

Subset Seed Automaton

Gregory Kucherov¹, **Laurent Noé**¹, Mikhail Roytberg²

¹ LIFL, Université Lille 1 - INRIA (France)

² Institute of Mathematical Problems in Biology, Pushchino (Moscow region)

CIAA 2007, Prague, July 16-18 2007

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

```
TTTTGAACTGGGACGAAAGTGCATCAGTGCAAATGCGCAAGAAAAA  
CGCCGAACGCTTCAGATCAGCGCAAATGCTCAAGAGGTCTCGTCGC  
TGAGGCACTACGGCCAGCCGAGCCAGTCAT
```

Sequence Alignment

on a very small DNA example

```
TTTTGAACTGGGACGAAAGTGCATCAGTGCAAATGCGCAAGAAAA  
CGCCGAACGCTTCAGATCAGCGCAAATGCTCAAGAGGTCTCGTCGC  
TGAGGCACTACGGCCAGCCGAGCCAGTCAT
```

Sequence Alignment

on a very small DNA example

```
TTTTGAACTGGGACGAAAGTGCATCAGTGCAAATGCGCAAGAAAAA  
CGCCGAACGCTTCAGATCAGCGCAAATGCTCAAGAGGTCTCGTCGC  
TGAGGCACTACGGCCAGCCGAGCCAGTCAT
```

Sequence Alignment

on a very small DNA example

```
TTTTGAACTGGGACGAAAGTGCATCAGTGCAAATGCGCAAGAAAAA  
CGCCGAACGCTTCAGATCAGCGCAAATGCTCAAGAGGTCTCGTCGC  
TGAGGCACTACGGCCAGCCGAGCCAGTCAT
```

ATCAGTGCAAATGCGCAAGA

ATCAGCGCAAATGCTCAAGA

Sequence Alignment

on a very small DNA example

```
TTTTGAACTGGGACGAAAGTGCATCAGTGCAAATGCGCAAGAAAAA  
CGCCGAACGCTTCAGATCAGCGCAAATGCTCAAGAGGTCTCGTCGC  
TGAGGCACTACGGCCAGCCGAGCCAGTCAT
```

```
ATCAGTGCAAATGCGCAAGA  
|||||:|||||||.|||||  
ATCAGCGCAAATGCTCAAGA
```

Sequence Alignment

methods used to solve this problem ...

Algorithm: Smith-Waterman algorithm (in $\mathcal{O}(n^2)$).

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 π detects one alignment motif of size k .

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

Example

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

$\#\#\#\#\#$

```
ATCAGTGCAAAATGCGCAAGA
| | | | : | | | | | | | | | |
ATCAGCGGCAAATGCTTCAAGA
```

Contiguous Seeds

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

Principle: A contiguous seed π detects one alignment motif of size k .

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

Example

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

$\#\#\#\#\#$

```
ATCAGTGCAAAATGCGCAAGA
| | | | : | | | | | | | | | |
ATCAGCGGCAAATGCTTCAAGA
```

Contiguous Seeds

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

Principle: A contiguous seed π detects one alignment motif of size k .

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

Example

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

$\#\#\#\#\#$

```
ATCAGTGCAAAATGCGCAAGA
| | | | : | | | | | | | | | |
ATCAGCGGCAAATGCTTCAAGA
```

Contiguous Seeds

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

Principle: A contiguous seed π detects one alignment motif of size k .

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

Example

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

$\#\#\#\#\#$

```
ATCAGTGCAAAATGCGCAAGA
| | | | : | | | | | | | | | |
ATCAGCGGCAAATGCTTCAAGA
```

Contiguous Seeds

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

Principle: A contiguous seed π detects one alignment motif of size k .

