Certain Query Answering on Compressed String Patterns: from Streams to Hyperstreams

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

[1]Inria Lille

[2]University of Lille

12^{th} International Conference on Reachability Problems, September 2018

General Context

- Hyperstreams as a generalization of streams [Maneth 2015], [Labath 2013]
- Certain Query Answering (CQA) [David 2010]
- Nondeterministic Finite Automata (NFAs) as regular path queries [Angles 2016]

From Streams to Hyperstreams

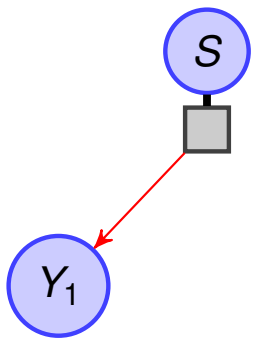
Empty stream

$$S = Y_1$$

From Streams to Hyperstreams

Empty stream

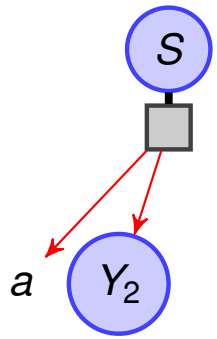
$$S = Y_1$$



From Streams to Hyperstreams

$Y_1 \rightarrow aY_2$

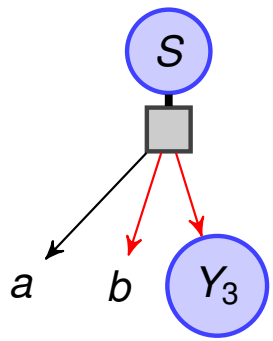
$$S = aY_2$$



From Streams to Hyperstreams

$Y_1 \rightarrow aY_2$, $Y_2 \rightarrow bY_3$,

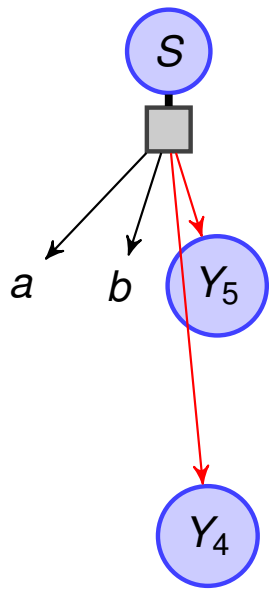
$$S = abY_3$$



From Streams to Hyperstreams

$Y_1 \rightarrow aY_2\ , Y_2 \rightarrow bY_3,$
$Y_3 \rightarrow Y_4 Y_5$

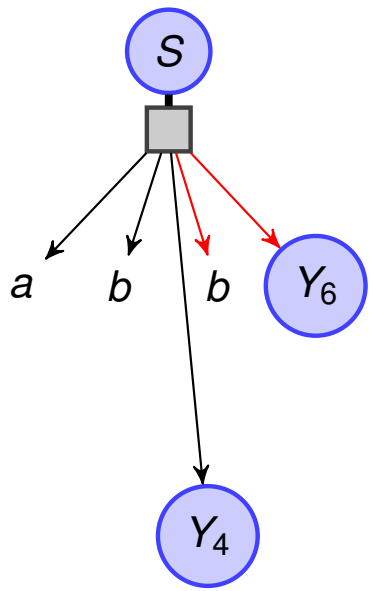
$$S = abY_4 Y_5$$



From Streams to Hyperstreams

$Y_1 \rightarrow aY_2$, $Y_2 \rightarrow bY_3$,
$Y_3 \rightarrow Y_4Y_5$, $Y_5 \rightarrow bY_6$,

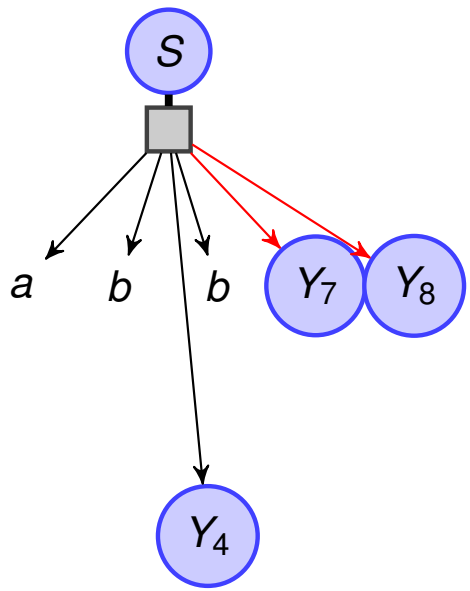
$$S = abY_4bY_6$$



From Streams to Hyperstreams

$Y_1 \rightarrow aY_2\ , Y_2 \rightarrow bY_3,$
$Y_3 \rightarrow Y_4Y_5\ , Y_5 \rightarrow bY_6,$
$Y_6 \rightarrow Y_7Y_8$

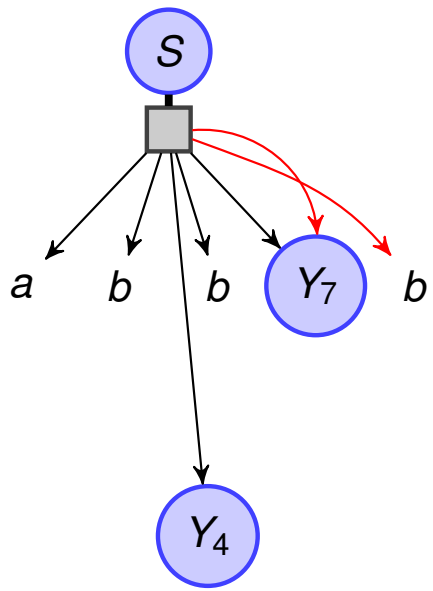
$$S = abY_4bY_7Y_8$$



From Streams to Hyperstreams

$Y_1 \rightarrow aY_2\ , Y_2 \rightarrow bY_3,$
$Y_3 \rightarrow Y_4Y_5\ , Y_5 \rightarrow bY_6,$
$Y_6 \rightarrow Y_7Y_8\ , Y_8 \rightarrow bY_7,$

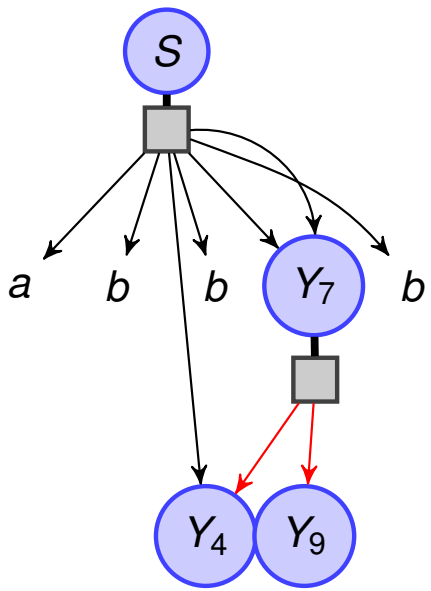
$S = abY_4bY_7bY_7$



From Streams to Hyperstreams

$Y_1 \rightarrow aY_2\ , Y_2 \rightarrow bY_3,$
$Y_3 \rightarrow Y_4 Y_5\ , Y_5 \rightarrow bY_6,$
$Y_6 \rightarrow Y_7 Y_8\ , Y_8 \rightarrow bY_7,$
$Y_7 \rightarrow Y_4 Y_9$

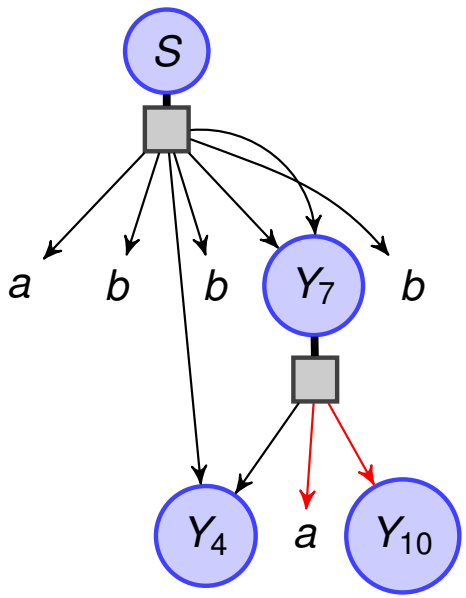
$$S = abY_4bY_4Y_9bY_4Y_9$$



From Streams to Hyperstreams

$Y_1 \rightarrow aY_2$, $Y_2 \rightarrow bY_3$,
$Y_3 \rightarrow Y_4Y_5$, $Y_5 \rightarrow bY_6$,
$Y_6 \rightarrow Y_7Y_8$, $Y_8 \rightarrow bY_7$,
$Y_7 \rightarrow Y_4Y_9$
, $Y_9 \rightarrow aY_{10}$,

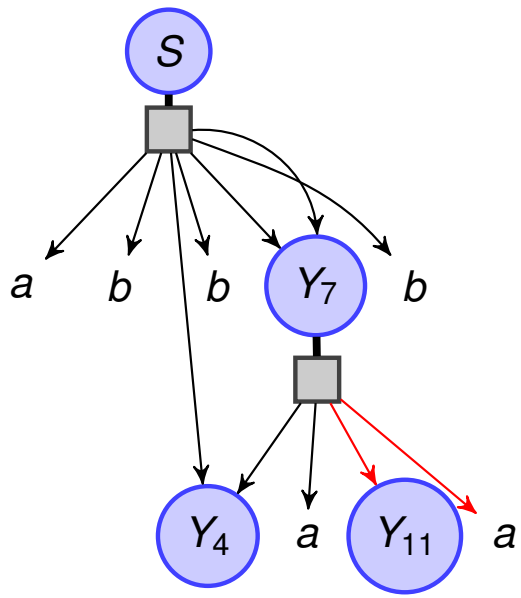
$$S = abY_4bY_4aY_{10}bY_4aY_{10}$$



From Streams to Hyperstreams

$Y_1 \rightarrow aY_2$, $Y_2 \rightarrow bY_3$,
$Y_3 \rightarrow Y_4Y_5$, $Y_5 \rightarrow bY_6$,
$Y_6 \rightarrow Y_7Y_8$, $Y_8 \rightarrow bY_7$,
$Y_7 \rightarrow Y_4Y_9$
, $Y_9 \rightarrow aY_{10}$,
$Y_{10} \rightarrow Y_{11}a$

$$S = abY_4bY_4aY_{11}abY_4aY_{11}a$$

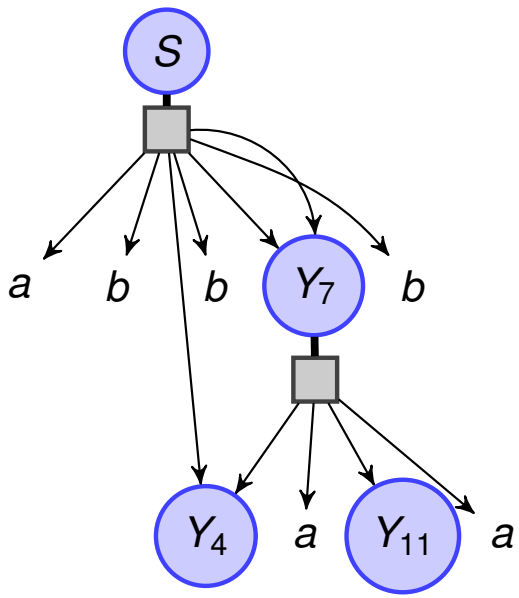


From Streams to Hyperstreams

$Y_1 \rightarrow aY_2$, $Y_2 \rightarrow bY_3$,
$Y_3 \rightarrow Y_4Y_5$, $Y_5 \rightarrow bY_6$,
$Y_6 \rightarrow Y_7Y_8$, $Y_8 \rightarrow bY_7$,
$Y_7 \rightarrow Y_4Y_9$
, $Y_9 \rightarrow aY_{10}$,
$Y_{10} \rightarrow Y_{11}a$

$$S = abY_4bY_4aY_{11}abY_4aY_{11}a$$

Hyperstreams are compressed string patterns.

