pi$.

The base of the exponent does not depend on $A$.

The automaton is strictly smaller than the AC automaton as there is a morphism from states of $S_\pi \rightarrow AC_\pi$.

In practice, it is much smaller for $|A| > 2$ and smaller even for $|A| = 2$ (ordinary spaced seeds).

The automaton admits an efficient incremental construction such that each transition is computed in constant time.
Subset seed automaton

Average size of $S_\pi$

- Comparison of Aho-Corasick automaton [Buhler, Keich, Sun] vs our automaton vs minimal automaton.
- Average size measured on a significant set of seeds.
Subset seed automaton

Average size of $S_\pi$

| $|A| = 2$ | Aho-Corasick | $S_\pi$ | Minimized |
|---------|-------------|--------|-----------|
|         | $w$ | avg. | ratio | avg. | ratio | avg. |
| 9       |     | 130.98 | 2.46 | 67.03 | 1.260 | 53.18 |
| 10      |     | 140.28 | 2.51 | 70.27 | 1.255 | 55.98 |
| 11      |     | 150.16 | 2.55 | 73.99 | 1.254 | 58.99 |
| 12      |     | 159.26 | 2.57 | 77.39 | 1.248 | 62.00 |
| 13      |     | 168.19 | 2.59 | 80.92 | 1.246 | 64.92 |

| $|A| = 3$ | Aho-Corasick | $S_\pi$ | Minimized |
|---------|-------------|--------|-----------|
|         | $w$ | avg. | ratio | avg. | ratio | avg. |
| 9       |     | 1103.5 | 16.46 | 86.71 | 1.293 | 67.05 |
| 10      |     | 1187.7 | 16.91 | 90.67 | 1.291 | 70.25 |
| 11      |     | 1265.3 | 17.18 | 95.05 | 1.291 | 73.65 |
| 12      |     | 1346.1 | 17.50 | 98.99 | 1.287 | 76.90 |
| 13      |     | 1419.3 | 17.67 | 103.10 | 1.284 | 80.31 |

Table: Average number of states of Aho-Corasick, $S_\pi$ and minimal automaton

| $|A| = 2$ | Aho-Corasick | $S_\pi$ | Minimized |
|---------|-------------|--------|-----------|
|         | $w$ | avg. | ratio | avg. | ratio | avg. |
| 9       |     | 224.49 | 2.01 | 122.82 | 1.10 | 111.43 |
| 10      |     | 243.32 | 2.07 | 129.68 | 1.10 | 117.71 |
| 11      |     | 264.04 | 2.11 | 137.78 | 1.10 | 125.02 |
| 12      |     | 282.51 | 2.15 | 144.97 | 1.10 | 131.68 |
| 13      |     | 300.59 | 2.18 | 151.59 | 1.10 | 137.74 |

| $|A| = 3$ | Aho-Corasick | $S_\pi$ | Minimized |
|---------|-------------|--------|-----------|
|         | $w$ | avg. | ratio | avg. | ratio | avg. |
| 9       |     | 2130.6 | 12.09 | 201.69 | 1.15 | 176.27 |
| 10      |     | 2297.8 | 12.53 | 209.75 | 1.14 | 183.40 |
| 11      |     | 2456.5 | 12.86 | 218.27 | 1.14 | 191.04 |
| 12      |     | 2600.6 | 13.14 | 226.14 | 1.14 | 198.00 |
| 13      |     | 2778.0 | 13.39 | 236.62 | 1.14 | 207.51 |

Table: Average number of states of Aho-Corasick, $S_\pi$ and minimized automata for the case of two seeds
## Subset seed automaton