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

Example

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

$\#\#\#\#\#$

```
ATCAGTGCAAAATGCGCAAGA
| | | | : | | | | | | | | | |
ATCAGCGGCAAATGCTTCAAGA
```

Spaced Seeds

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

Definition

A spaced seed π 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 : $\pi = \#\#\#- \#- \#\#$

```
ATCAGTGC GAATGCGCAAGA
| | | | : | | : | | | | . | | | |
ATCAGCGCA AATGCTCAAGA
```

Spaced Seeds

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

Definition

A spaced seed π 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 : $\pi = \#\#\#- \#- \#\#$

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

```
ATCAGTGC GAATGCGCAAGA
||| |:| |:| |. ||| |
ATCAGCGCA AATGCTCAAGA
```

Spaced Seeds

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

Definition

A spaced seed π 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 : $\pi = \#\#\#-\#-\#\#$

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

```
ATCAGTGC GAATGCGCAAGA
||| |:| |:| |. ||| |
ATCAGCGCA AATGCTCAAGA
```

Spaced Seeds

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

Definition

A spaced seed π 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 : $\pi = \#\#\#- \#- \#\#$

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

```
ATCAGTGC GAATGCGCAAGA
||| |:| |:| |. ||| |
ATCAGCGCA AATGCTCAAGA
```

Spaced Seeds

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

Definition

A spaced seed π 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 : $\pi = \#\#\#- \#- \#\#$

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

```
ATCAGTGCGAATGCGCAAGA
||| |:| |:| |. ||| |
ATCAGCGCAAAATGCTCAAGA
```

Spaced Seeds

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

Definition

A spaced seed π 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 : $\pi = \#\#\#- \#- \#\#$

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

```
ATCAGTGC GAATGCGCAAGA
| | | | : | | : | | | | . | | | |
ATCAGCGC AAATGCTCAAGA
```

Spaced Seeds

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

Definition

A spaced seed π 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 : $\pi = \#\#\#- \#- \#\#$

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

```
ATCAGTGCGAATGCGCAAGA
||| |:| |:| |. ||| |
ATCAGCGCAAAATGCTCAAGA
```

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.)

- **Estimating the sensitivity of a seed:** Keich et al 2002, Buhler et al 2003, Brejova et al 2003, Choi et al 2004, Kucherov et al 2004, Mak&Benson 2007
- **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
- **Lossless (multiple) seeds:** Burkhardt&Karkkainen 2001, Kucherov et al 2004, Farach et al 2004, Fontaine et al 2004, Nicolas&Rivals 2005
- **Surveys:** Brown&Li&Ma 2005, Brown 2007

Spaced Seeds

How to choose the best one

The main question in (most of) these papers: how to choose the best *seed* ...

Spaced Seeds

How to choose the best one

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.

Spaced Seeds

How to choose the best one

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).

Spaced Seeds

Estimating Seed Sensitivity

Need to determine the *language* recognized by a given seed π .

Spaced Seeds

Estimating Seed Sensitivity

Need to determine the *language* recognized by a given seed π .

Alignment : word on the binary match/mismatch alphabet

Spaced Seeds

Estimating Seed Sensitivity

Need to determine the *language* recognized by a given seed π .

Alignment : word on the binary match/mismatch alphabet

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 : #-#-#

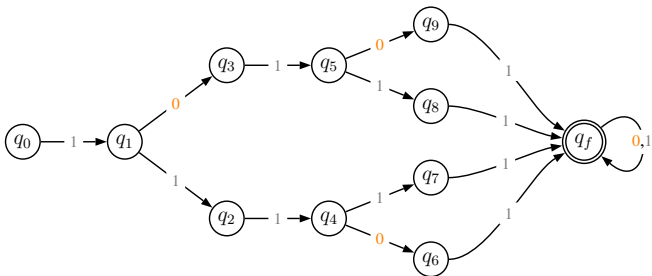
words matched : {10101,
10111,
11101,
11111}