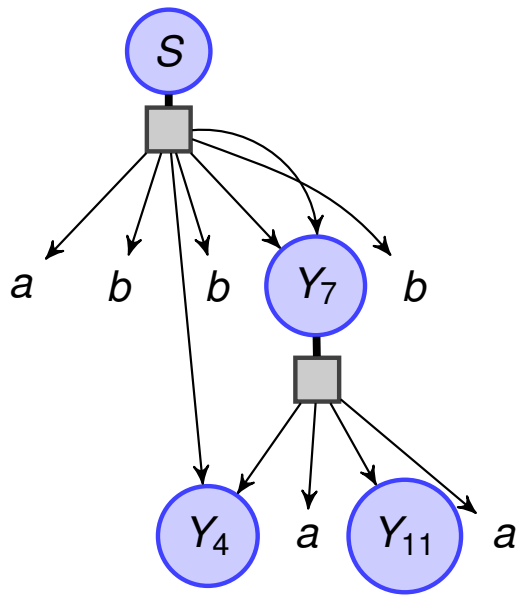


From Streams to Hyperstreams

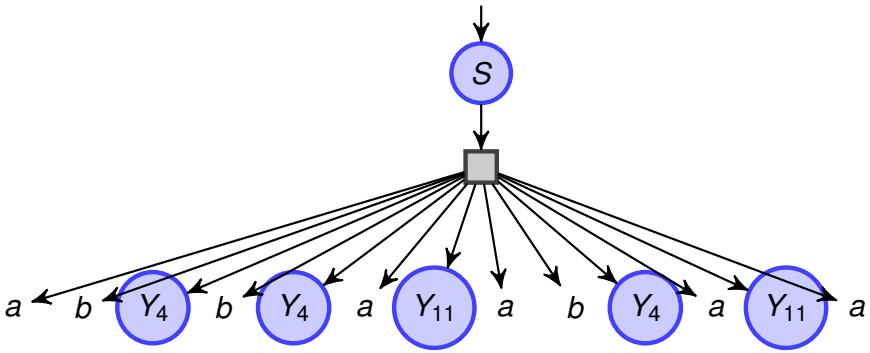
$Y_1 \rightarrow aY_2$, $Y_2 \rightarrow bY_3$,
$Y_3 \rightarrow Y_4Y_5$, $Y_5 \rightarrow bY_6$,
$Y_6 \rightarrow Y_7Y_8$, $Y_8 \rightarrow bY_7$,
$Y_7 \rightarrow Y_4Y_9$
, $Y_9 \rightarrow aY_{10}$,
$Y_{10} \rightarrow Y_{11}a$

$$S = abY_4bY_4aY_{11}abY_4aY_{11}a$$

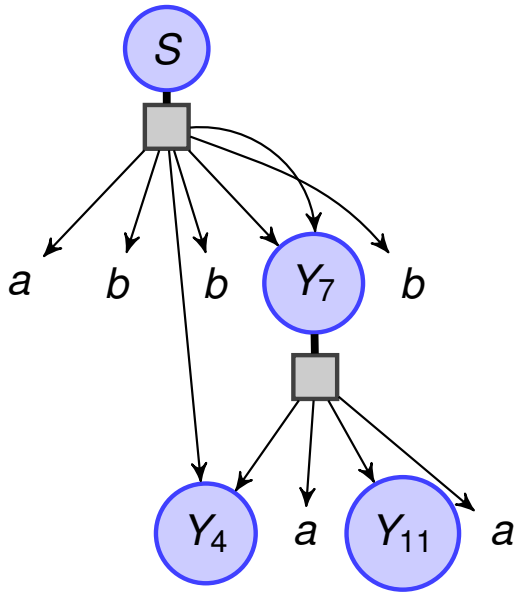
$$G = (N, \Sigma, R, S)$$
$$N = \{S, Y_4, Y_7, Y_{11}\}$$
$$R = \left\{ \begin{array}{l} S \rightarrow abY_4bY_7bY_7, \\ Y_7 \rightarrow Y_4aY_{11}a \end{array} \right\}$$



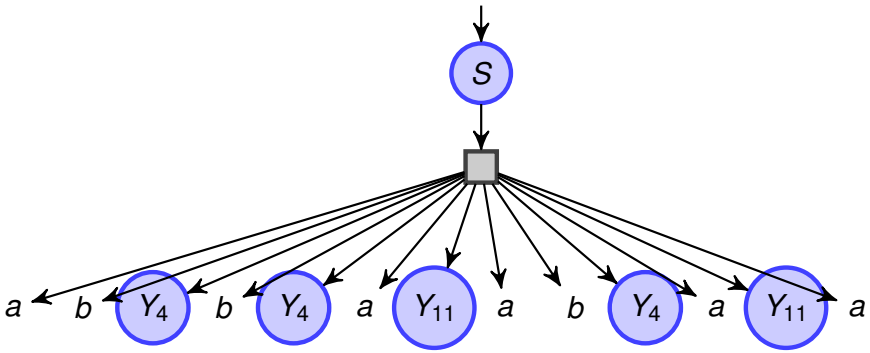
From Streams to Hyperstreams



$$G = (N, \Sigma, R, S)$$
$$N = \{S, Y_4, Y_7, Y_{11}\}$$
$$R = \left\{ \begin{array}{l} S \to abY_4bY_7bY_7, \\ Y_7 \to Y_4aY_{11}a \end{array} \right\}$$



From Streams to Hyperstreams

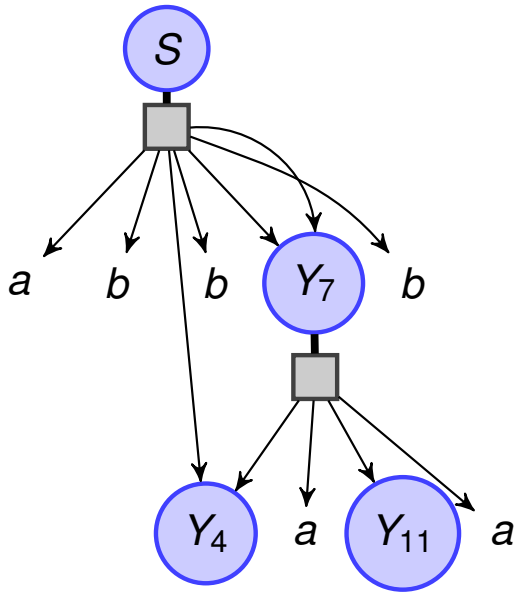


$G = (N, \Sigma, R, S)$

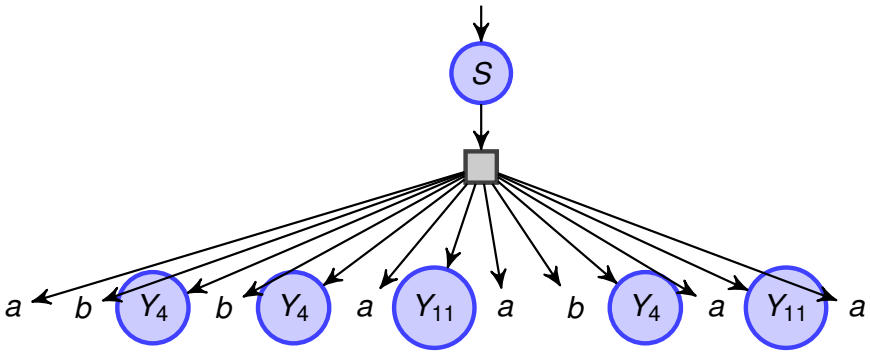
$N = \{S, Y_4, Y_7, Y_{11}\}$

$$R = \left\{ \begin{array}{l} S \to abY_4bY_7bY_7, \\ Y_7 \to Y_4aY_{11}a \end{array} \right\}$$

Represents $pat(G)$



From Streams to Hyperstreams



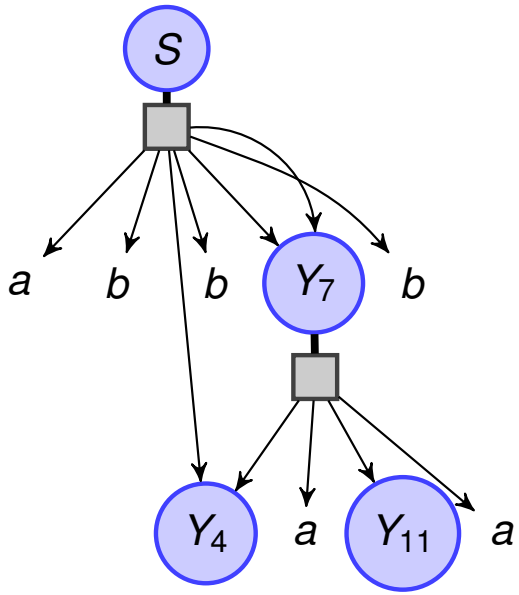
$G = (N, \Sigma, R, S)$

$N = \{S, Y_4, Y_7, Y_{11}\}$

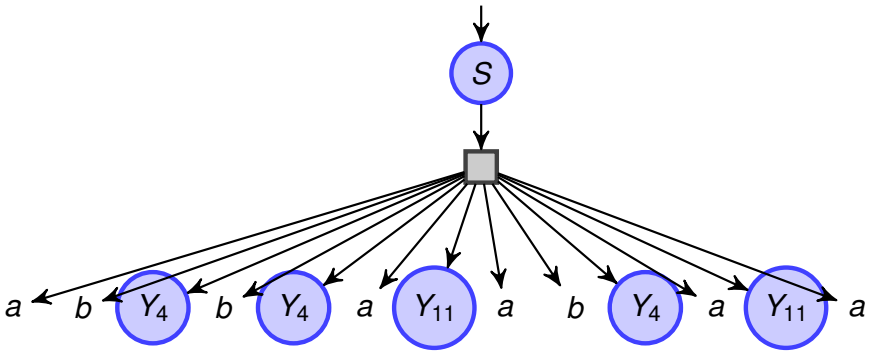
$$R = \left\{ \begin{array}{l} S \rightarrow abY_4bY_7bY_7, \\ Y_7 \rightarrow Y_4aY_{11}a \end{array} \right\}$$

