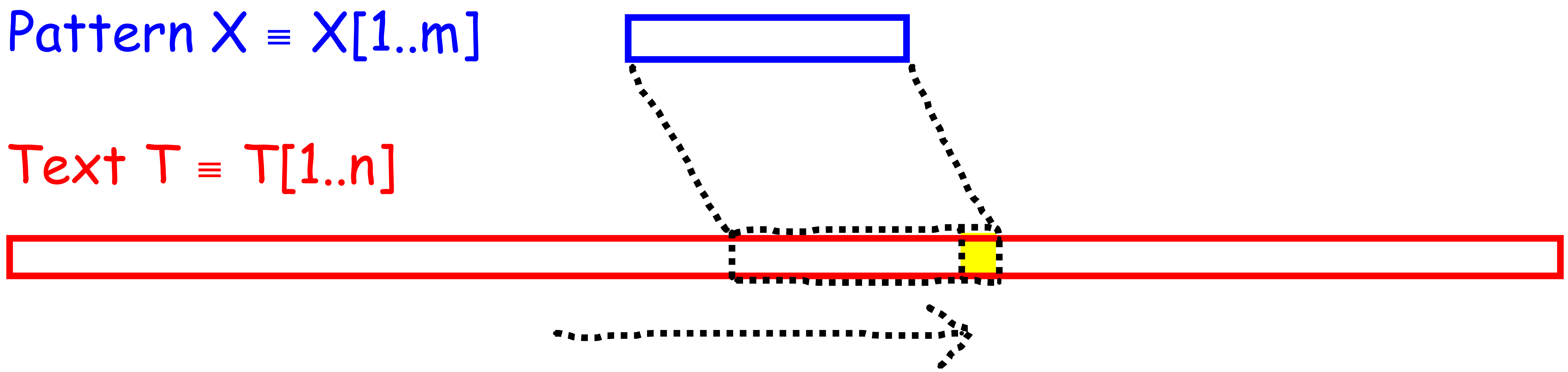
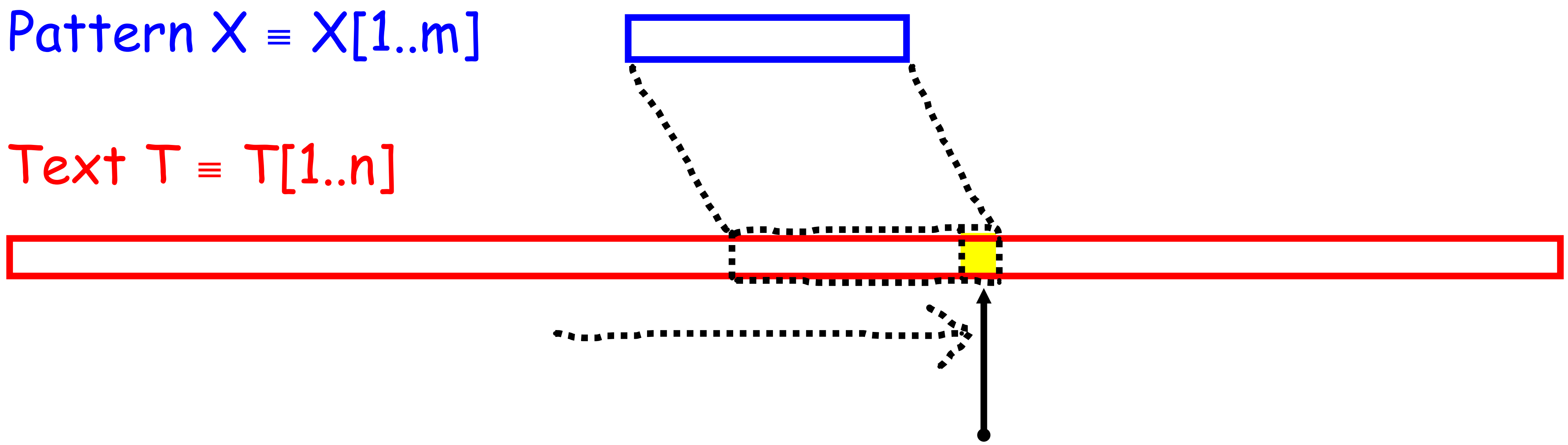
Simple Real-Time Constant-Space String Matching