Estimating Seed Sensitivity

using the Aho-Corasick automaton (Buhler et al 2003)

seed pattern : #-#-#

words matched : {10101,
10111,
11101,
11111}

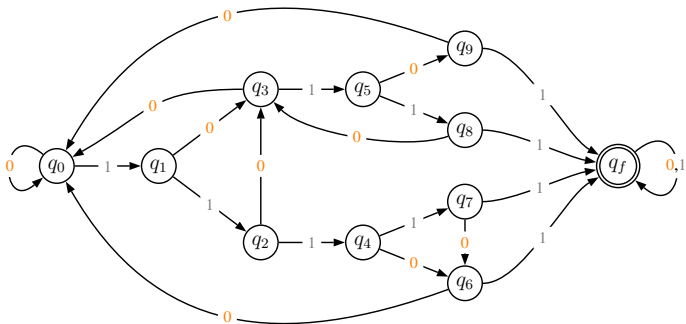


Estimating Seed Sensitivity

using the Aho-Corasick automaton (Buhler et al 2003)

seed pattern : #-#-#

words matched : {10101,
10111,
11101,
11111}



Mutations on DNA

Transitions and Transversions ...

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 ...

- \cdot is a transition symbol (noted **h**).
- \circ is a transversion symbol (noted **0**).

Example

```
ATCAGTGC GAATGC CAAGA
| | | | : | | : | | | | . | | | |
ATCAGCGC AAATGCT CAAGA
```

= 11111**h**11**h**11111**0**11111

Transition Constrained Seed (YASS 04)

Definition

A transition constrained seed π 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 : $\pi = \#\#@\#-\#@\#$

```
ATCAGTGC GAATGC CAAGA
| | | | : | | : | | | | . | | | |
ATCAGCGC AAATGCT CAAGA
```

Definition

A transition constrained seed π 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 : $\pi = \#\#@\#-\#@\#$

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

ATCAGTGC**G**AATGC**G**CAAGA

| | | | : | | : | | | | . | | | |

ATCAG**C**GC**A**AATGCT**C**AAGA

Definition

A transition constrained seed π 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
| | | | : | | : | | | | . | | | |
ATCAGCGCAAATGCTTCAAGA
```

Transition Constrained Seed (YASS 04)

Definition

A transition constrained seed π 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

|||||:| |:|||||.|||||

ATCAGCGCAAATGCTCAAGA

Transition Constrained Seed (YASS 04)

Definition

A transition constrained seed π 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 : $\pi = \#\#@\#-\@##$

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

```
ATCAGTGC GAATGCGCAAGA
| | | | : | | : | | | | . | | | |
ATCAGCGC AAATGCTCAAGA
```

Transition Constrained Seed (YASS 04)

Definition

A transition constrained seed π 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 : $\pi = \#\#@\#-\@##$

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

```
ATCAGTGC GAATGCGCAAGA
| | | | : | | : | | | | . | | | |
ATCAGCGC AAATGCTCAAGA
```

Definition

A transition constrained seed π 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 : $\pi = \#\#@\#-\@##$

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