Represents $pat(G)$

$Inst(G) = Inst(pat(G))$



From Streams to Hyperstreams



$G = (N, \Sigma, R, S)$

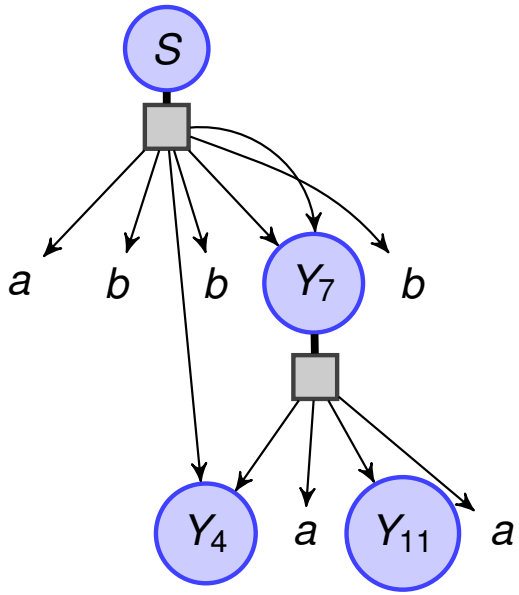
$N = \{S, Y_4, Y_7, Y_{11}\}$

$$R = \left\{ \begin{array}{l} S \rightarrow abY_4bY_7bY_7, \\ Y_7 \rightarrow Y_4aY_{11}a \end{array} \right\}$$

Represents $pat(G)$

$Inst(G) = Inst(pat(G))$

Not linear



Outline

Outline

Queries

Queries

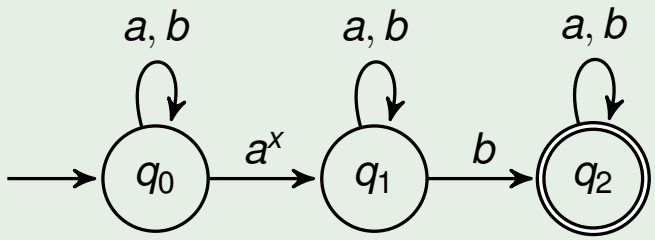
Example (Monadic query)

$\mathbf{Q}_1(w) =$ Set of positions of w labeled by a and followed by b.

Queries

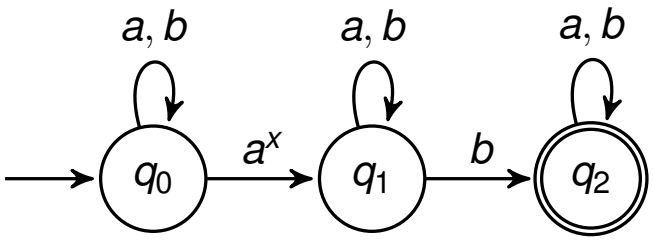
Example (Monadic query)

$\mathbf{Q}_1(w) =$ Set of positions of w labeled by a and followed by b.