Average size of $S_\pi$

| $|\mathcal{A}| = 2$ | Aho-Corasick | $S_\pi$ | Minimized | $|\mathcal{A}| = 3$ | Aho-Corasick | $S_\pi$ | Minimized |
|----------------|--------------|---------|-----------|----------------|--------------|---------|-----------|
|                | $w$ | avg. | ratio | avg. | ratio | avg. | ratio | avg. | ratio | avg. | ratio | avg. | ratio |
| 9              | 130.98 | 2.46 | 67.03 | 1.260 | 53.18 |
| 10             | 140.28 | 2.51 | 70.27 | 1.255 | 55.98 |
| 11             | 150.16 | 2.55 | 73.99 | 1.254 | 58.99 |
| 12             | 159.26 | 2.57 | 77.39 | 1.248 | 62.00 |
| 13             | 168.19 | 2.59 | 80.92 | 1.246 | 64.92 |
| 9              | 1103.5 | 16.46 | 86.71 | 1.10 | 67.05 |
| 10             | 1187.7 | 16.91 | 90.67 | 1.10 | 70.25 |
| 11             | 1265.3 | 17.18 | 95.05 | 1.10 | 73.65 |
| 12             | 1346.1 | 17.50 | 98.99 | 1.14 | 76.90 |
| 13             | 1419.3 | 17.67 | 103.10 | 1.14 | 80.31 |

### Table: Average number of states of Aho-Corasick, $S_\pi$ and minimal automaton

| $|\mathcal{A}| = 2$ | Aho-Corasick | $S_\pi$ | Minimized | $|\mathcal{A}| = 3$ | Aho-Corasick | $S_\pi$ | Minimized |
|----------------|--------------|---------|-----------|----------------|--------------|---------|-----------|
|                | $w$ | avg. | ratio | avg. | ratio | avg. | ratio | avg. | ratio | avg. | ratio | avg. | ratio |
| 9              | 224.49 | 2.01 | 122.82 | 1.10 | 111.43 |
| 10             | 243.32 | 2.07 | 129.68 | 1.10 | 117.71 |
| 11             | 264.04 | 2.11 | 137.78 | 1.10 | 125.02 |
| 12             | 282.51 | 2.15 | 144.97 | 1.10 | 131.68 |
| 13             | 300.59 | 2.18 | 151.59 | 1.10 | 137.74 |
| 9              | 2130.6 | 12.09 | 201.69 | 1.15 | 176.27 |
| 10             | 2297.8 | 12.53 | 209.75 | 1.14 | 183.40 |
| 11             | 2456.5 | 12.86 | 218.27 | 1.14 | 191.04 |
| 12             | 2600.6 | 13.14 | 226.14 | 1.14 | 198.00 |
| 13             | 2778.0 | 13.39 | 236.62 | 1.14 | 207.51 |

### Table: Average number of states of Aho-Corasick, $S_\pi$ and minimal automata for the case of two seeds
That’s it

We have shown:

1. what is the subset seed model ... 
2. how to efficiently build the subset seed automaton ... 
3. some practical experiments on it ...

Possible practical application: IUPAC patterns

\[ \text{[GA][GA]GGGNNNNAN[CT]ATGNN[AT]NNNNN[CTG]} \rightarrow 138 \text{ states (minimized:127)} \]
We have shown:

1. what is the subset seed model ...
2. how to efficiently build the subset seed automaton ...
3. some practical experiments on it ...

Possible practical application: IUPAC patterns
That’s it

We have shown:

1. what is the subset seed model ...
2. how to efficiently build the subset seed automaton ...
3. some practical experiments on it ...

Possible practical application: IUPAC patterns

\[ [GA] [GA] GGGNNNNAN [CT] ATGNN [AT] NNNNN [CTG] \]
That’s it

We have shown:
1. what is the subset seed model ...
2. how to efficiently build the subset seed automaton ...
3. some practical experiments on it ...

Possible practical application: IUPAC patterns

\[[GA] [GA] GGGNNNAN [CT] ATGNN [AT] NNNNN [CTG]\]
→ 138 states (minimized:127).
Subset seed automaton

a very small example ...

seed pattern:

```
#-@#
```
Subset seed automaton

a very small example ...

seed pattern:

#-@#

words matched:

\{10h1,
  1011,
  1hh1,
  1h11,
  11h1,
  1111\}
Subset seed automaton

seed pattern:

```
#-@#
```

words matched:

```
{10h1, 1011, 1hh1, 1h11, 11h1, 1111}
```
Subset seed automaton

a very small example ...

seed pattern:
	`#-@#`

words matched:

{10h1, 1011, 1hh1, 1h11, 11h1, 1111}
Subset seed automaton

a very small example ...

seed pattern:

#-@#

words matched:

\{10h1, 1011, 1hh1, 1h11, 11h1, 1111\}
Subset seed automaton

a very small example ...

seed pattern:

```
#-@#
```

words matched:

```
{10h1, 1011, 1hh1, 1h11, 11h1, 1111}
```
Subset seed automaton

seed pattern:

#-@#

words matched:

\{10h1, 1011, 1hh1, 1h11, 11h1, 1111\}
seed pattern: 

```
#-@#
```

words matched: 

```
{10h1, 1011, 1hh1, 1h11, 11h1, 1111}
```
Subset seed automaton

a very small example ...

seed pattern :

```
#-@#
```

words matched :

```
{10h1,
  1011,
  1hh1,
  1h11,
  11h1,
  1111}
```
Subset seed automaton

a very small example ...

seed pattern:

# - @#

words matched:

\{ 10h1, 1011, 1hh1, 1h11, 11h1, 1111 \}

\[ X = \emptyset \]
\[ t = 0 \]
Subset seed automaton

a very small example ...

seed pattern :

#-@#

words matched :

\{10h1, 1011, 1hh1, 1h11, 11h1, 1111\}
Subset seed automaton

a very small example ...

seed pattern:

`#-@#`

words matched:

`{10h1, 1011, 1hh1, 1h11, 11h1, 1111}`
seed pattern:

#-@#

words matched:

\{10h1, 1011, 1hh1, 1h11, 11h1, 1111\}
seed pattern: 

```
#-@#
```

words matched:

{10h1, 1011, 1hh1, 1h11, 11h1, 1111}
Subset seed automaton

a very small example ...

seed pattern :

#-@#

words matched :

\{10h1,
1011,
1hh1,
1h11,
11h1,
1111\}
Subset seed automaton

seed pattern:

#@-#
Subset seed automaton

seed pattern:

```plaintext
#@-#
```

words matched:

```plaintext
{1h01, 1hh1, 1h11, 1101, 11h1, 1111}
```
seed pattern:

`#@-#`

words matched:

\{ 1h01, 1hh1, 1h11, 1101, 11h1, 1111 \}
Subset seed automaton

a very small example ...

seed pattern :

#@-#

words matched :

\{1h01, 1hh1, 1h11, 1101, 11h1, 1111\}

\[\psi(q_3, 0) = ?\]
Subset seed automaton

a very small example ...

seed pattern:

```
#@-#
```

words matched:

```
{1h01, 1hh1, 1h11, 1101, 11h1, 1111}
```

\[
\psi(q_3, 0) = \langle X = \{3\}, t = 0 \rangle
\]
Subset seed automaton

a very small example ...

seed pattern :

# @ - #

words matched :

\{ 1h01, 1hh1, 1h11, 1101, 11h1, 1111 \}

\( \psi(q_3, 0) = \langle X = \{3\}, t = 0 \rangle \)
Subset seed automaton

a very small example ...

seed pattern:
#
@-
#

words matched:
{1h01, 1hh1, 1h11, 1101, 11h1, 1111}

\[ \psi(q_3, 0) = \langle X = \{3\}, t = 0 \rangle \]
Subset seed automaton

a very small example ...

seed pattern :

#@-#

words matched :

{1h01,
  1hh1,
  1h11,
  1101,
  11h1,
  1111}

\[ \psi(q_3, 0) = \langle X = \{3\}, t = 0 \rangle \]
Subset seed automaton
a very small example ...

seed pattern :
  #@-#

words matched :
  {1h01, 1hh1, 1h11, 1101, 11h1, 1111}

ψ(q₃, 0) = ⟨X = {3}, t = 0⟩
Subset seed automaton

a very small example ...

seed pattern:

# @-#

words matched:

\{ 1h01, 1hh1, 1h11, 1101, 11h1, 1111 \}

\( \psi(q_3, 0) = \langle X = \{3\}, t = 0 \rangle \)
Subset seed automaton

a very small example ...

seed pattern :

#@-#

words matched :

\{1h01, 1hh1, 1h11, 1101, 11h1, 1111\}

\(\psi(q_3, 0) = \langle X = \{3\}, t = 0 \rangle\)