```
ATCAGTGC GAATGCGCAAGA
| | | | : | | : | | | | . | | | |
ATCAGCGC AAATGCTCAAGA
```

Subset seeds

a general model for transition constrained seeds

Definition

- **Alignment alphabet** : $\mathcal{A} = \{1, \dots\}$
- **Seed alphabet** : $\mathcal{B} = \{1\} \cup 2^{\mathcal{A}}$

Example

- $\mathcal{A} = \{1, \mathbf{h}, \mathbf{0}\}$ where \mathbf{h} is a transition mismatch, $\mathbf{0}$ a transversion mismatch.
- $\mathcal{B} = \{\#, \mathbf{@}, \mathbf{-}\}$ with

$$\left\{ \begin{array}{l} \# = \{1\} \\ \mathbf{@} = \{1, \mathbf{h}\} \\ \mathbf{-} = \{1, \mathbf{h}, \mathbf{0}\} \end{array} \right.$$

Subset seeds

automaton with the Aho-Corasick approach

seed pattern : #@-#

Subset seeds

automaton with the Aho-Corasick approach

seed pattern : #@-#

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

Subset seeds

automaton with the Aho-Corasick approach

seed pattern : #@-#

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

- The resulting automaton size is $\mathcal{O}(w|\mathcal{A}|^{s-w})$, where \mathcal{A} is the alignment alphabet, s the span of π , and w the number of # in π .
- ...

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 π 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 π 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 π 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.

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 π 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_π that don't end with #.

Subset seeds

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

π : #@-#@-##

- *Knuth-Morris-Pratt idea*: After reading a prefix of an alignment, what are the prefixes of seed π 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_π 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 π 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_π that don't end with #.

π : #@-#@-##

101h111h11

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 π 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_π that don't end with #.

π : #@-#@-##

101h111h11

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 π 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_π that don't end with #.

π : #@-#@-##

101h111h11

#@-#@-##

#@-#@-#

#@-#@-

#@-#@

#@-#

#@-

#@

#

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 π 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_π that don't end with #.

π : #@-#@-##

101h111h11

#@-#@-##

#@-#@-#

#@-#@-

#@-#@

#@-#

#@-

#@

#

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 π 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_π that don't end with #.

π : #@-#@-##

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 π 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_π that don't end with #.

π : #@-#@-##

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 π 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_π that don't end with #.

π : #@-#@-##

101h111h11

#@-#@-##

#@-#@-#

#@-#@-

#@-#@

#@-#

#@-

#@

#

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

Subset seed automaton

state definition

State definition :

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

- $X \subseteq R_\pi$: a subset of the prefixes of π 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 $\max\{|X|\} + t \geq s$

π : #0-#0-##

101h111h11

#0-#0-##

#0-#0-#

#0-#0-

#0-#0

#0-#

#0-

#0

#

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

Subset seed automaton

state definition

State definition :

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

