Complexity of Certain Query Answering on Hyperstreams

Momar Sakho[1] - Iovka Boneva[2] - Joachim Niehren[1]

1: Inria Lille - Nord Europe
2: Université de Lille

Processing streams with semi-structured data

Complex event processing [Mozafari 12, Luckham 98]

XML stream processing [Suciu 04, Olteanu 06, Kay 10, Koch 07, Gauwin 09, Sebastian15]

RDF streams [Shinavier 10]

Data-centric workflows [Abiteboul et al. 13]

From Streams to Hyperstreams

Pipelines of stream transformations

produce and consume multiple streams

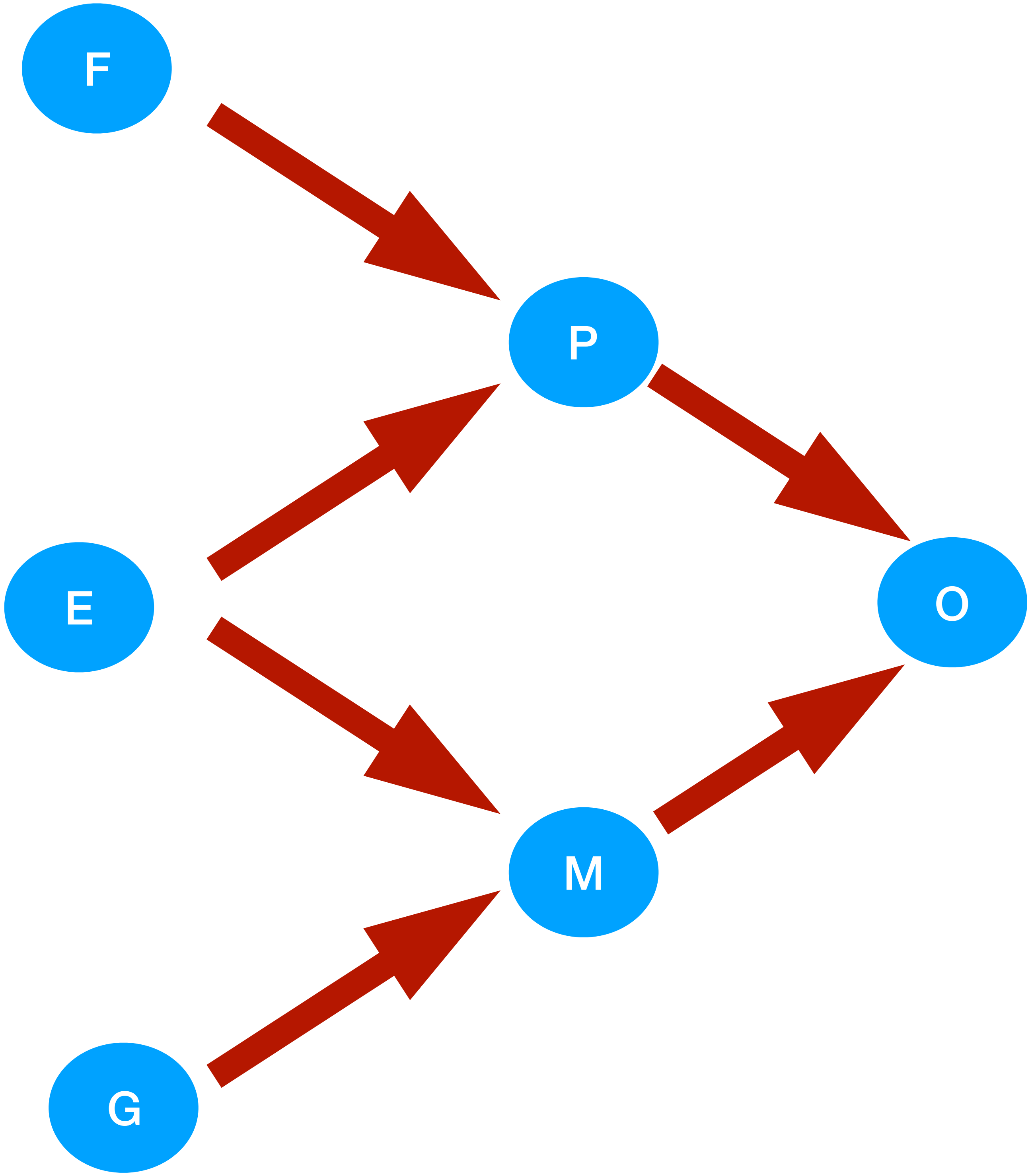
composition of transformations

sharing input streams

=> Streams with reference to others : Hyperstreams

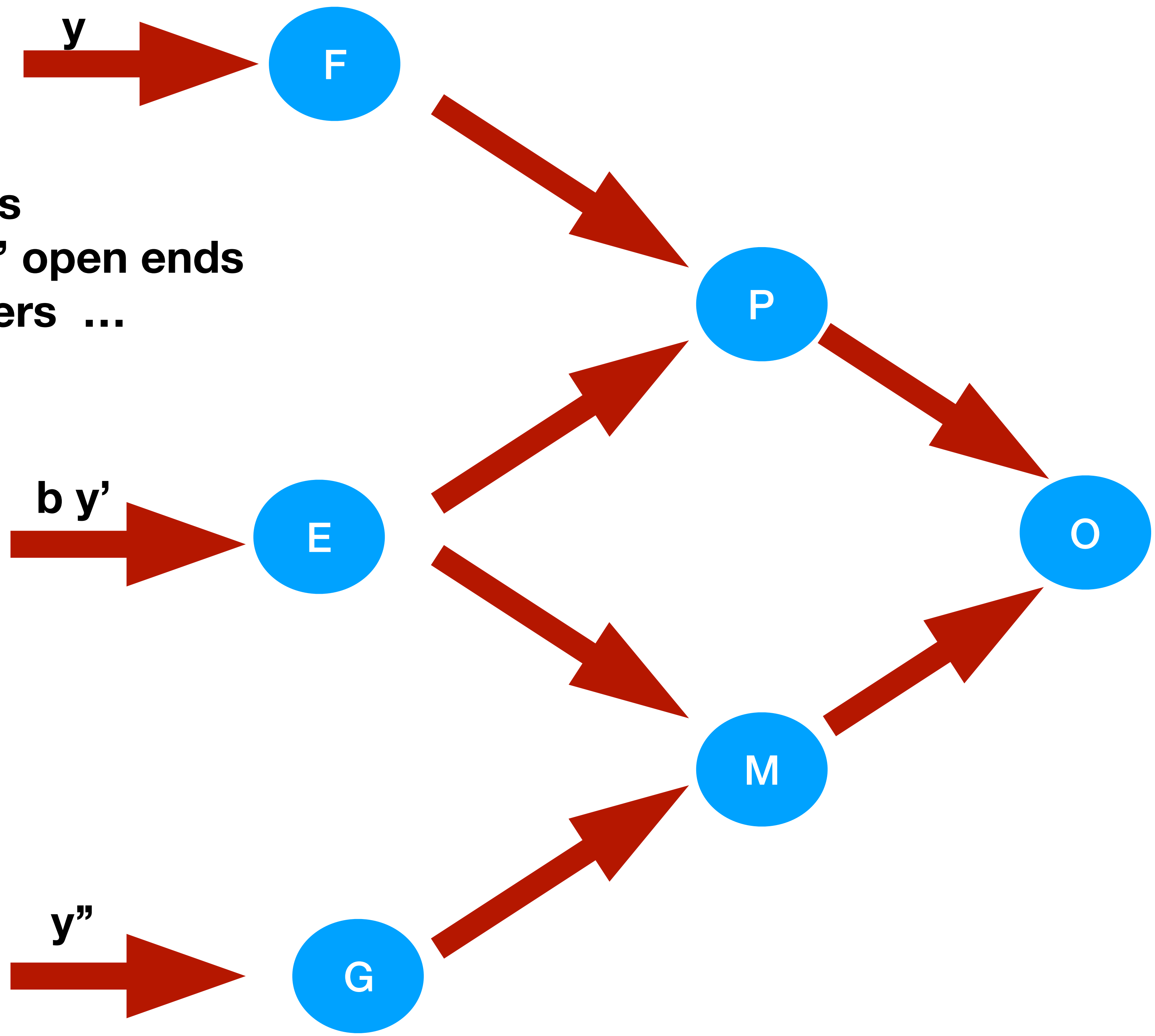
[Maneth & Seidl 15, Labath & Niehren 13]