Dany Breslauer, Roberto Grossi and Filippo Mignosi

Real-time string matching

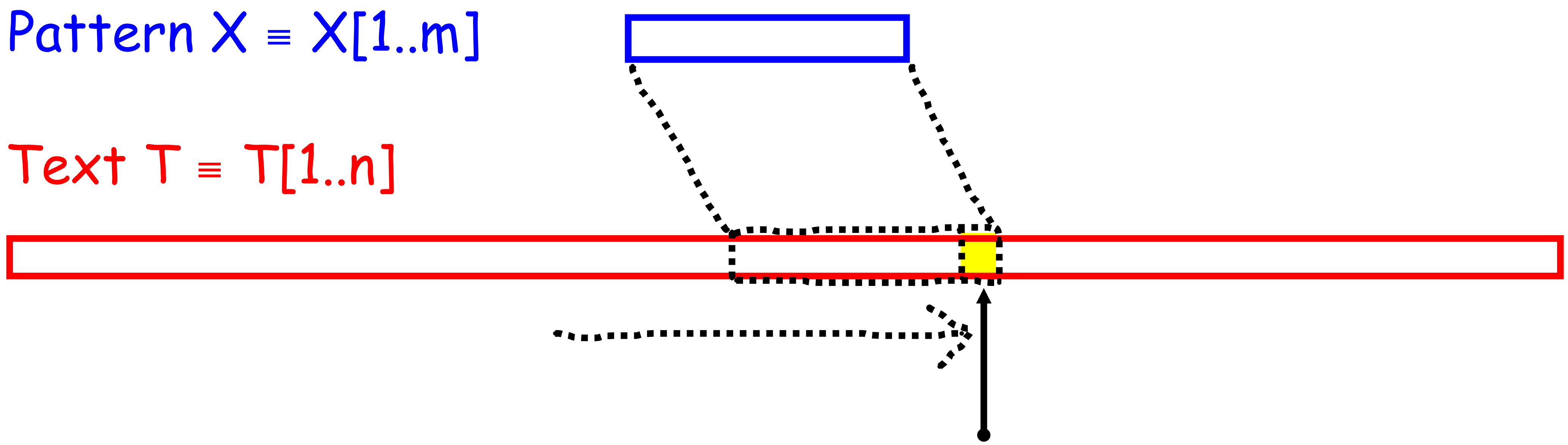


Real-time string matching



O(1) worst-case time to answer after reading the text symbol

Real-time string matching



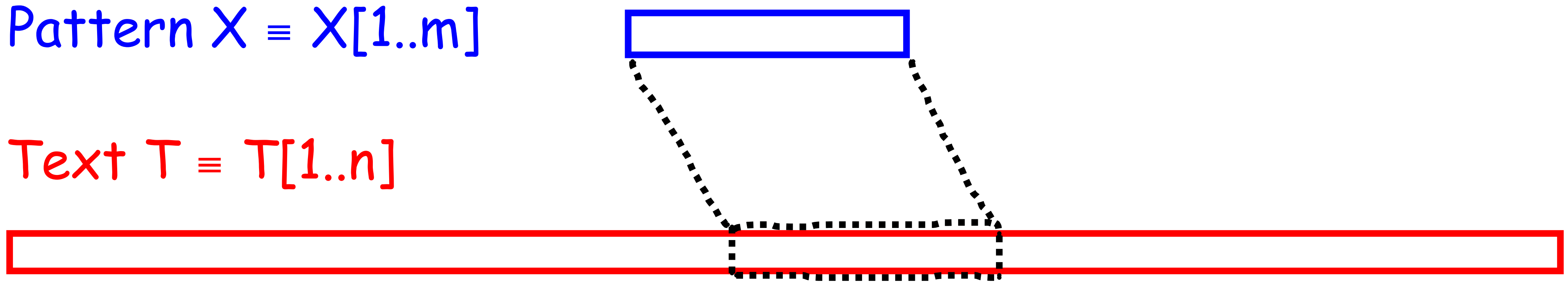
O(1) worst-case time to answer after reading the text symbol

Different from real-time streaming s.m., where X and T cannot be entirely stored!

Constant-space string matching

Pattern X ≡ X[1..m]

Text T ≡ T[1..n]



(a.k.a. in-place)

O(1) working space apart from that required by X and T

O(log n) bits

We propose a simple way to combine the two features

- Take a simple version of the constant-space Crochemore-Perrin (CP) algorithm

We propose a simple way to combine the two features

- Take a simple version of the constant-space Crochemore-Perrin (CP) algorithm

- Make CP also real-time by running **two** instances simultaneously

Some related work

- Galil '81: real-time string matching
- Galil, Seiferas '83: constant space
- Karp, Rabin '87: randomized constant space real-time
- Crochemore, Perrin '91: constant space
- Gasieniec, Plandowski, Rytter '95: constant space
- Gasienec, Kolpakov '04: real-time + sublinear space (extends GPR'95)
- • • more papers [Crochemore, Rytter '91,'95] [Crochemore '92] [...]
- Porat, Porat '09: randomized streaming, O(log m) space, no real-time
- Breslauer, Galil '10: randomized real-time streaming, O(log m) space