- $X \subseteq R_\pi$: a subset of the prefixes of π 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 $\max\{X\} + t \geq s$

π : #0-#0-##

101h111h11

#0-#0-##

#0-#0-#

#0-#0-

#0-#0

#0-#

#0-

#0

#

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

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

Subset seed automaton

state definition

State definition :

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

- $X \subseteq R_\pi$: a subset of the prefixes of π 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 $\max\{X\} + t \geq s$

π : #0-#0-##

101h111h11

#0-#0-##

#0-#0-#

#0-#0-##

#0-#0

#0-#

#0-#0

#0-#

#

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

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

Subset seed automaton

transition function

Transition function :

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

if $a = 1$ **then**

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

else

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

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q, a \in \mathcal{A}$) 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 $1^t \cdot a$

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q$, $a \in \mathcal{A}$) 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 $1^t \cdot a$

$\pi : \#0\text{-}\#0\text{-}\#\#$

101h111h

#0-#0-##

#0-#0-#

#0-#0-

#0-#0

#0-#

#0-

#0

#

$$q : \begin{cases} X & = \{2, 3, 6\} \\ t & = 0 \end{cases}$$

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q$, $a \in \mathcal{A}$) 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 $1^t \cdot a$

$\pi : \#0\text{-}\#0\text{-}\#\#$

101h111h1

#0-#0-##

#0-#0-#

#0-#0-

#0-#0

#0-#

#0-

#0

#

$$q : \begin{cases} X & = \{2, 3, 6\} \\ t & = 0 \end{cases}$$

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q$, $a \in \mathcal{A}$) 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 $1^t \cdot a$

$\pi : \#0\text{-}\#0\text{-}\#\#$

101h111h1

#0-#0-##

#0-#0-#

#0-#0-#

#0-#0

#0-#

#0-#

#0-

#

$$q : \begin{cases} X & = \{2, 3, 6\} \\ t & = 0 \end{cases}$$

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q, a \in \mathcal{A}$) 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 $1^t \cdot a$

$\pi : \#0\text{-}\#0\text{-}\#\#$

101h111h1

#0-#0-##

#0-#0-#

#0-#0-#

#0-#0

#0-#

#0-#

#0-

#

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

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q, a \in \mathcal{A}$) 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 $1^t \cdot a$

$\pi : \#0\text{-}\#0\text{-}\#\#$

101h111h1h

#0-#0-##

#0-#0-#

#0-#0-#

#0-#0

#0-#

#0-#

#0-

#

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

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q, a \in \mathcal{A}$) 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 $1^t \cdot a$

$\pi : \#0\text{-}\#0\text{-}\#\#$

101h111h1h

#0-#0-##

#0-#0-#

#0-#0-##

#0-#0

#0-#

#0-#0

#0-#

#

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

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q$, $a \in \mathcal{A}$) 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 $1^t \cdot a$

$\pi : \#0\text{-}\#0\text{-}\#\#$

101h111h1h

#0-#0-##

#0-#0-#

#0-#0-##

#0-#0

#0-#

#0-#0

#0-#

#

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

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q$, $a \in \mathcal{A}$) 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 $1^t \cdot a$

$\pi : \#0\text{-}\#0\text{-}\#\#$

101h111h1h

#0-#0-##

#0-#0-#

#0-#0-##

#0-#0

#0-#

#0-#0

#0-#

#

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