Pipelines of streams

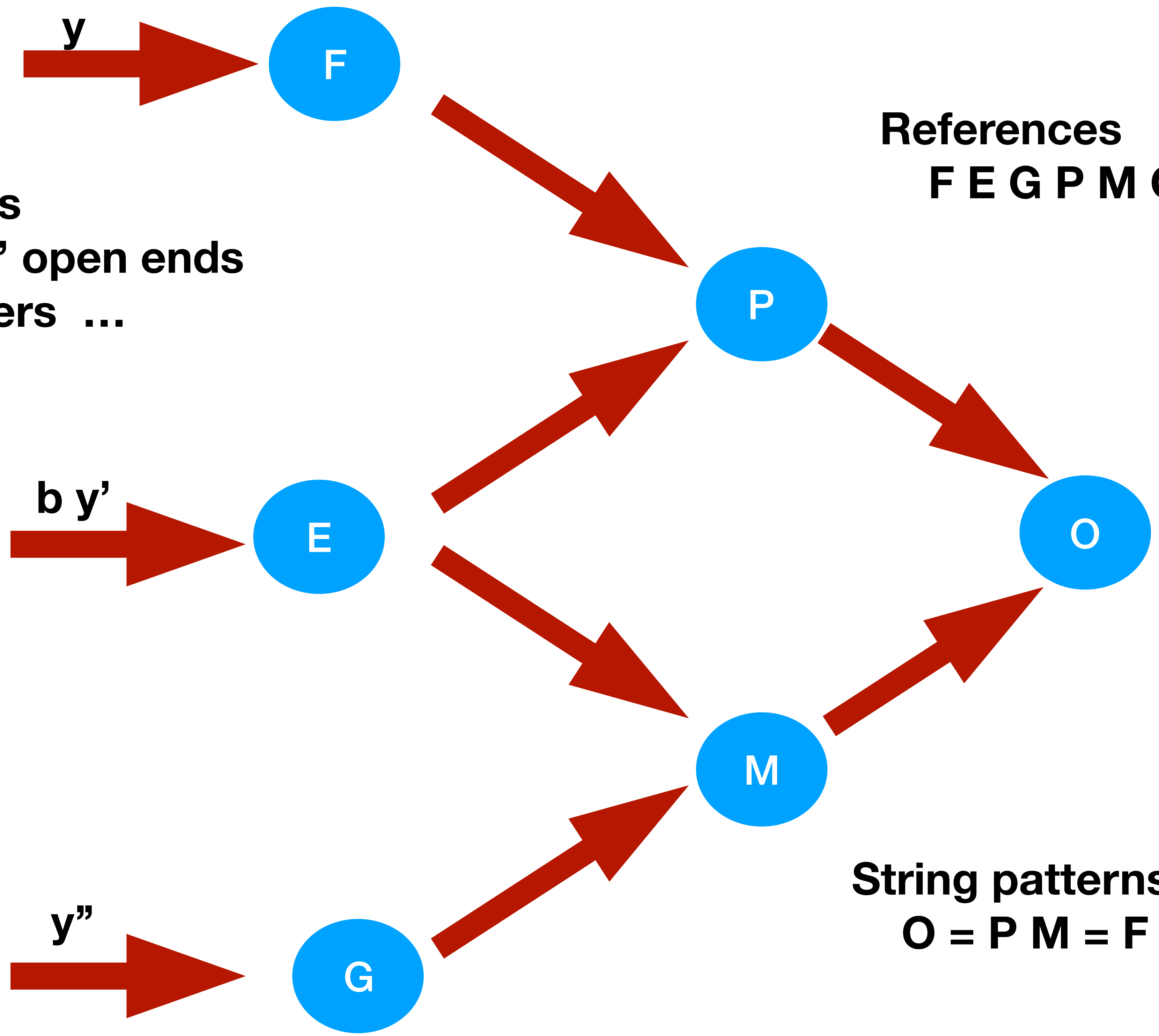


Pipelines of streams

Streams
y y’ y” open ends
b letters ...



Pipelines of streams



References
F E G P M O

Streams
y y’ y” open ends
b letters …

String patterns
O = P M = F E E G = y b y’ b y’ y”

Certain Query Answers

String patterns

O = y b y’ b y’ y”

b = book(title:”Tom Sawyer”,
author:”Mark Twain”)

Certain Query Answers

String patterns

O = y b y’ b y’ y”

b = book(title:”Tom Sawyer”,
author:”Mark Twain”)

Query 1

all the books of Mark Twain

Answers on O

first b certain

second b certain

…

Certain Query Answers

String patterns

O = y b y’ b y’ y”

b = book(title:”Tom Sawyer”,
author:”Mark Twain”)

Query 1

all the books of Mark Twain

Answers on O

first b certain

second b certain

…

Query 2

last book of Mark Twain

Answers on O

none certain

second b alive candidate

…

Non-answers on O

first b certain

…

Question

How difficult it is to decide given

- a query Q
- a hyperstream D
- a position π of pattern of D,

wether π is

- certain for selection by Q?
- certain for rejection by Q?