Our result

- Real-time constant-space string matching
  - O(1) words in addition to those for read-only X and T
  - O(1) worst-case time to answer after each text symbol

Our result

- Real-time constant-space string matching deterministic
  - O(1) words in addition to those for read-only X and T
  - O(1) worst-case time to answer after each text symbol

Not to be confused with

- Real-time streaming string matching Montecarlo
  - O(log m) memory words (X and T cannot be kept)
  - O(1) worst-case time to answer after each text symbol

We propose a simple way to combine the two features

- Take a simple version of the constant-space Crochemore-Perrin (CP) algorithm

- Make CP also real-time by running two instances simultaneously

Simple version of the Crochemore-Perrin (CP) algorithm

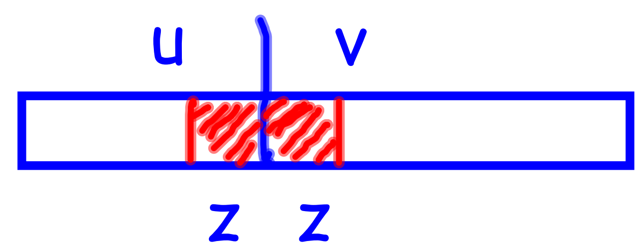
Consider a non-empty prefix-suffix factorization X = u v

The local period is the shortest z such that

z is suffix of u or vice versa

and

z is a prefix of v or vice versa



$\mu(u,v) \equiv$ length $|z|$ of the local period

Simple version of the Crochemore-Perrin (CP) algorithm

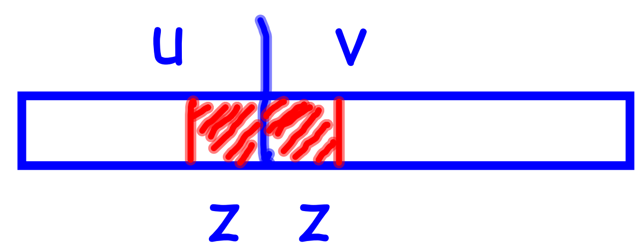
Consider a non-empty prefix-suffix factorization X = u v

The local period is the shortest z such that

z is suffix of u or vice versa

and

z is a prefix of v or vice versa



μ(u,v) ≡ length |z| of the local period

Example: X = abaaaba

X = u v

a baaaba	ab aaaba	aba aaba
ba ba	aaab aaab	a a
z		

Simple version of the Crochemore-Perrin (CP) algorithm

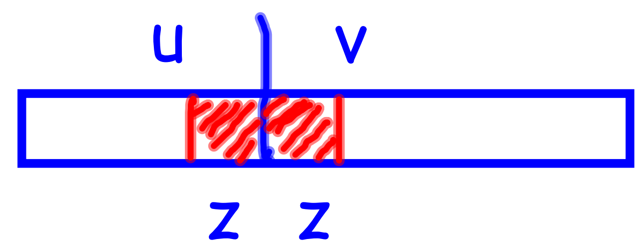
Consider a non-empty prefix-suffix factorization X = u v

The local period is the shortest z such that

z is suffix of u or vice versa

and

z is a prefix of v or vice versa



μ(u,v) ≡ length |z| of the local period

Example: X = abaaaba

	X = u v	
a baaaba	ab aaaba	aba aaba
ba ba	aaab aaab	a a
	z	

Simple version of the Crochemore-Perrin (CP) algorithm

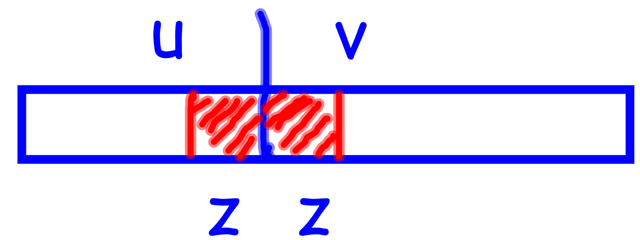
Consider a non-empty prefix-suffix factorization X = u v

The local period is the shortest z such that

z is suffix of u or vice versa

and

z is a prefix of v or vice versa



$\mu(u,v) \equiv$ length $|z|$ of the local period

Example: X = abaaaba

		X = u v
a baaaba	ab aaaba	aba aaba
ba ba	aaab aaab	a a
		z

Simple version of the Crochemore-Perrin (CP) algorithm

Consider a non-empty prefix-suffix factorization $X = u\, v$

The local period is the shortest z such that

z is suffix of u or vice versa

and

z is a prefix of v or vice versa

$\mu(u,v) \equiv$ length of the local period

Critical factorization if $\mu(u,v) = \pi(X)$ [len. of the period of X]