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q$, $a \in \mathcal{A}$) 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 $1^t \cdot a$

$\pi : \#0\text{-}\#0\text{-}\#\#$

101h111h1h

#0-#0-##

#0-#0-#

#0-#0-##

#0-#0

#0-#

#0-#0

#0-#

#

#0

#

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

Subset seed automaton

transition function

Transition function :

$\psi(q, a)$ ($q : \langle X, t \rangle \in Q$, $a \in \mathcal{A}$) 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 $1^t \cdot a$

$\pi : \#0\text{-}\#0\text{-}\#\#$

101h111h1h

#0-#0-##

#0-#0-#

#0-#0-##

#0-#0

#0-#

#0-#0

#0-#

#

#0

#

$$q : \begin{cases} Y & = & \{2, 5\} \\ t & = & 0 \end{cases}$$

Subset seed automaton

automaton construction

Automaton construction

- incremental construction of states (breadth-first)

Subset seed automaton

automaton construction

Automaton construction

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

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 (\approx Aho-Corasick) :

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

- **Fail**⁻¹ :

$$RevMaxFail(\langle X', t \rangle) = \text{last created state } \langle X, t \rangle \text{ with } X' = X \setminus \max\{X\}$$

Subset seed automaton

reachable states computed constant time

$$q : \begin{cases} X & = & \{2, 3, 6\} \\ t & = & 1 \end{cases} \xrightarrow{a} q_r : \begin{cases} X & = & \{2, 5, 8\} \\ t & = & 0 \end{cases}$$

Subset seed automaton

reachable states computed constant time

$$\begin{array}{ccc} q : \begin{cases} X = \{2, 3, 6\} \\ t = 1 \end{cases} & \xrightarrow{a} & q_r : \begin{cases} X = \{2, 5, 8\} \\ t = 0 \end{cases} \\ \downarrow \text{Fail} & & \uparrow \text{RevMaxFail} \\ q' : \begin{cases} X = \{2, 3\} \\ t = 1 \end{cases} & \xrightarrow{a} & q'_r : \begin{cases} X = \{2, 5\} \\ t = 0 \end{cases} \end{array}$$

Subset seed automaton

automaton size

- The number of states is $\mathcal{O}(w2^{s-w})$, where s the span of π , and w the number of # in π .

Subset seed automaton

automaton size

- The number of states is $\mathcal{O}(w2^{s-w})$, where s the span of π , and w the number of # in π .
- The base of the exponent does not depend on \mathcal{A} .

Subset seed automaton

automaton size

- The number of states is $\mathcal{O}(w2^{s-w})$, where s the span of π , and w the number of # in π .
- 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 $S_\pi \rightarrow AC_\pi$.

Subset seed automaton

automaton size

- The number of states is $\mathcal{O}(w2^{s-w})$, where s the span of π , and w the number of $\#$ in π .
- 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).

Subset seed automaton

automaton size

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

Subset seed automaton

automaton size

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

Subset seed automaton

Average size of S_π

- 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_π

| $ \mathcal{A} = 2$ | | Aho-Corasick | | S_π | | Minimized | $ \mathcal{A} = 3$ | | Aho-Corasick | | S_π | | Minimized |
|---------------------|--|--------------|-------|---------|-------|-----------|---------------------|--------|--------------|--------|---------|-------|-----------|
| w | | avg. | ratio | avg. | ratio | avg. | w | avg. | ratio | avg. | ratio | avg. | |
| 9 | | 130.98 | 2.46 | 67.03 | 1.260 | 53.18 | 9 | 1103.5 | 16.46 | 86.71 | 1.293 | 67.05 | |
| 10 | | 140.28 | 2.51 | 70.27 | 1.255 | 55.98 | 10 | 1187.7 | 16.91 | 90.67 | 1.291 | 70.25 | |
| 11 | | 150.16 | 2.55 | 73.99 | 1.254 | 58.99 | 11 | 1265.3 | 17.18 | 95.05 | 1.291 | 73.65 | |