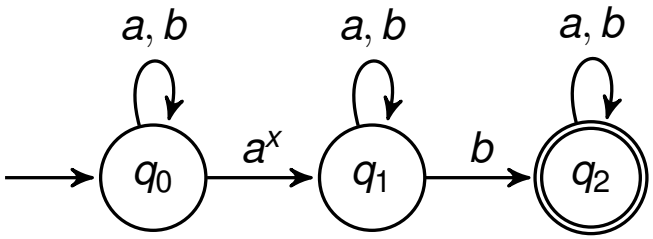


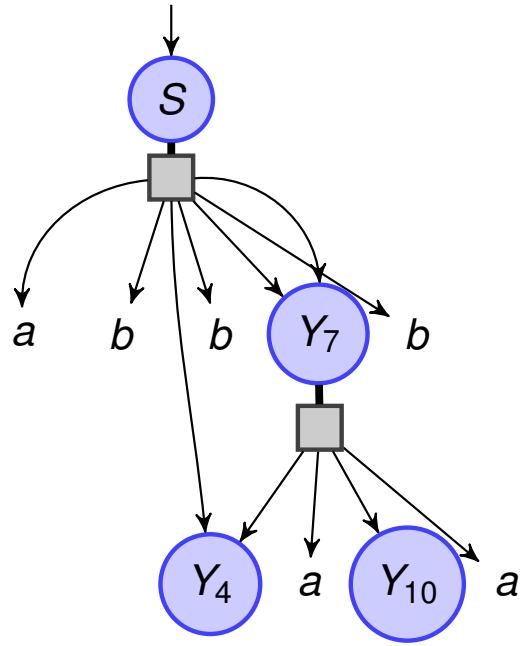
Certain (non) Answers

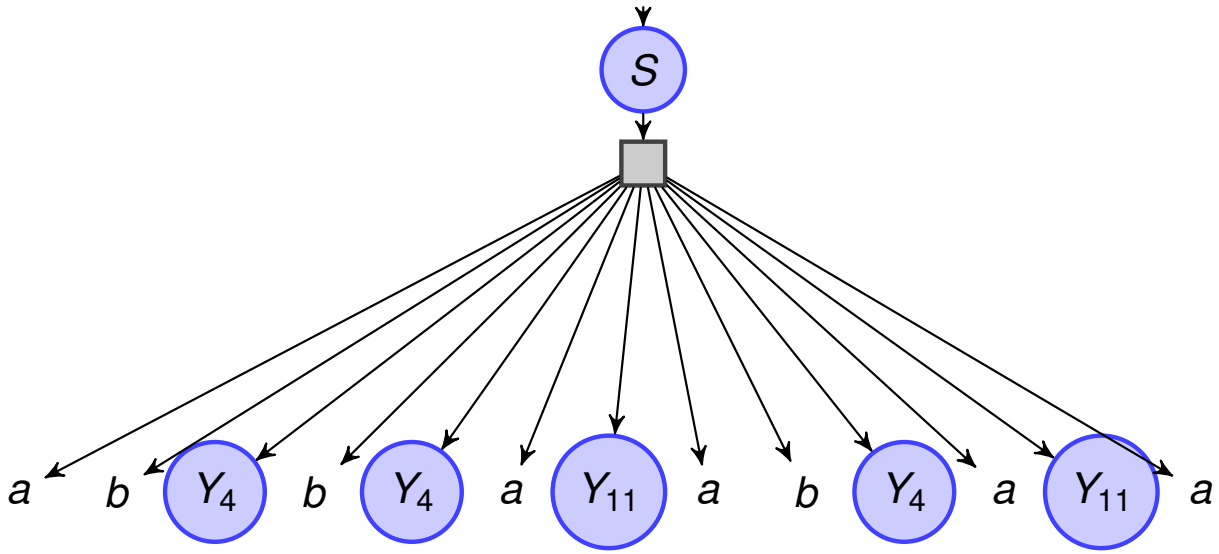
Certain (non) Answers



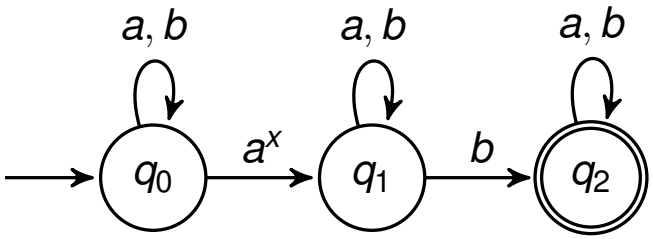
Certain (non) Answers

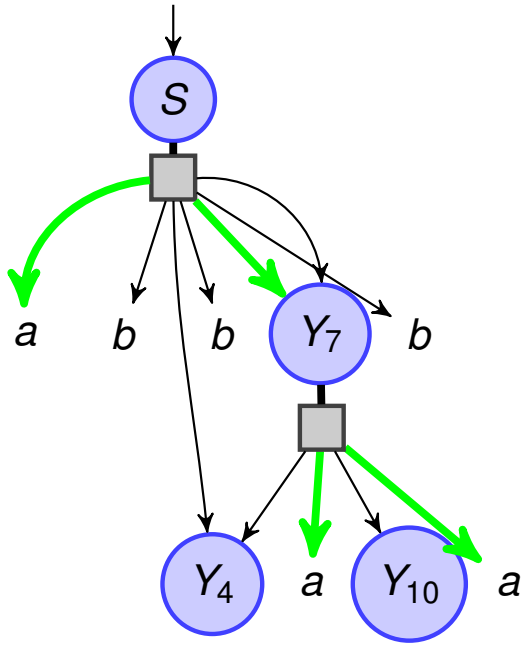




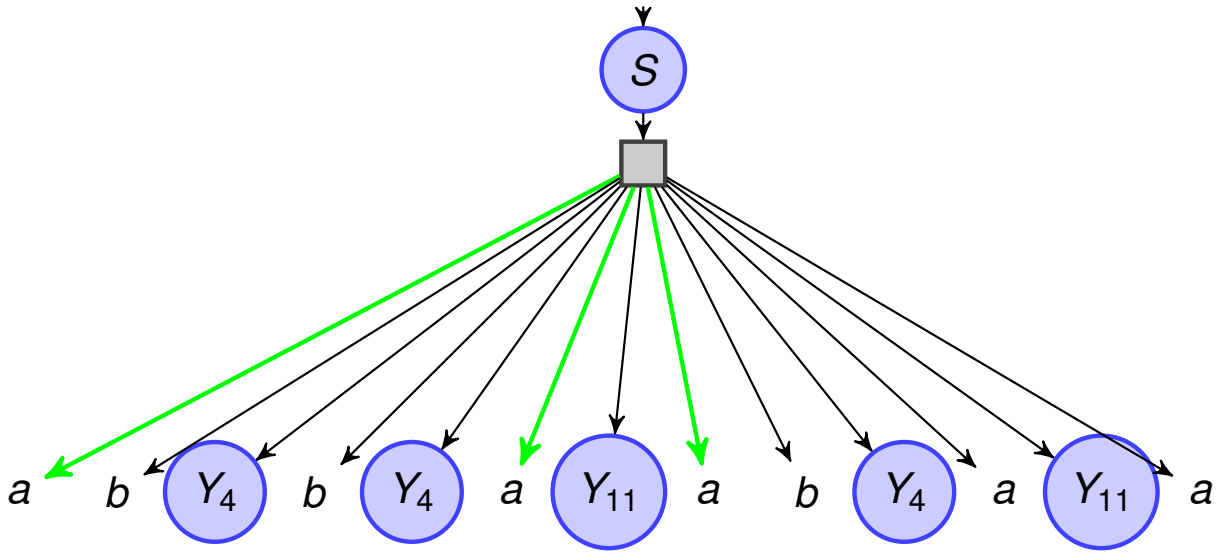


Certain (non) Answers

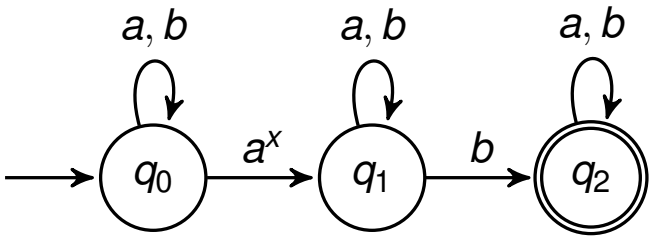


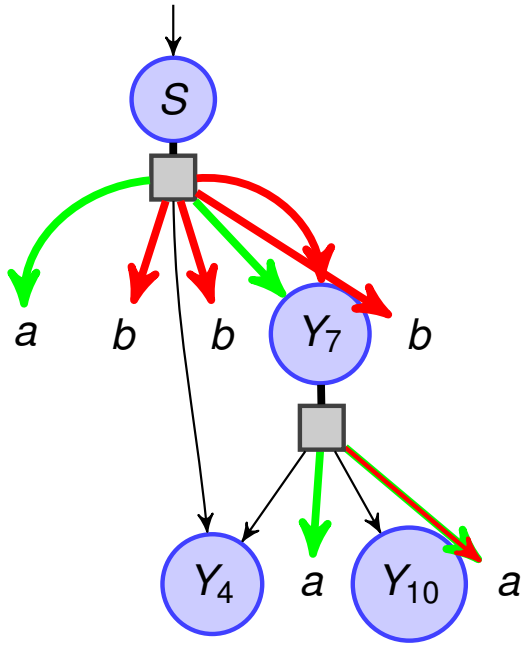


Certain answers



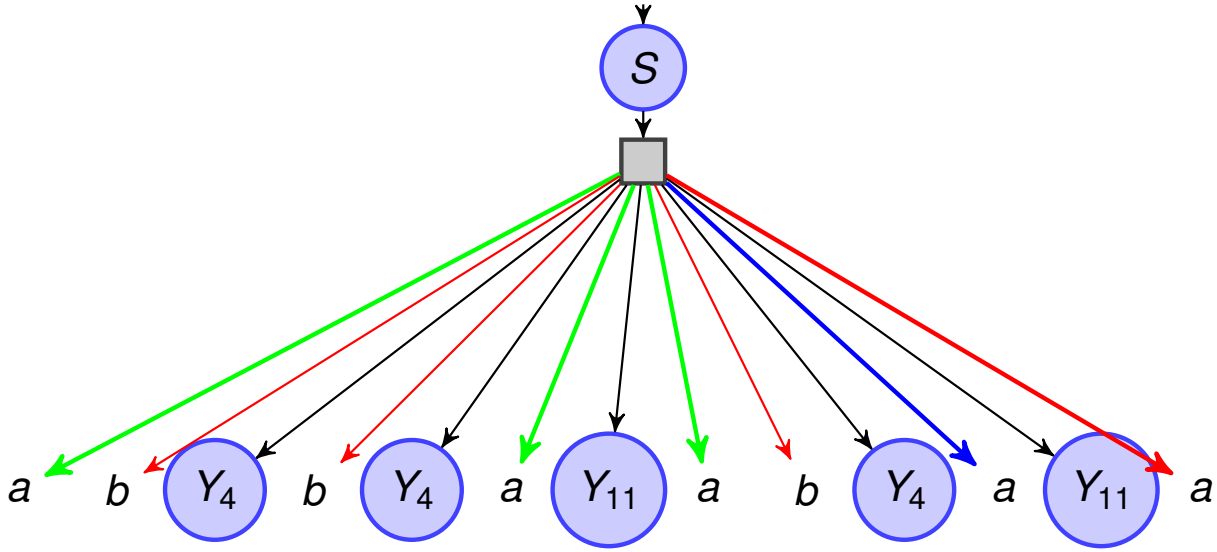
Certain (non) Answers



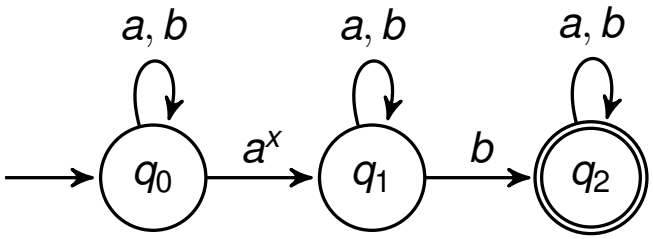


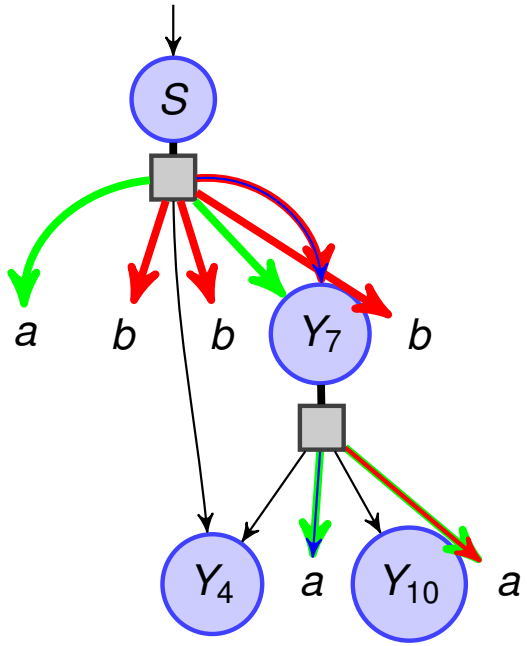
Certain answers

Certain non answers



Certain (non) Answers

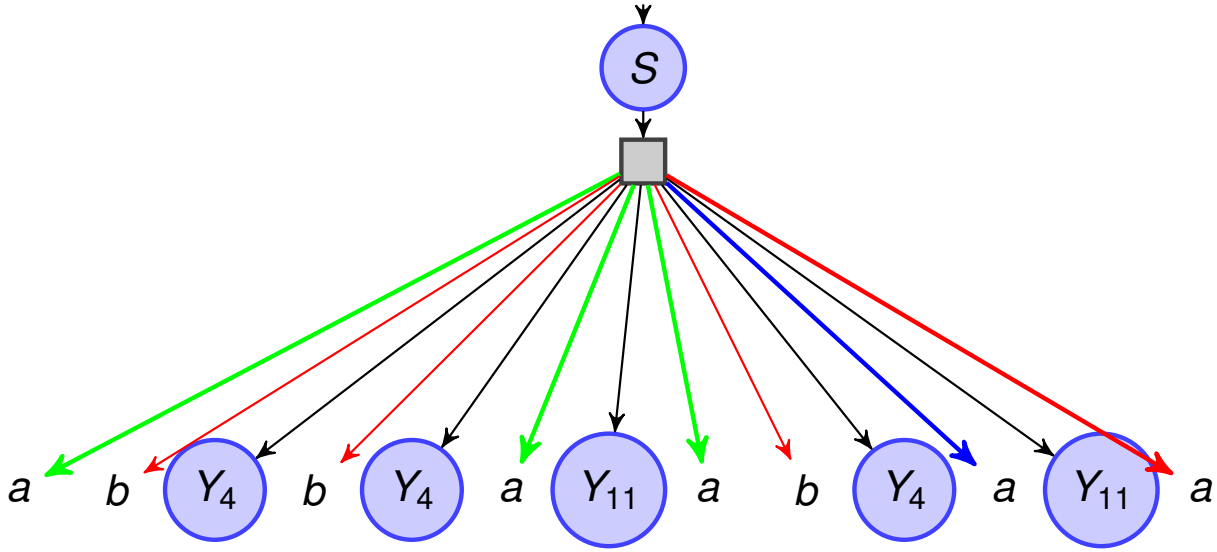




Certain answers

Certain non answers

Neither certain nor certain non answers



Problem

Certain query answering problem

Input: query automaton A, hyperstream G, tuple of positions π of symbols of Σ on $pat(G)$.
Output: π certain answer for $L(A)$ on G ?

Problem

Certain query answering problem

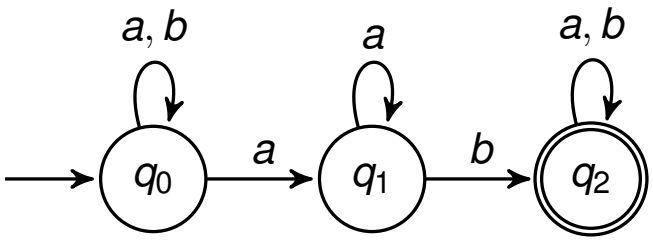
Input: query automaton A, hyperstream G, tuple of positions π of symbols of Σ on $pat(G)$.
Output: π certain answer for $L(A)$ on G ?

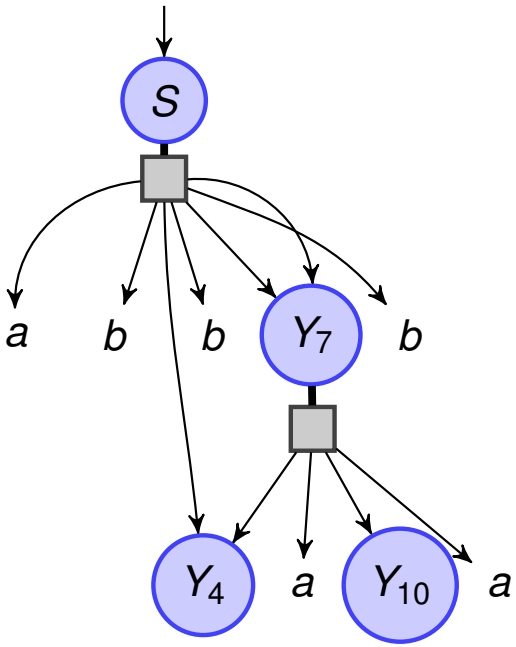
Certain query non answering problem

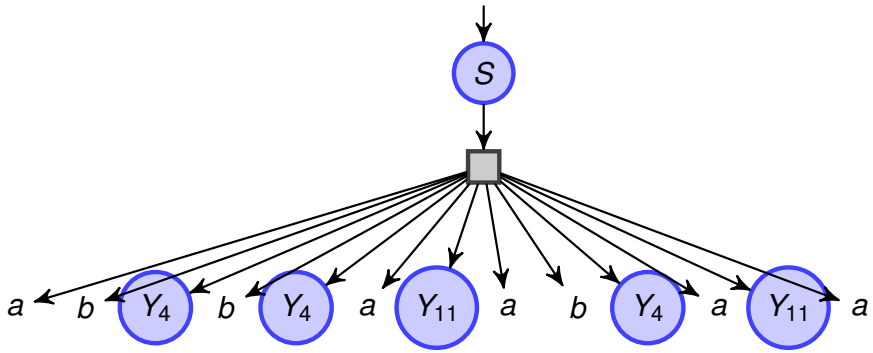
Input: query automaton A, hyperstream G, tuple of positions π of symbols of Σ on $pat(G)$.
Output: π certain non answer for $L(A)$ on G ?