Example:

X = u v

a baaaba	ab aaaba	aba aaba
ba ba	aaab aaab	a a

z

Example:

X = u v

a baaaba

ab aaaba

aba aaba

ba ba

aaab aaab

a a

z

Example:

```
 a  baaaba        X = u  v            aba  aaba
ba  ba              ab  aaaba           a  a
                  aaab  aaab
                         z
```

critical!

the period is "abaa"

Example:

a baaaba

ba ba

ab aaaba

aaab aaab

X = u v

aba aaba

a a

z

Example:

a baaaba   ab aaaba   aba aaba

ba ba   aaab aaab   a a

Critical Factorization Theorem (Cesari and Vincent):

Among $\pi(X) - 1$ consecutive factorizations:

at least one is a critical factorization

Example:

```
 a  baaaba        ab  aaaba        aba  aaba
ba  ba          aaab  aaab           a  a
```

Critical Factorization Theorem (Cesari and Vincent):

Among $\pi(X) - 1$ consecutive factorizations:

at least one is a critical factorization

There always exists a critical factorization $X = u\ v$ such that $|u| < \pi(X)$

Crochemore-Perrin (CP) Algorithm:

Take such a critical factorization of the pattern $X = u\ v$

Crochemore-Perrin (CP) Algorithm:

Take such a critical factorization of the pattern $X = u\,v$

Forward scan: match v left-to-right with the current aligned portion of the text

Crochemore-Perrin (CP) Algorithm:

Take such a critical factorization of the pattern $X = u\,v$

Forward scan: match v left-to-right with the current aligned portion of the text

Back fill: match u left-to-right with the current aligned portion of the text [originally right-to-left]

Crochemore-Perrin (CP) Algorithm:

Take such a critical factorization of the pattern $X = u\ v$

Forward scan: match v left-to-right with the current aligned portion of the text

Back fill: match u left-to-right with the current aligned portion of the text [originally right-to-left]

How to handle mismatches?

We propose a simple way to combine the two features

- Take a simple version of the constant-space Crochemore-Perrin (CP) algorithm

- Make CP also real-time by running two instances simultaneously

Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

X = ab aaaba critical factorization

abaaaba

abaabaaabaa

Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

X = ab aaaba critical factorization

abaaaba

ababaaabaa

Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

X = ab aaaba critical factorization

abaaaba

abaabaaabaa

Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

X = ab aaaba critical factorization

z

abaaaba

≠

abaabaaabaa

↑

mismatch

Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

X = ab aaaba critical factorization

z

abaaaba

abaabaaabaa

|z|+1

abaaaba

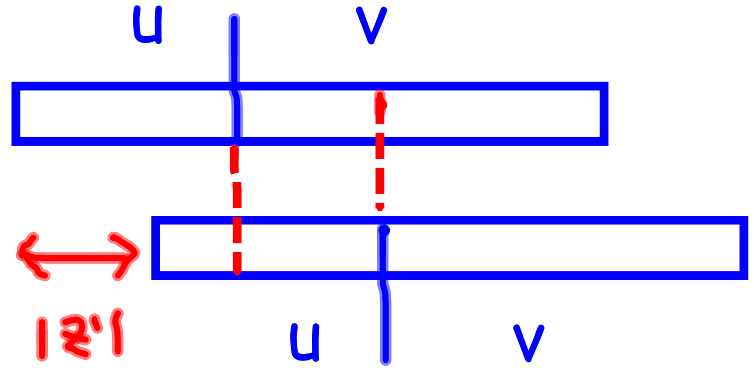
abaabaaabaa

shift by |z|+1 positions

(and charge the O(|z|+1) cost to the symbols in z in real time)

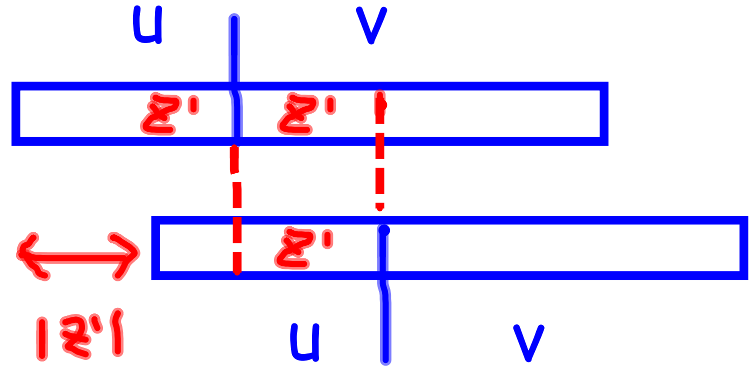
By contradiction, suppose there is a valid shift that is shorter...