Restrictions

Hyperstreams

describe words (not trees, nor graphs)

no data values (finite alphabet)

Queries defined by finite automata

obtained from logical queries

motivated by XPath but on words

Known results for streams

	NFA queries	DFA queries
Selection	PSPACE-c	PTIME
Rejection	PTIME	PTIME

[Gauwin 09]

Contribution for hyperstreams

	NFA queries	DFA queries
Selection	PSPACE-c	PSPACE-c
Rejection	PSPACE-c	PSPACE-c

Contribution for hyperstreams

	NFA queries	DFA queries
Selection	PSPACE-c	PSPACE-c
Rejection	PSPACE-c	PSPACE-c

But not everything is lost…

Contributions for Linear Hyperstreams

	NFA queries	DFA queries
Selection	PSPACE-c	PTIME
Rejection	PTIME	PTIME

Queries on strings

Finite alphabet

$a, b, c \in \Sigma$

Strings

$w \in \Sigma^*$

Boolean queries

$Q \subseteq \Sigma^*$

Queries can be defined by NFAs or DFAs A

$Q = L(A)$

String patterns

String variables

$y \in \Upsilon$

String Patterns

$p \in (\Sigma \cup \Upsilon)^*$

Instance

$\mathrm{Inst}(p) = \{p\sigma \mid \sigma\colon \Upsilon \to \Sigma^*\}$

String patterns

String variables

$y \in Y$

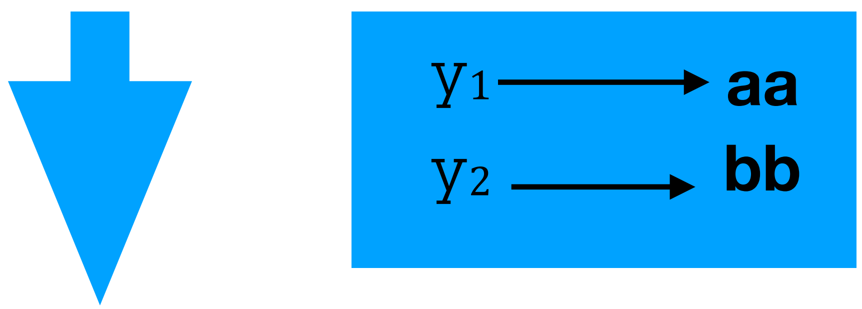
String Patterns

$p \in (\Sigma \cup Y)^*$

Instance

$\mathrm{Inst}(p) = \{p\sigma \mid \sigma: Y \to \Sigma^*\}$

$aby_1bay_2bby_1$



abaababbbbaa

String patterns

String variables

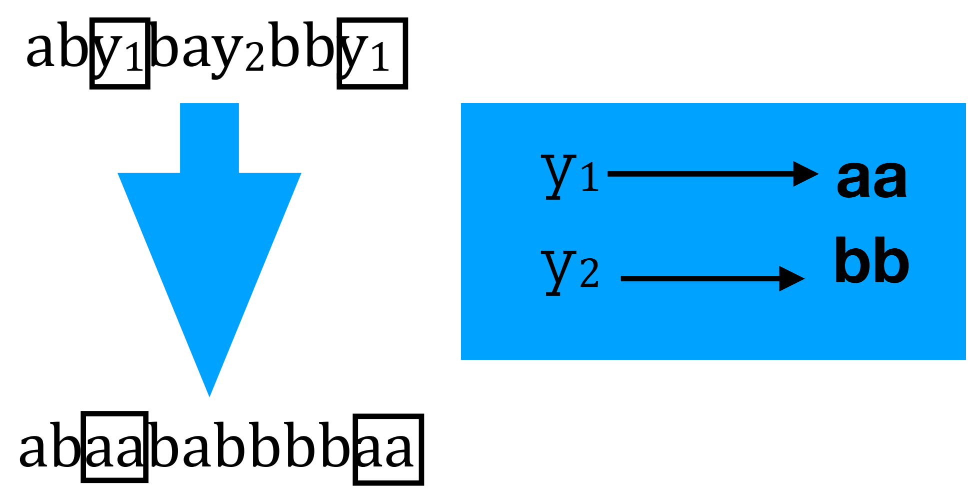
$y \in Y$

String Patterns

$p \in (\Sigma \cup Y)^*$

Instance

$\mathrm{Inst}(p) = \{p\sigma \mid \sigma\colon Y \to \Sigma^*\}$



String patterns

String variables

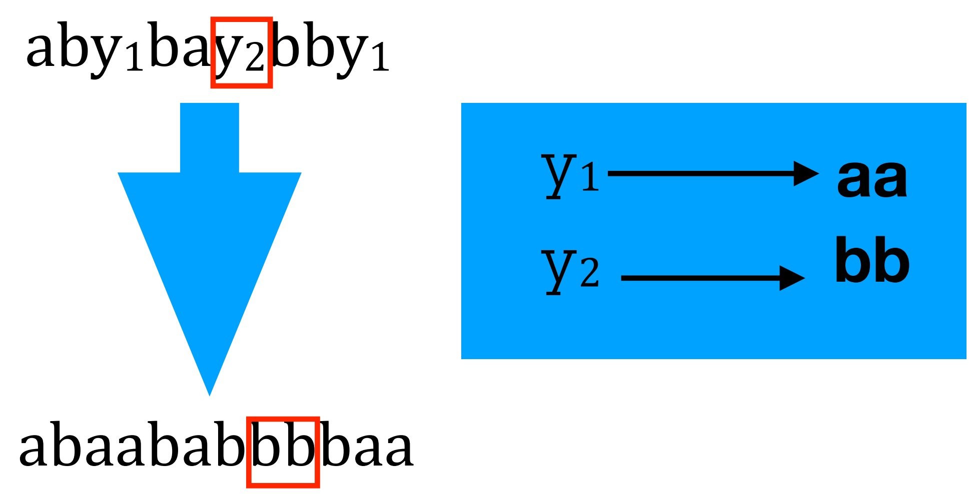
$y \in Y$

String Patterns

$p \in (\Sigma \cup Y)^*$

Instance

$\mathrm{Inst}(p) = \{p\sigma \mid \sigma: Y \to \Sigma^*\}$



String patterns

String variables

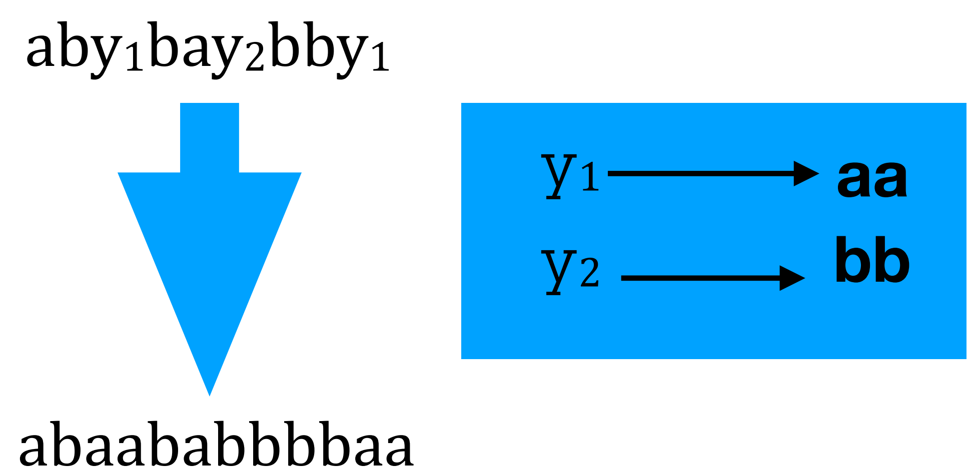
$y \in Y$

String Patterns

$p \in (\Sigma \cup Y)^*$

Instance

$\mathrm{Inst}(p) = \{p\sigma \mid \sigma: Y \to \Sigma^*\}$



Theorem (folklore)