Equivalences in the boolean case

Boolean query automaton A, hyperstream G

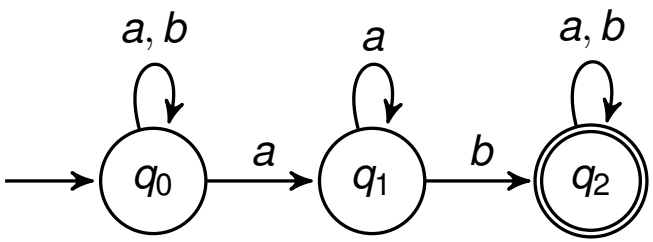




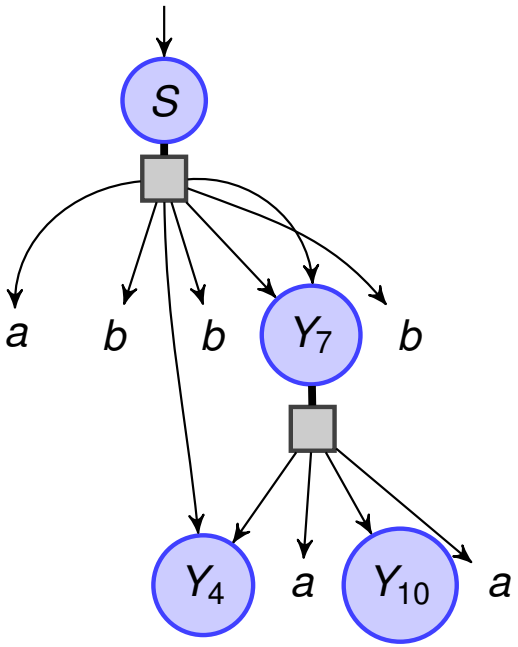


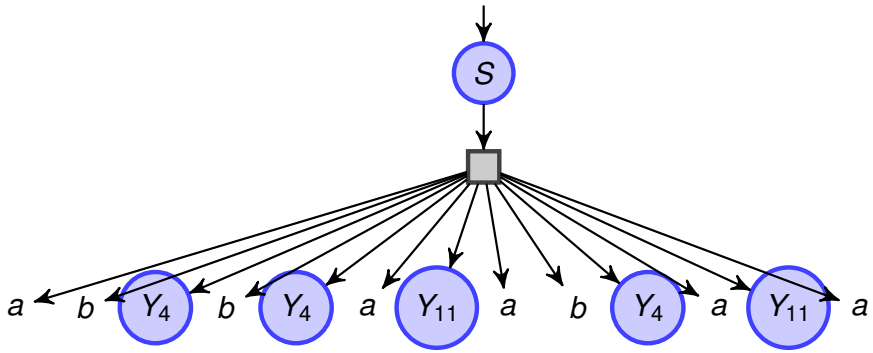
Equivalences in the boolean case

Boolean query automaton A, hyperstream G



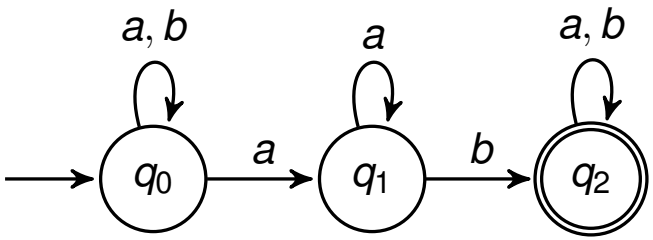
() certain answer iff
$Inst(G) = Inst(pat(G)) \subseteq L(A)$





Equivalences in the boolean case

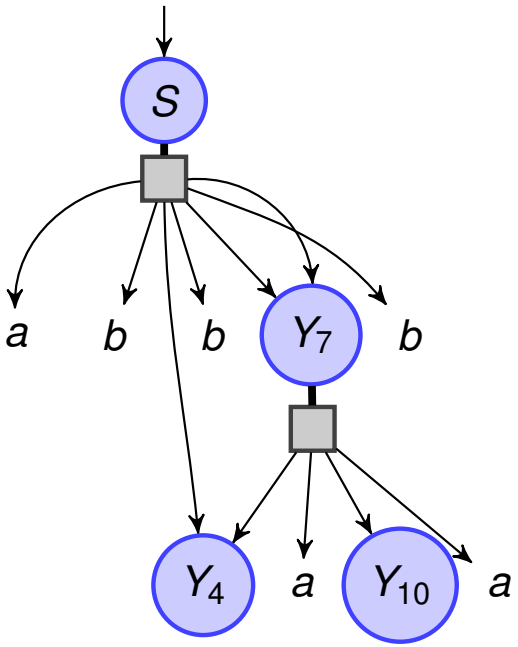
Boolean query automaton A, hyperstream G

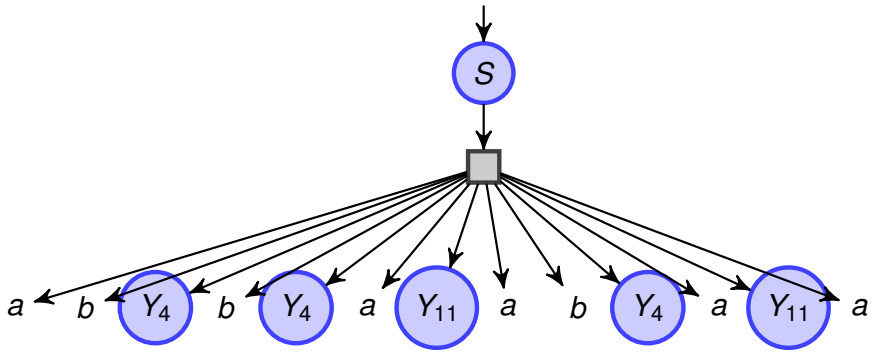


$CQA_{bool} = REGINCL$

() certain answer iff

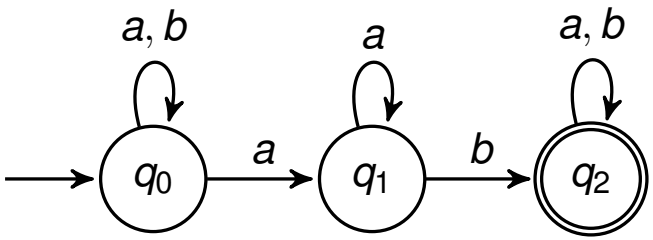
$Inst(G) = Inst(pat(G)) \subseteq L(A)$





Equivalences in the boolean case

Boolean query automaton A, hyperstream G



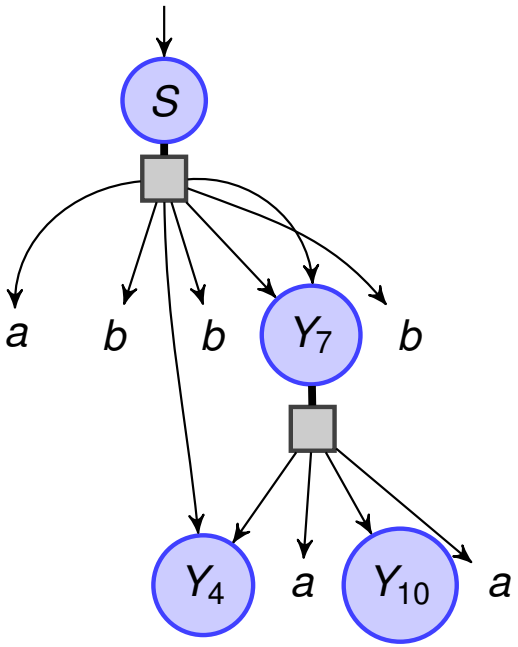
$CQA_{bool} = REGINCL$

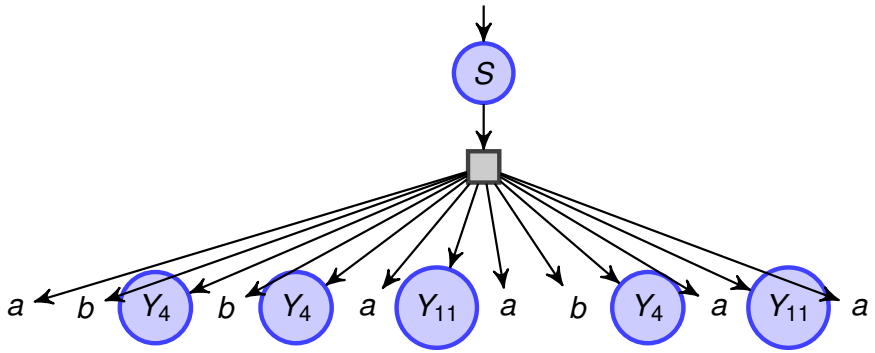
() certain answer iff

$Inst(G) = Inst(pat(G)) \subseteq L(A)$

() certain non answer iff

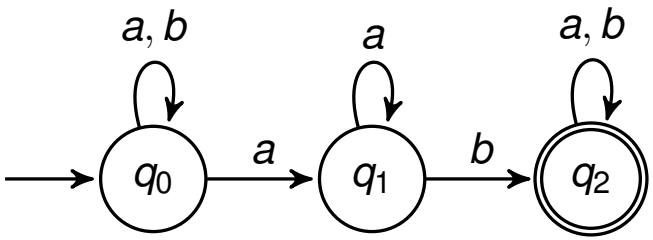
$Inst(G) = Inst(pat(G)) \cap L(A) = \emptyset$





Equivalences in the boolean case

Boolean query automaton A, hyperstream G



$CQA_{bool} = REGINCL$

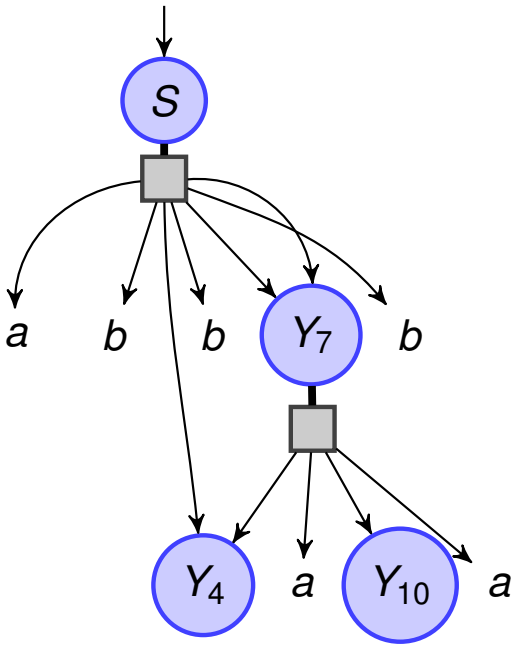
() certain answer iff

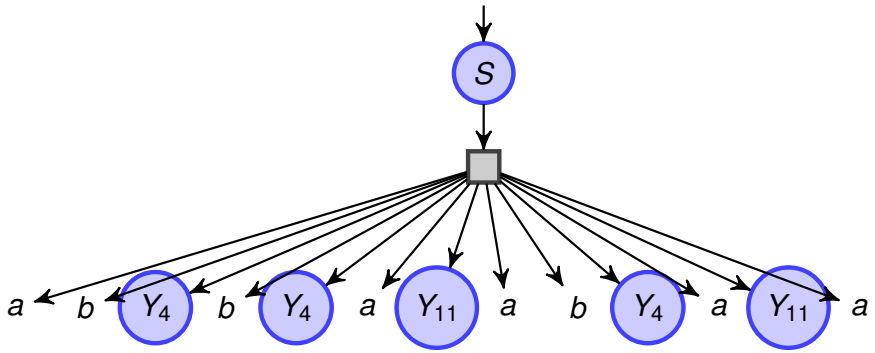
$Inst(G) = Inst(pat(G)) \subseteq L(A)$

$CQ\neg A_{bool} = coREGMATCH$

() certain non answer iff

$Inst(G) = Inst(pat(G)) \cap L(A) = \emptyset$





Results

Known results (Complexity of CQA on streams [Gauwin 2009])

	DFAs	NFAs
Answers	PTIME	PSPACE-c
Non-answers	PTIME	PTIME

Results

Known results (Complexity of CQA on streams [Gauwin 2009])

	DFAs	NFAs
Answers	PTIME	PSPACE-c
Non-answers	PTIME	PTIME

Contribution (Complexity of CQA on **Hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	PSPACE-c	PSPACE-c
Non-answers (*coREGMATCH*)	PSPACE-c	PSPACE-c

Results

Known results (Complexity of CQA on streams [Gauwin 2009])

	DFAs	NFAs
Answers	PTIME	PSPACE-c
Non-answers	PTIME	PTIME

Contribution (Complexity of CQA on **linear hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	PTIME	PSPACE-c
Non-answers (*coREGMATCH*)	PTIME	PTIME

Outline

Hardness of regular compressed pattern inclusion

Lemma

REGINCL(*Linear_String_patterns*, NFA*s*) *is* PSPACE*-hard.*

Hardness of regular compressed pattern inclusion

Lemma

REGINCL(*Linear_String_patterns*, NFA*s*) *is* PSPACE*-hard.*

Proof.