... recall that |u| < π(X), the length of the period



By contradiction, suppose there is a valid shift that is shorter...

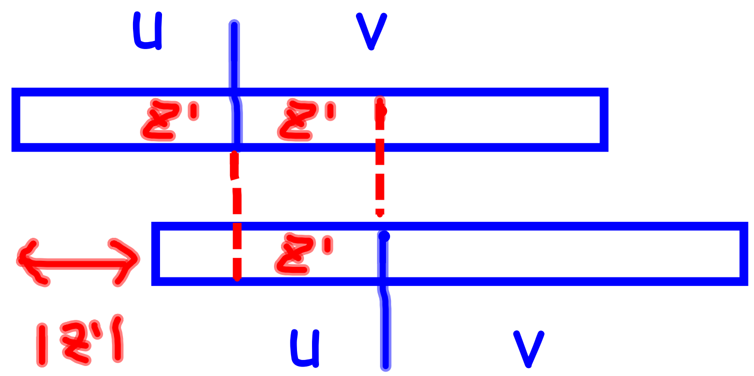
... recall that $|u| < \pi(X)$, the length of the period



$|z'| < \pi(X)$

By contradiction, suppose there is a valid shift that is shorter...

... recall that $|u| < \pi(X)$, the length of the period

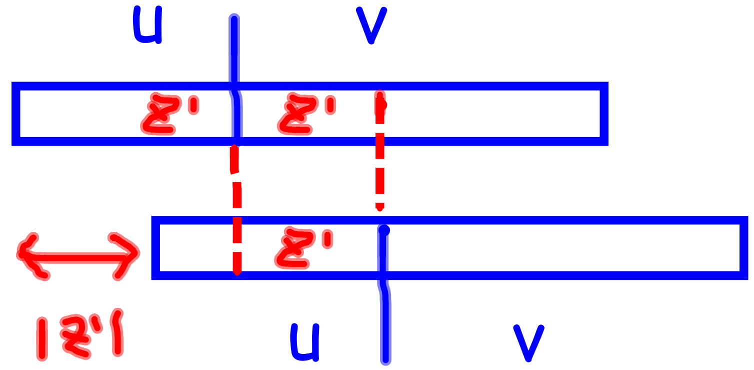


$|z'| < \pi(X)$

Contradiction: a local period at u v that is shorter than $\pi(X)$!!

By contradiction, suppose there is a valid shift that is shorter...

... recall that $|u| < \pi(X)$, the length of the period



$|z'| < \pi(X)$

Contradiction: a local period at u v that is shorter than $\pi(X)$!!

It follows from the Crochemore-Perrin result [other case $|z'| \geq \pi(X)$ not displayed: periodicity rules out occurrences]

Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

Output an occurrence when the forward scan terminates (and interrupt the back fill if needed)

Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

Output an occurrence when the forward scan terminates (and interrupt the back fill if needed)

Let z be the matched prefix of v, where $X = u\,v$ is c.f.:

- if $z \neq v \Rightarrow$ shift by $|z|+1$ positions and reset z = empty
- if $z = v \Rightarrow$ shift by $\pi(X)$ positions and update z

Basic Real-Time Algorithm

Interleave $O(1)$ comparisons from the forward scan with $O(1)$ comparisons from the back fill

Output an occurrence when the forward scan terminates (and interrupt the back fill if needed)