String pattern matching is NP-complete, i.e. whether $w \in \mathrm{Inst}(p)$

Hyperstreams

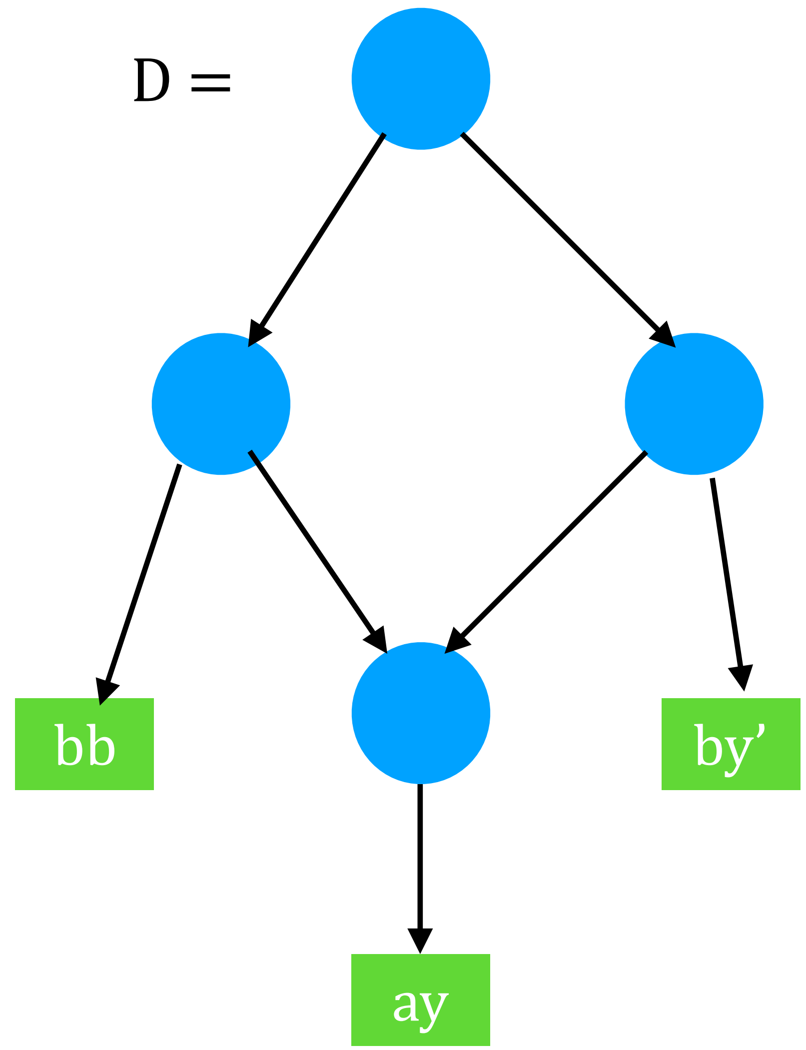
DAGs

leaves are streams

outgoing edges are ordered

inner nodes are called references

String pattern



pat(D) = bbayayby'

Certainty

Given a boolean query Q, a hyperstream D is called:

- certain for selection on D if $\mathrm{Inst}(\mathrm{pat}(\mathrm{D})) \subseteq \mathrm{Q}$
- certain for rejection on D if $\mathrm{Inst}(\mathrm{pat}(\mathrm{D})) \cap \mathrm{Q} = \emptyset$

Certainty

Given a boolean query Q, a hyperstream D is called:

- certain for selection on D if $\mathrm{Inst}(\mathrm{pat}(D)) \subseteq Q$
- certain for rejection on D if $\mathrm{Inst}(\mathrm{pat}(D)) \cap Q = \emptyset$

Theorem

Certainty for selection (resp. rejection) on hyperstreams (resp. string patterns) is PSPACE-complete for queries defined by NFA (resp. DFA).

PSpace hardness

Proof
for string patterns (without compression),
for DFAs (without nondeterminism)

Thm (folklore)
Emptiness of intersection of a list of DFAs is PSPACE-complete.

PSpace hardness

Proof

for string patterns (without compression),
for DFAs (without nondeterminism)

Thm (folklore)

Emptiness of intersection of a list of DFAs is PSPACE-complete.

Reduction

DFAs $A_1, \ldots, A_n$

$L(A_1) \cap \ldots \cap L(A_n) = \emptyset$

iff

$\mathrm{Inst}(\underbrace{y\# \ldots \#y}_{\text{n times}}) \cap L(A_1)\# \ldots \#L(A_n) = \emptyset$ **Rejection**

PSpace completeness

Let D be a hyperstream and A an NFA

Hyperstream D is certain for selection for query L(A)

iff for all substitutions σ from string variables to A-inhabited transitions: $eval^A(D, \sigma) \cap (Q_{init} \times Q_{final}) \neq \emptyset$

All these substitutions can be enumerated and tested in PSPACE.

An efficient fragment

Theorem

Certainty for selection (resp. rejection) on **linear** hyperstreams (resp. on string patterns) for DFA queries is in PTIME.

An efficient fragment

Theorem

Certainty for selection (resp. rejection) on **linear** hyperstreams (resp. on string patterns) for DFA queries is in PTIME.

Proof:

- Inst(pat(D)) can be recognized by an NFA A' of size linear in |D|
- Inclusion $L(A') \subseteq L(A)$ can be decided in time $O(|A'||A|)$ since A is deterministic
- Non empty intersection $L(A') \cap L(A) = \emptyset$ is in PTIME (even for NFAs)

Non boolean queries

- All results remain true in the general case
- One more difficulty in the proof of the efficient fragment

Lifting Efficiency Result for Linear Hyperstreams

Difficulty

Given a hyperstream D and a position π of pat(D), can one uncompress D in PTIME, so that the node of D that represents π is not shared.