String pattern Y and NFA A

Hardness of regular compressed pattern inclusion

Lemma

REGINCL(*Linear_String_patterns*, NFA*s*) *is* PSPACE*-hard.*

Proof.

String pattern Y and NFA A
$Inst(Y) \subseteq L(A)$ iff $L(A) = \Sigma^*$.

Hardness of regular compressed pattern inclusion

Lemma

REGINCL(*Linear_String_patterns*, NFA*s*) *is* PSPACE*-hard.*

Proof.

String pattern Y and NFA A
$Inst(Y) \subseteq L(A)$ iff $L(A) = \Sigma^*$.
PSPACE-complete [Kozen 1977]. □

Contribution (Complexity of CQA on **Hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	-
Non-answers (*coREGMATCH*)	-	-

Contribution (Complexity of CQA on **linear hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-hard
Non-answers (*coREGMATCH*)	-	-

Contribution (Complexity of CQA on **Hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-hard
Non-answers (*coREGMATCH*)	-	-

Contribution (Complexity of CQA on **linear hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-hard
Non-answers (*coREGMATCH*)	-	-

Complexity of regular compressed pattern matching

Theorem ([Angluin 1980])

The simpler problem of string pattern matching is NP-complete.

Complexity of regular compressed pattern matching

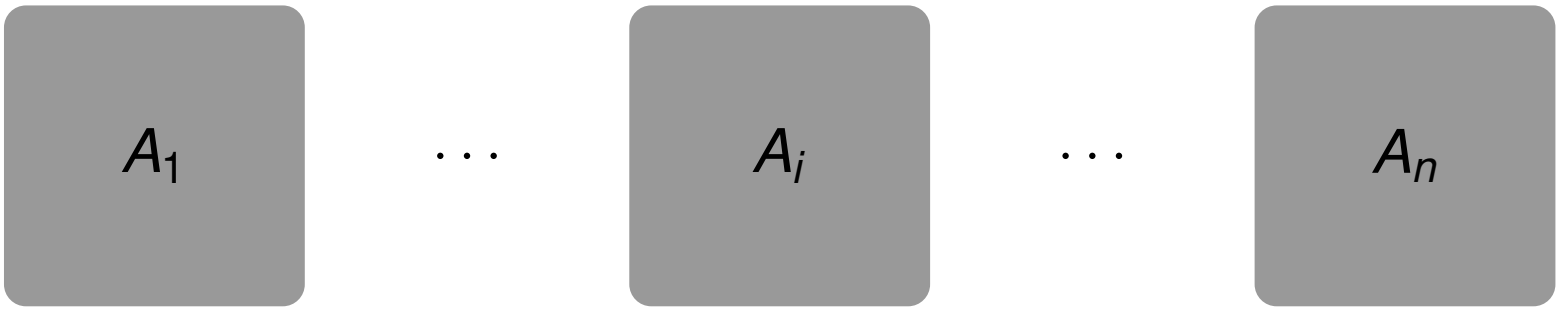
Theorem ([Angluin 1980])

The simpler problem of string pattern matching is NP-complete.

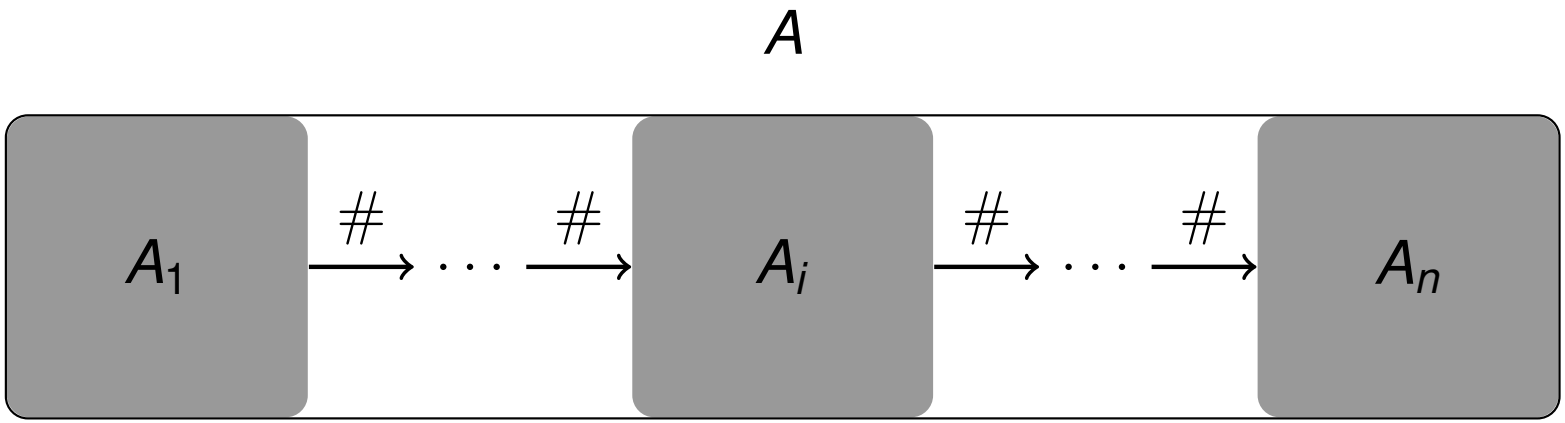
Theorem

REGINCL(*String_patterns*, DFA*s*) *is* PSPACE*-complete.*

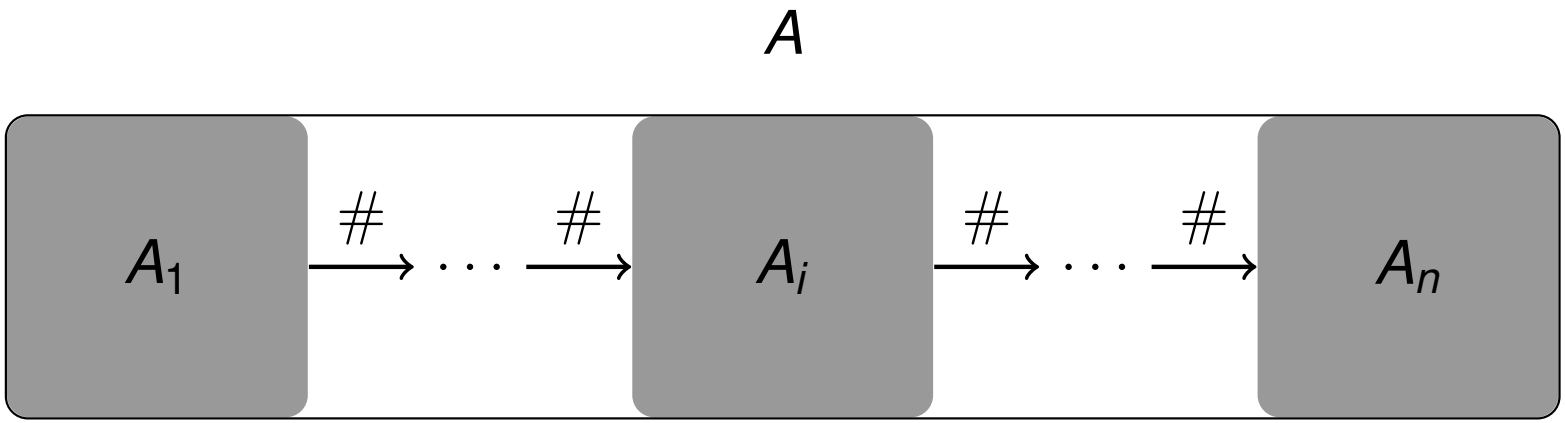
Hardness of regular compressed pattern matching