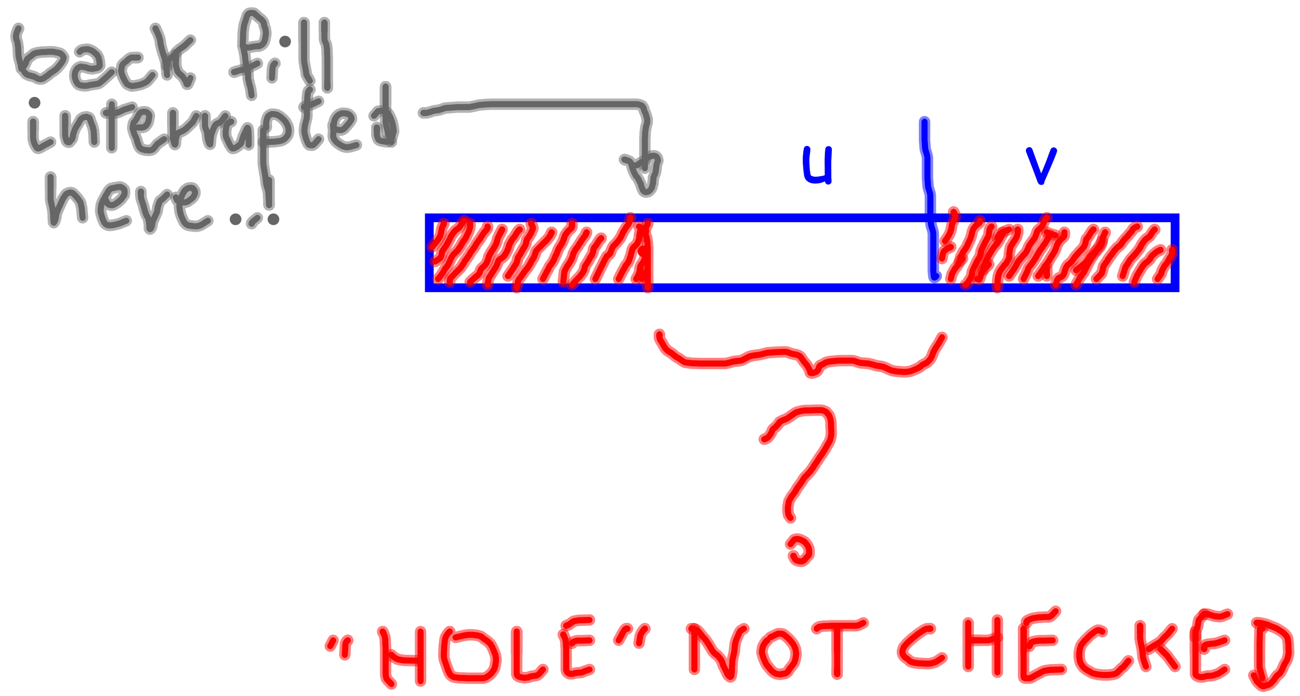
Let z be the matched prefix of v, where $X = u\,v$ is c.f.:

- if $z \neq v \Rightarrow$ shift by $|z|+1$ positions and reset z = empty
- if $z = v \Rightarrow$ shift by $\pi(X)$ positions and update z

Total cost is $O(1)$ worst-case per symbol:
the algorithm is real-time

Q: What if $|u| > |v|$?

Q: What if |u| > |v|?



Real-Time Variation of CP

Consider a 3-way non-empty factorizaton $X = u\ v\ w$ such that

$X = (uv)\ w$ is a critical factorization with $|uv| \leq |w|$

OR

$X = (uv)\ w$ is a critical factorization, and
$X' = u\ (vv')$ is a critical factorization for a prefix X' of X
with $|u| \leq |vv'|$

Real-Time Variation of CP

Consider a 3-way non-empty factorizaton $X = u\ v\ w$ such that

$X = (uv)\ w$ is a critical factorization with $|uv| \leq |w|$

HAPPY!

OR

$X = (uv)\ w$ is a critical factorization, and
$X' = u\ (vv')$ is a critical factorization for a prefix X' of X
with $|u| \leq |vv'|$

Real-Time Variation of CP

Consider a 3-way non-empty factorizaton X = u v w such that

X = (uv) w is a critical factorization with |uv| ≤ |w|

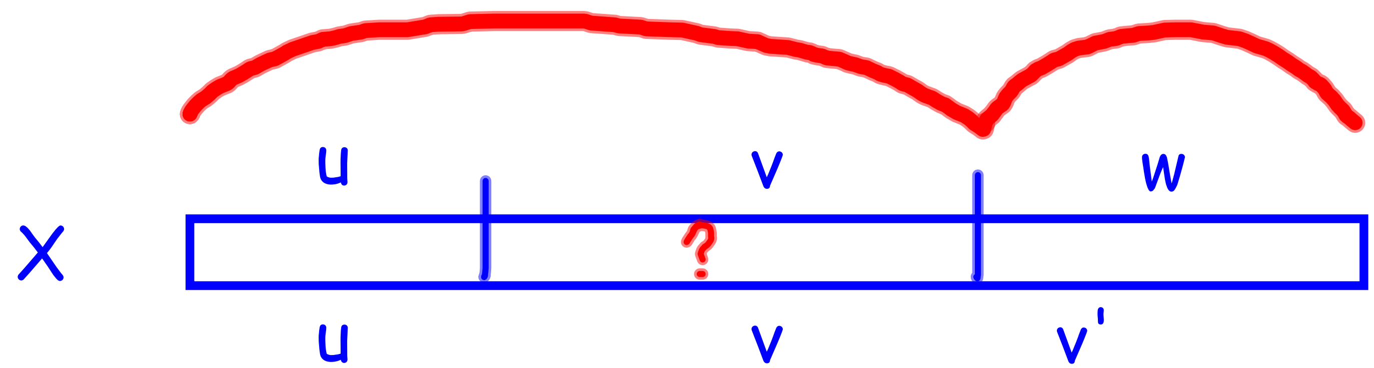
OR

X = (uv) w is a critical factorization, and
X' = u (vv') is a critical factorization for a prefix X' of X
with |u| ≤ |vv'|

we focus on this...