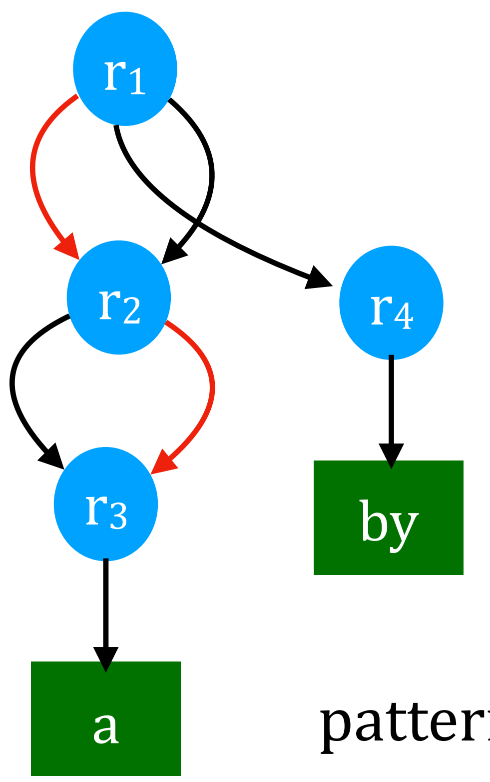
Yes, indeed

Lifting Efficiency Result for Linear Hyperstreams

Difficulty

Given a hyperstream D and a position π of pat(D), can one uncompress D in PTIME, so that the node of D that represents π is not shared.

Yes, indeed



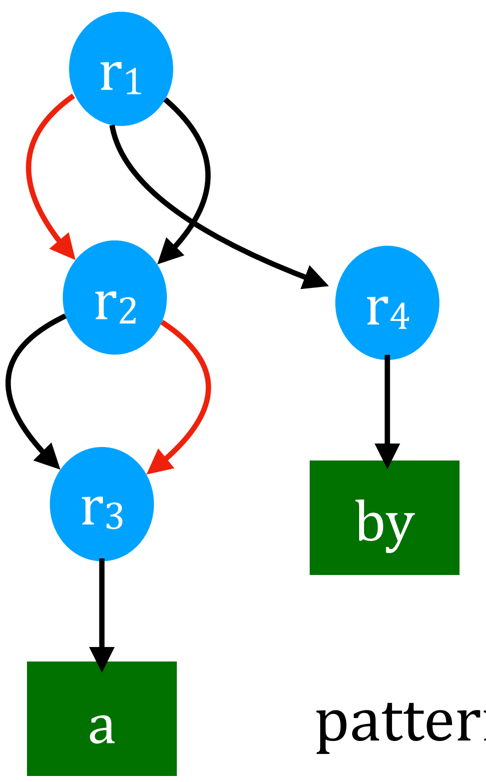
pattern: aabyaaby

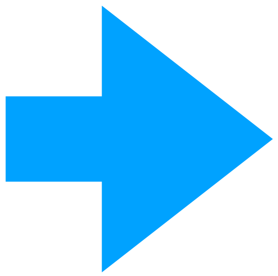
Lifting Efficiency Result for Linear Hyperstreams

Difficulty

Given a hyperstream D and a position π of pat(D), can one uncompress D in PTIME, so that the node of D that represents π is not shared.