Hardness of regular compressed pattern matching

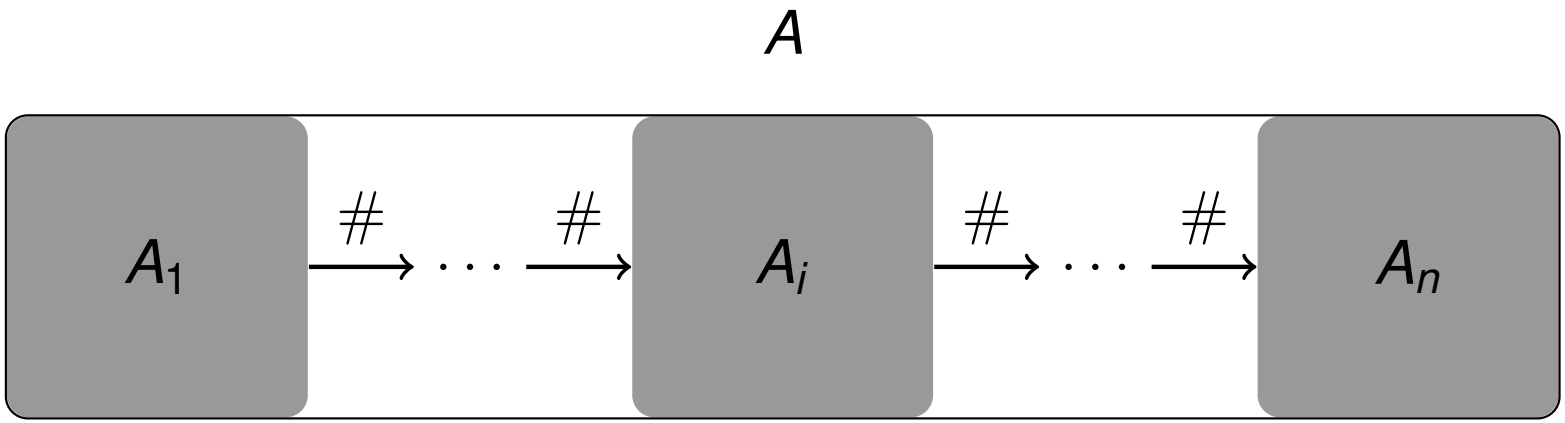


Hardness of regular compressed pattern matching



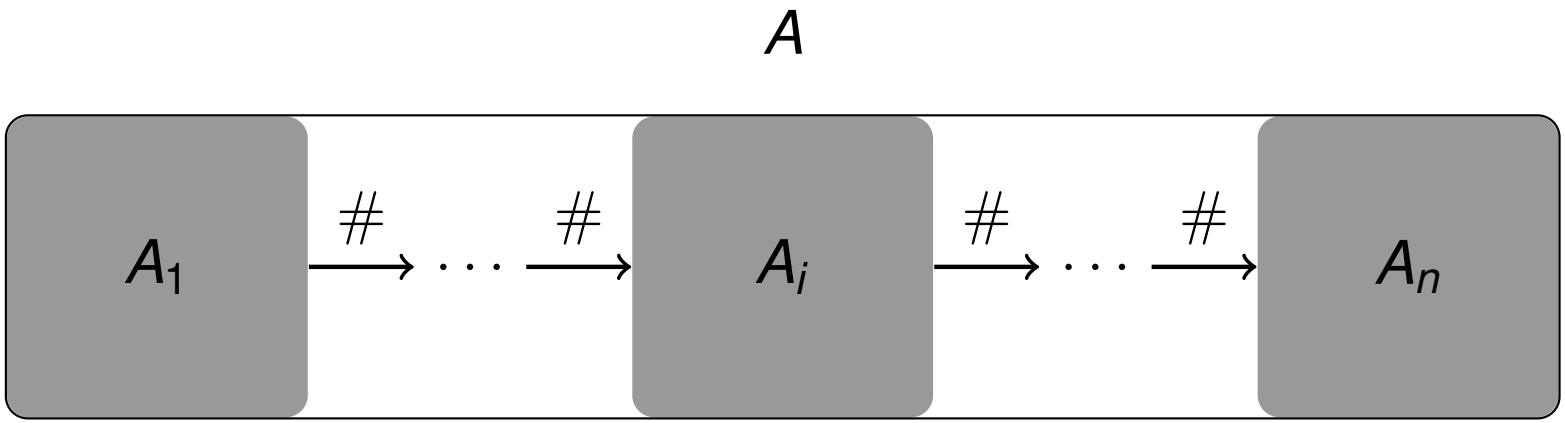
$$L(A) = \{u_1 \# \ldots \# u_n \mid u_1 \in L(A_1), \ldots, u_n \in L(A_n)\}$$

Hardness of regular compressed pattern matching



$L(A) = \{u_1 \# \ldots \# u_n \mid u_1 \in L(A_1), \ldots, u_n \in L(A_n)\}$
String pattern $p = \underbrace{Y \# \ldots \# Y}_{n}$

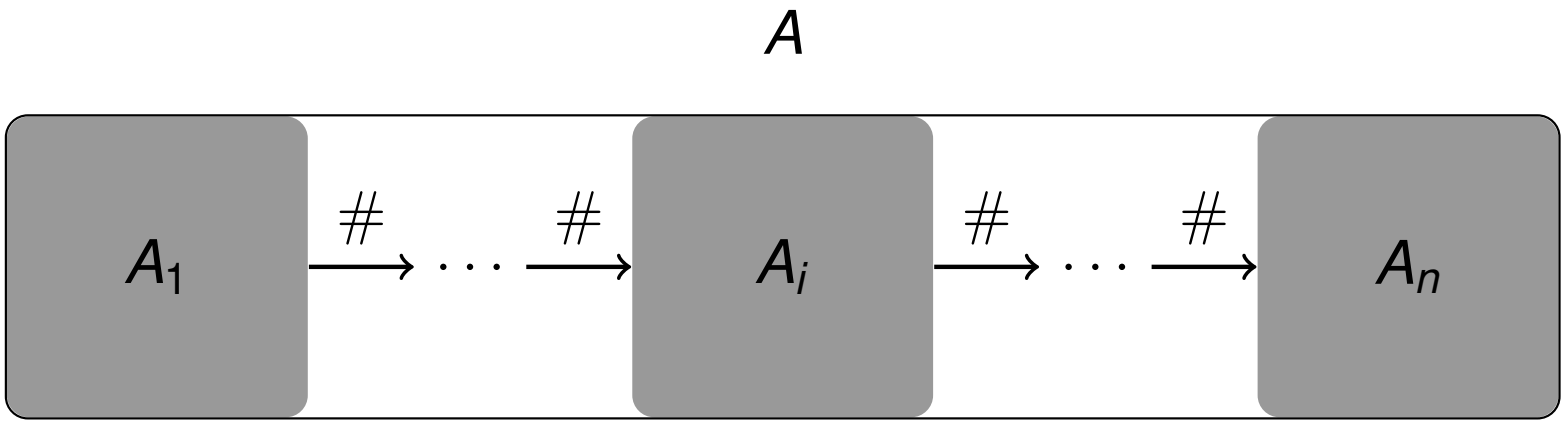
Hardness of regular compressed pattern matching



$L(A) = \{u_1 \# \ldots \# u_n \mid u_1 \in L(A_1), \ldots, u_n \in L(A_n)\}$
String pattern $p = \underbrace{Y \# \ldots \# Y}_{n}$

$Inst(p) \cap L(A) \neq \emptyset$ iff $L(A_1) \cap \ldots \cap L(A_n) \neq \emptyset$.

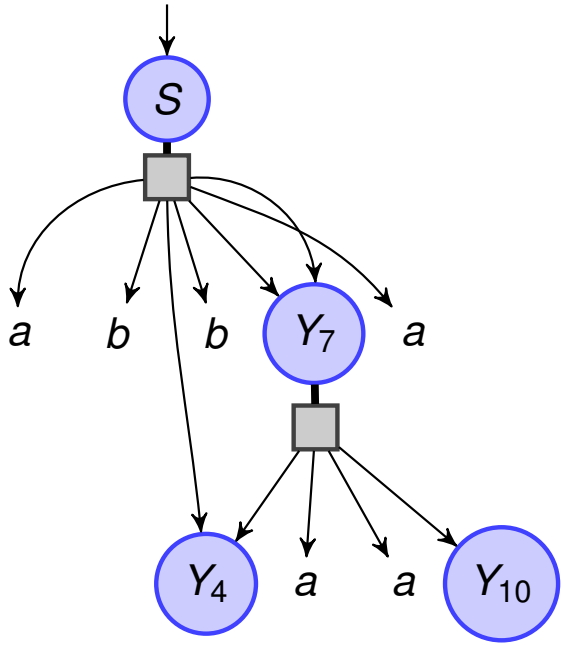
Hardness of regular compressed pattern matching



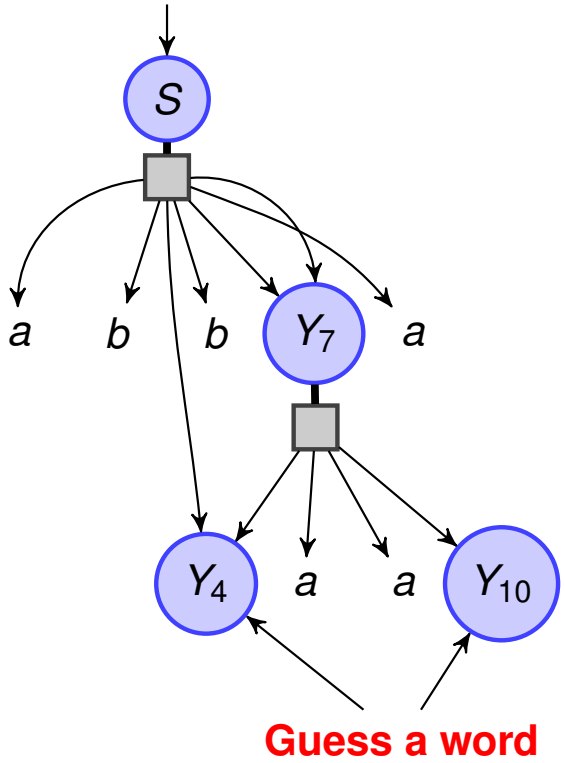
$L(A) = \{u_1 \# \ldots \# u_n \mid u_1 \in L(A_1), \ldots, u_n \in L(A_n)\}$
String pattern $p = \underbrace{Y \# \ldots \# Y}_{n}$

$Inst(p) \cap L(A) \neq \emptyset$ iff $L(A_1) \cap \ldots \cap L(A_n) \neq \emptyset$.
PSPACE-complete [Kozen 1977].

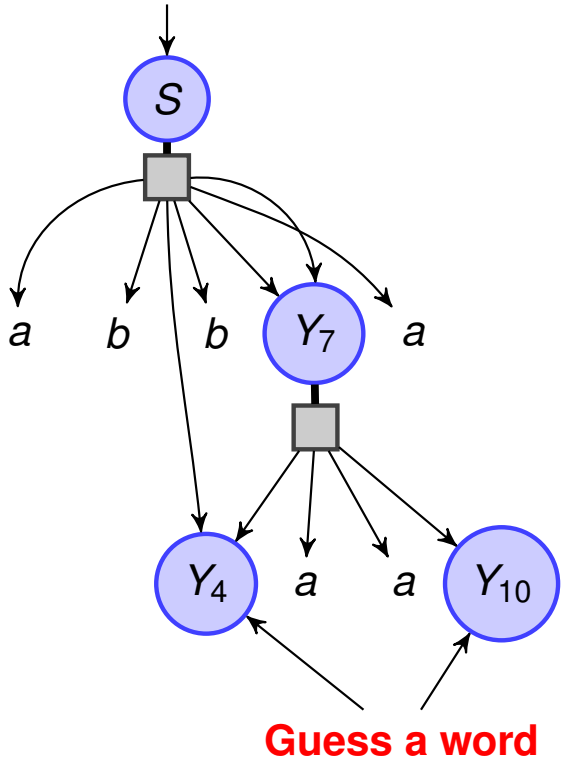
How to decide regular compressed pattern matching ?



How to decide regular compressed pattern matching ?

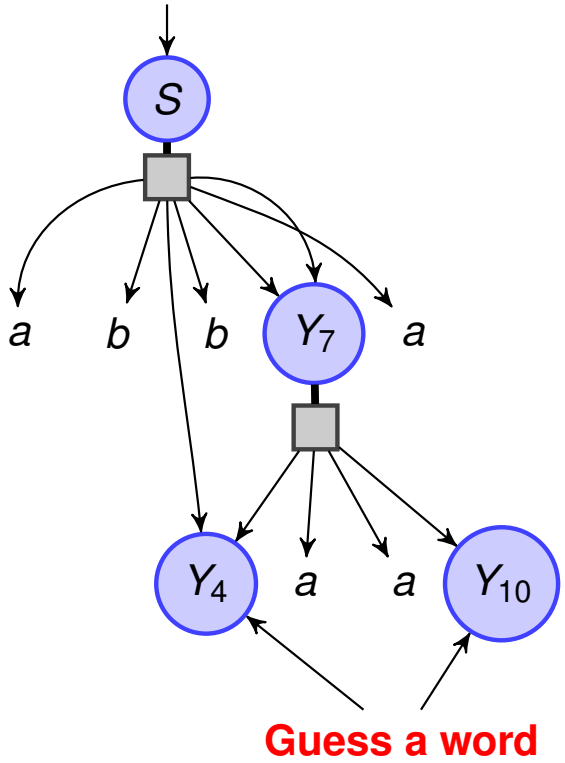


How to decide regular compressed pattern matching ?