Real-Time Variation of CP

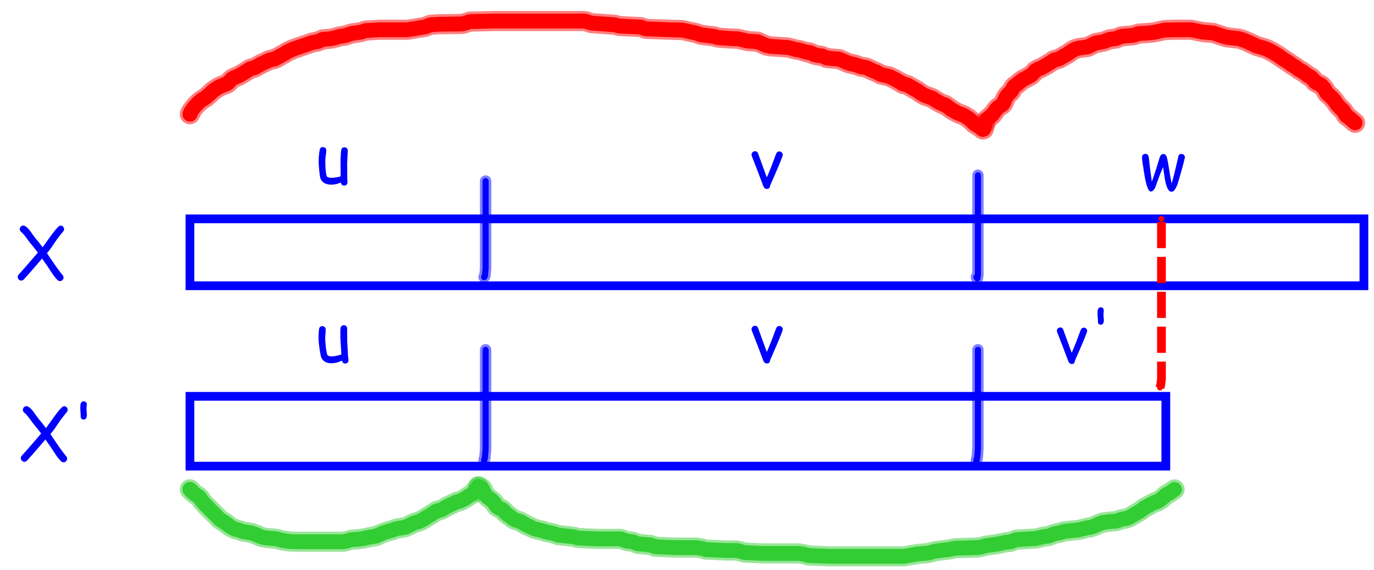
$X = (uv)\,w$ is a critical factorization, and



Recall we may leave a "hole" to the left of w:
this hole has to be covered by X'...

Real-Time Variation of CP

$X = (uv)\, w$ is a critical factorization, and
$X' = u\,(vv')$ is a critical factorization for a prefix X' of X
with $|u| \le |vv'|$



Note that X' is entirely matched since $|u| \le |vv'|$

Real-Time Variation of the CP Algorithm

Interleave O(1) steps of two instances of the Basic Real-Time Algorithms, one looking for X and the other for X', aligned with |X|-|X'| positions apart.

Real-Time Variation of the CP Algorithm

Interleave O(1) steps of two instances of the Basic Real-Time Algorithms, one looking for X and the other for X', aligned with $|X|-|X'|$ positions apart.

Total cost is O(1) worst-case per symbol: the algorithm is real-time and reports correctly all the occurrences

Simple pseudocode

Pattern preprocessing

GOAL:

Find the desired 3-way non-empty factorizaton $X = u\ v\ w$
and the length of the periods of X and X'

Pattern preprocessing

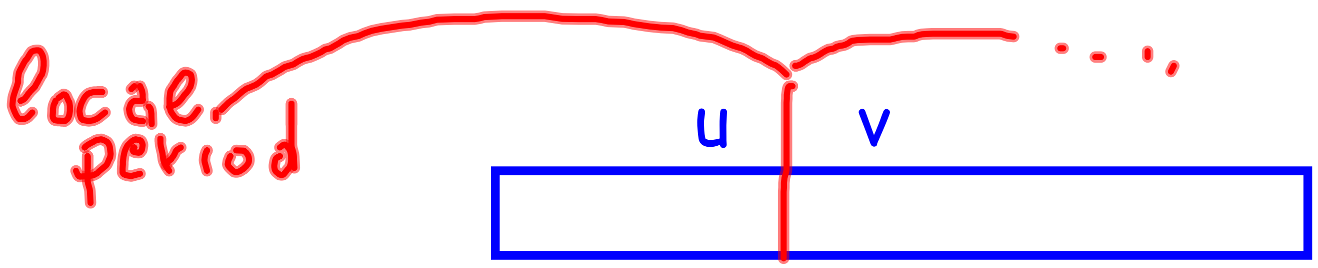
GOAL:

Find the desired 3-way non-empty factorizaton $X = u\ v\ w$
and the length of the periods of X and X'

We focus on this...