Yes, indeed





Unshare position π=2
with linear size increase

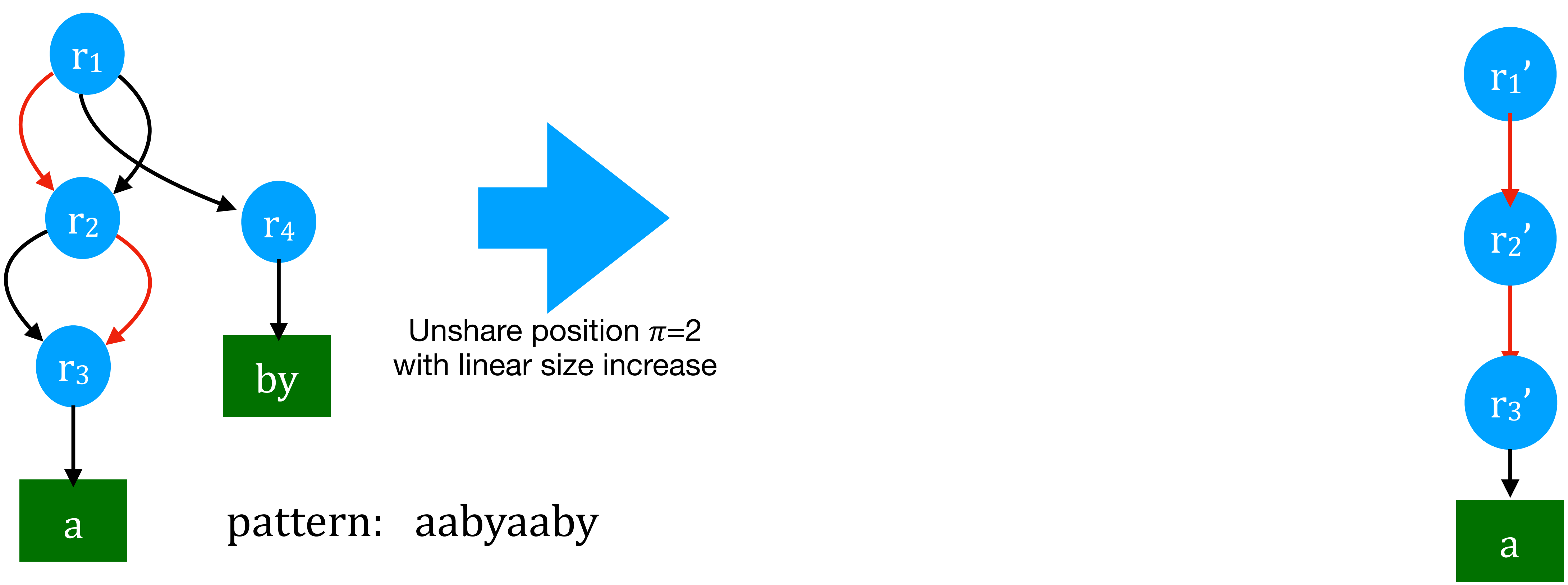
pattern: aabyaaby

Lifting Efficiency Result for Linear Hyperstreams

Difficulty

Given a hyperstream D and a position π of pat(D), can one uncompress D in PTIME, so that the node of D that represents π is not shared.

Yes, indeed

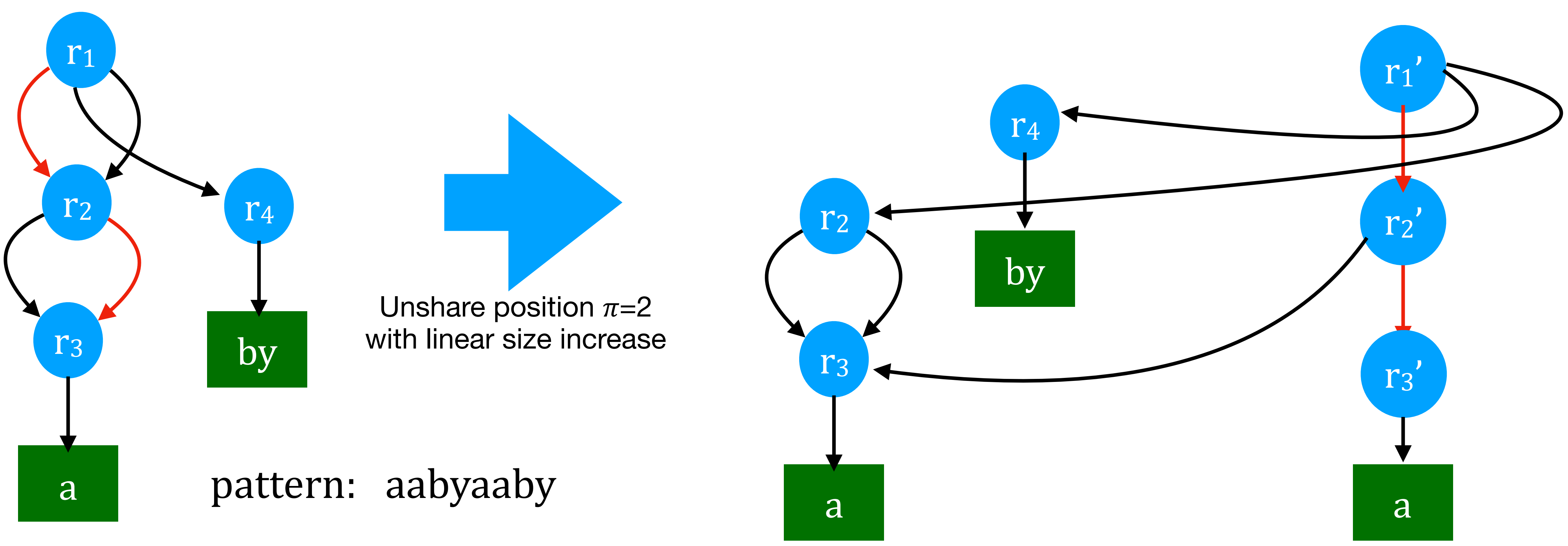


Lifting Efficiency Result for Linear Hyperstreams

Difficulty

Given a hyperstream D and a position π of pat(D), can one uncompress D in PTIME, so that the node of D that represents π is not shared.

Yes, indeed



Conclusions

Complexity of certain query answering increases with hyperstreams.

Still, efficient query answering on hyperstreams may be possible in practice.

- needs incremental algorithm maintaining alive answer candidates
- hope for lowest latency for DFA queries on linear hyperstreams

Extensions needed for hyperstreams with semi-structured data:

- JSON
- RDF

Extensions needed for hyperstreaming query-based programs.

Questions ?