| 12 | | 159.26 | 2.57 | 77.39 | 1.248 | 62.00 | 12 | 1346.1 | 17.50 | 98.99 | 1.287 | 76.90 | |
| 13 | | 168.19 | 2.59 | 80.92 | 1.246 | 64.92 | 13 | 1419.3 | 17.67 | 103.10 | 1.284 | 80.31 | |

Table: Average number of states of Aho-Corasick, S_π and minimal automaton

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

Table: Average number of states of Aho-Corasick, S_π and minimized automata for the case of two seeds

Subset seed automaton

Average size of S_π

| $ \mathcal{A} = 2$ | Aho-Corasick | | S_π | | Minimized avg. | $ \mathcal{A} = 3$ | Aho-Corasick | | S_π | | Minimized avg. |
|---------------------|--------------|------|---------|-------|-------------------|---------------------|--------------|-------|---------|-------|-------------------|
| | w | avg. | ratio | avg. | | | ratio | w | avg. | ratio | |
| 9 | 130.98 | 2.46 | 67.03 | 1.260 | 53.18 | 9 | 1103.5 | 16.46 | 86.71 | 1.293 | 67.05 |
| 10 | 140.28 | 2.51 | 70.27 | 1.255 | 55.98 | 10 | 1187.7 | 16.91 | 90.67 | 1.291 | 70.25 |
| 11 | 150.16 | 2.55 | 73.99 | 1.254 | 58.99 | 11 | 1265.3 | 17.18 | 95.05 | 1.291 | 73.65 |
| 12 | 159.26 | 2.57 | 77.39 | 1.248 | 62.00 | 12 | 1346.1 | 17.50 | 98.99 | 1.287 | 76.90 |
| 13 | 168.19 | 2.59 | 80.92 | 1.246 | 64.92 | 13 | 1419.3 | 17.67 | 103.10 | 1.284 | 80.31 |

Table: Average number of states of Aho-Corasick, S_π and minimal automaton

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

Table: Average number of states of Aho-Corasick, S_π and minimized 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 ...

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

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] GGGNNNNAN [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

a very small example ...

seed pattern :

`#-@#`

words matched :

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

$X = \emptyset$
 $t = 0$



Subset seed automaton

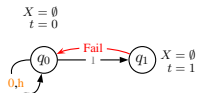
a very small example ...

seed pattern :

#-@#

words matched :

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



Subset seed automaton

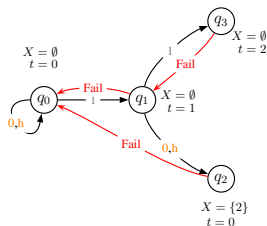
a very small example ...

seed pattern :

#-@#

words matched :

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



Subset seed automaton

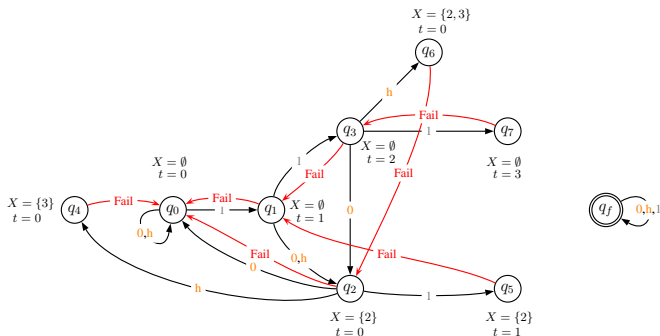
a very small example ...

seed pattern :

#-@#

words matched :

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



Subset seed automaton

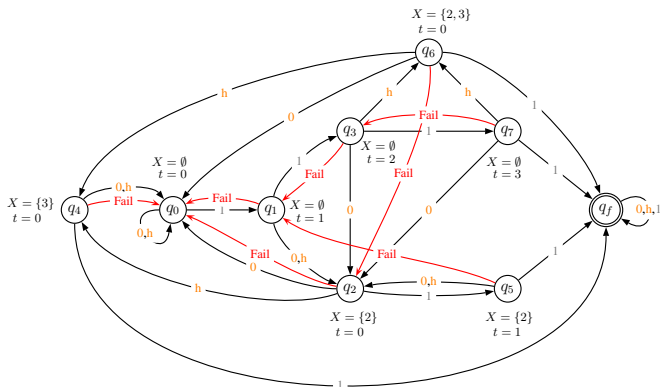
a very small example ...

seed pattern :

#-@#

words matched :

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



Subset seed automaton

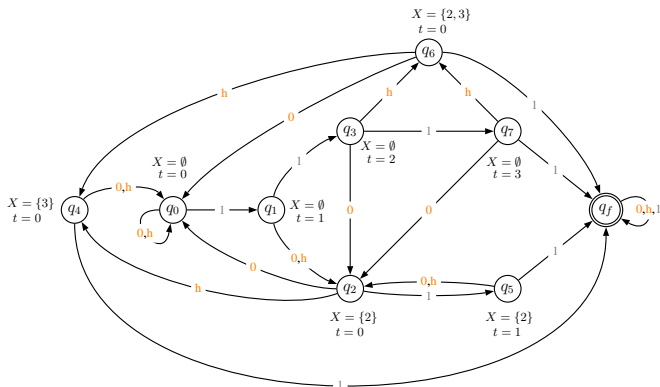
a very small example ...

seed pattern :

#-@#

words matched :

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



Subset seed automaton

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}

$$\begin{aligned} X &= \emptyset \\ t &= 0 \end{aligned}$$



Subset seed automaton

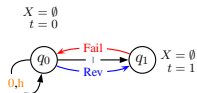
a very small example ...

seed pattern :

#-@#

words matched :

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



Subset seed automaton

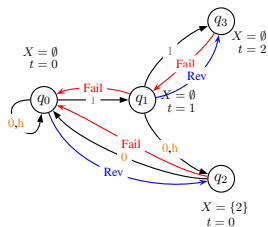
a very small example ...

seed pattern :

#-@#

words matched :

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



Subset seed automaton

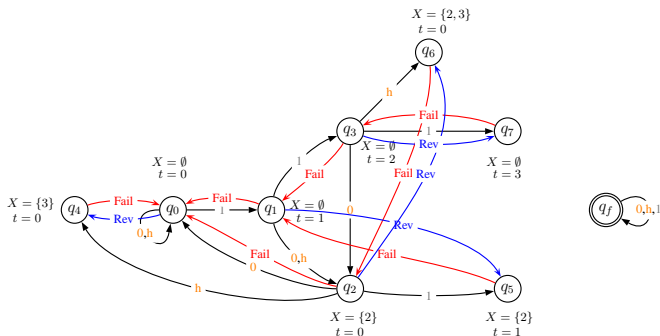
a very small example ...

seed pattern :

#-@#

words matched :

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



Subset seed automaton

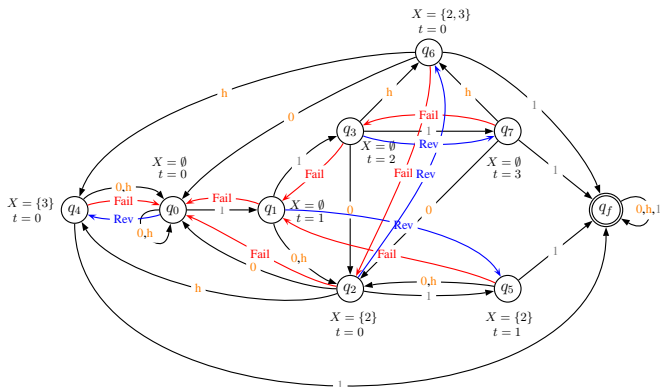
a very small example ...

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}