Some more definitions...

A factorization u v is left-external if $|u| \leq \mu(u,v)$ for non-empty u, v



Define L(X) = { u v : X = u v is left-external }

L(X) non-empty because of the Critical Factorization Theorem

Pattern preprocessing

Let $X = u_1 w$ be the first critical factorization in $L(X)$

HINT: use CP preprocessing on the prefixes of X

Lemma: $u\, v \in L(X) \Rightarrow$ prefix $X' = u'\, v'$ s.t. $\mu(u', v') = \mu(u, v)$

Pattern preprocessing

Let $X = u_1\, w$ be the first critical factorization in $L(X)$

HINT: use CP preprocessing on the prefixes of X

Lemma: $u\ v \in L(X) \Rightarrow$ prefix $X' = u'\ v'$ s.t. $\mu(u',v') = \mu(u,v)$

Compute CP critical factorization for $u_1 = u\ v$

where $|u| \leq \mu(u,v)$

Pattern preprocessing

Let $X = u_1\, w$ be the first critical factorization in $L(X)$

HINT: use CP preprocessing on the prefixes of X

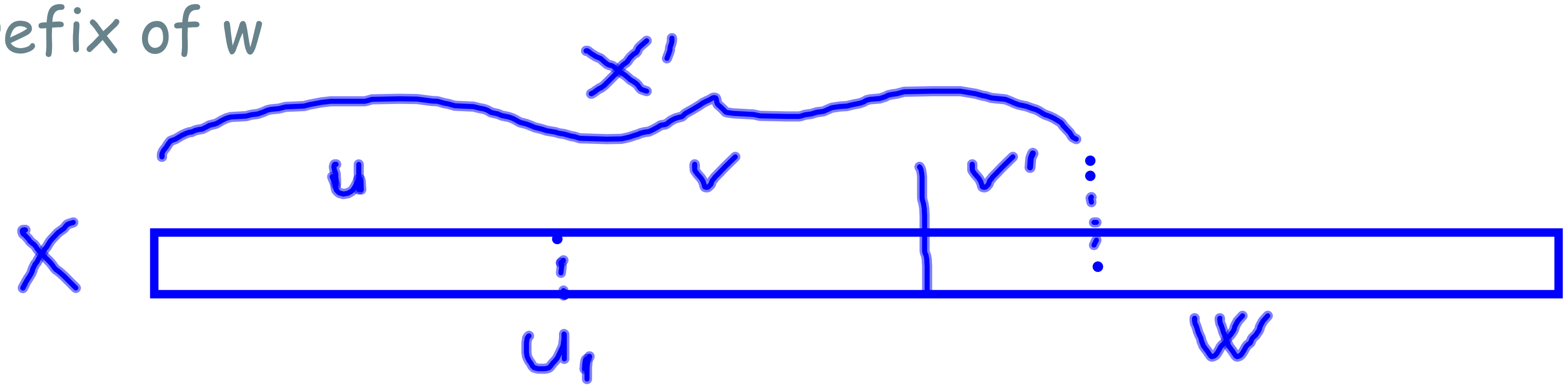
Lemma: $u\ v \in L(X) \Rightarrow$ prefix $X' = u'\ v'$ s.t. $\mu(u',v') = \mu(u,v)$

Compute CP critical factorization for $u_1 = u\ v$

where $|u| \leq \mu(u,v)$

Extend u_1 by periodicity $\mu(u,vw) < |vw|$: set $X' = u\ (vv')$

where v' prefix of w



Pattern preprocessing

Let $X = u_1\, w$ be the first critical factorization in $L(X)$

HINT: use CP preprocessing on the prefixes of X

Lemma: $u\, v \in L(X) \Rightarrow$ prefix $X' = u'\, v'$ s.t. $\mu(u', v') = \mu(u, v)$

Compute CP critical factorization for $u_1 = u\, v$

where $|u| \leq \mu(u, v)$

Extend u_1 by periodicity $\mu(u, vw) < |vw|$: set $X' = u\,(vv')$

where v' prefix of w

It is $|u| \leq \mu(u, v) \leq \mu(u, vv') = \mu(u, vw) \leq |vv'|$

Q.E.D.

Questions?