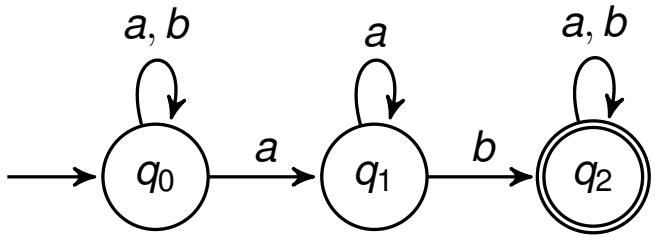
Σ^* is not finite...

How to decide regular compressed pattern matching ?

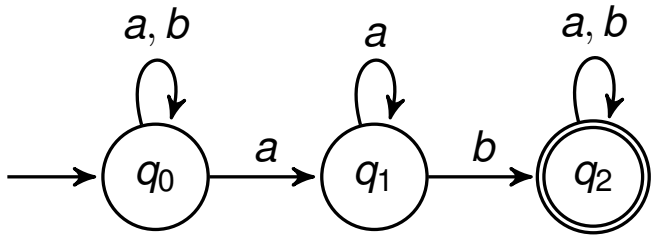


Σ^* is not finite... but the transition monoid is !

Transition monoid

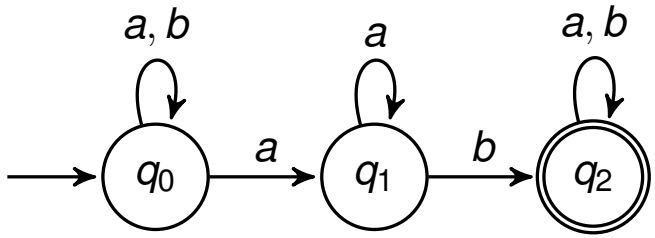


Transition monoid



	τ_a
q_0	q_0
q_1	q_1
q_2	q_2

Transition monoid



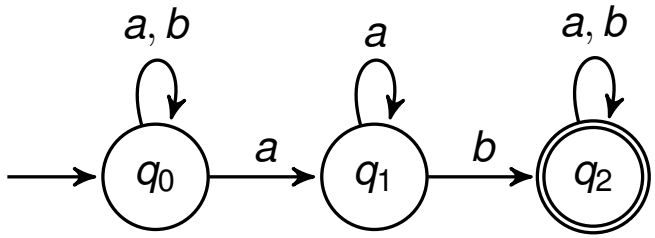
τ_a

$q_0 \longrightarrow q_0$

$q_1 \longrightarrow q_1$

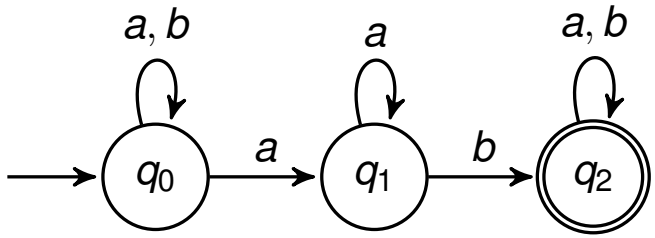
$q_2 \longrightarrow q_2$

Transition monoid



	τ_a	τ_b
q_0	q_0	q_0
q_1	q_1	q_1
q_2	q_2	q_2

Transition monoid



τ_a

$q_0 \longrightarrow q_0$

$q_1 \longrightarrow q_1$

$q_2 \longrightarrow q_2$

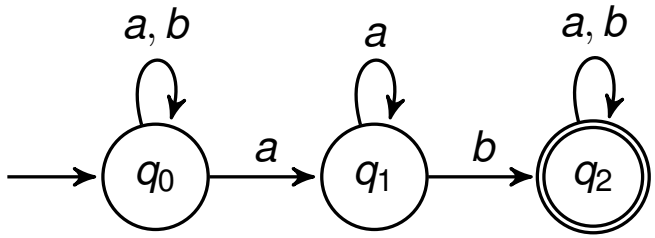
τ_b

$q_0 \longrightarrow q_0$

$q_1 \longrightarrow q_1$

$q_2 \longrightarrow q_2$

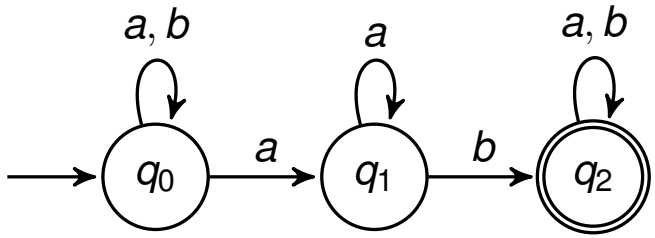
Transition monoid



τ_a τ_b

$$\begin{array}{ccc} q_0 \longrightarrow q_0 & & q_0 \longrightarrow q_0 \\ \searrow & & \\ q_1 \longrightarrow q_1 & \circ & q_1 \longrightarrow q_1 \\ & & \searrow \\ q_2 \longrightarrow q_2 & & q_2 \longrightarrow q_2 \end{array}$$

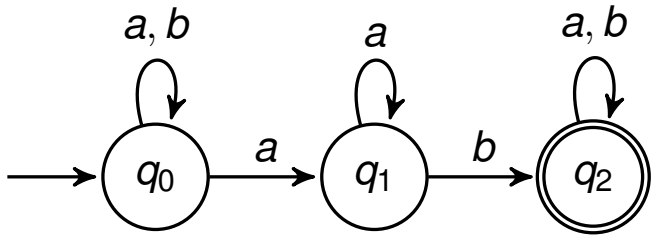
Transition monoid



τ_a τ_b τ_{ab}

$$
\begin{array}{ccc}
q_0 \longrightarrow q_0 & q_0 \longrightarrow q_0 & q_0 \quad q_0 \\
\searrow & & \\
q_1 \longrightarrow q_1 & \circ \quad q_1 \longrightarrow q_1 & q_1 \quad q_1 \\
 & \searrow & \\
q_2 \longrightarrow q_2 & q_2 \longrightarrow q_2 & q_2 \quad q_2
\end{array}
$$

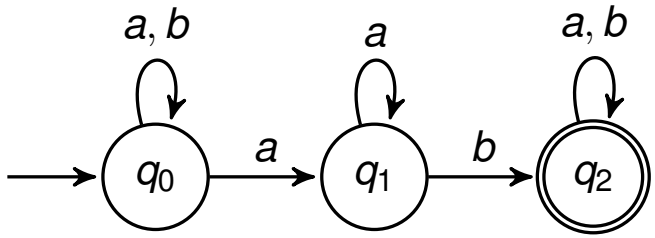
Transition monoid



τ_a $\quad\quad$ τ_b $\quad\quad$ τ_{ab}

$$\begin{array}{ccc} q_0 \longrightarrow q_0 & & q_0 \longrightarrow q_0 \\ \searrow & & \\ q_1 \longrightarrow q_1 & \circ & q_1 \longrightarrow q_1 \\ & & \searrow \\ q_2 \longrightarrow q_2 & & q_2 \longrightarrow q_2 \end{array} \qquad \begin{array}{ccc} q_0 \longrightarrow q_0 \\ \\ q_1 \longrightarrow q_1 \\ \searrow \\ q_2 \longrightarrow q_2 \end{array} \qquad \begin{array}{ccc} q_0 \longrightarrow q_0 \\ \searrow \quad \searrow \\ q_1 \longrightarrow q_1 \\ \searrow \quad \searrow \\ q_2 \longrightarrow q_2 \end{array}$$

Transition monoid



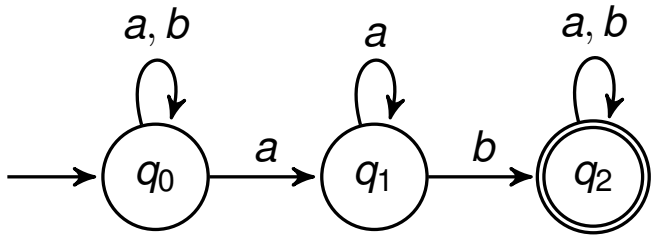
τ_a τ_b τ_{ab}

$$\begin{array}{ccc} q_0 \longrightarrow q_0 & & q_0 \longrightarrow q_0 \\ \searrow & & \\ q_1 \longrightarrow q_1 & \circ & q_1 \longrightarrow q_1 \\ & & \searrow \\ q_2 \longrightarrow q_2 & & q_2 \longrightarrow q_2 \end{array} \qquad \begin{array}{ccc} q_0 \longrightarrow q_0 & & q_0 \longrightarrow q_0 \\ & & \searrow \\ q_1 \longrightarrow q_1 & = & q_1 \longrightarrow q_1 \\ \searrow & & \\ q_2 \longrightarrow q_2 & & q_2 \longrightarrow q_2 \end{array}$$

Transition inhabitation problem

Given a binary relation τ' over Q, is there any $w \in \Sigma^*$ such that $\tau' = \tau_w$?

Transition monoid



τ_a τ_b τ_{ab}

$q_0 \longrightarrow q_0$ $q_0 \longrightarrow q_0$ $q_0 \longrightarrow q_0$

$q_1 \longrightarrow q_1$ $\circ$ $q_1 \longrightarrow q_1$ $q_1 \longrightarrow q_1$

$q_2 \longrightarrow q_2$ $q_2 \longrightarrow q_2$ $q_2 \longrightarrow q_2$

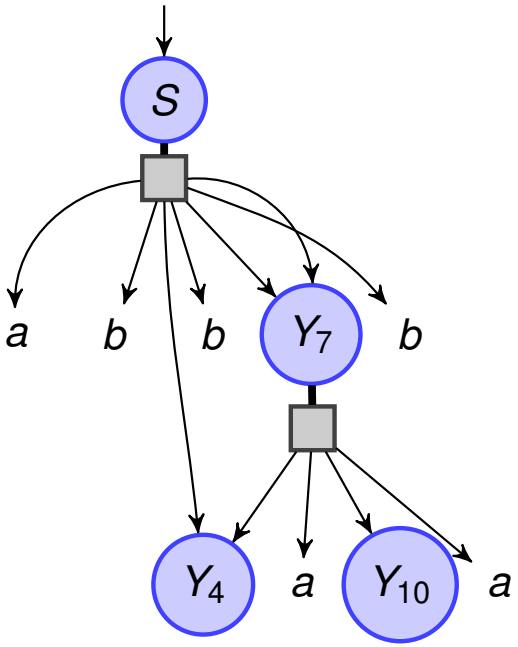
Transition inhabitation problem

Given a binary relation τ' over Q, is there any $w \in \Sigma^*$ such that $\tau' = \tau_w$?

Inhabitation problem is PSPACE-complete [Kozen 1977].

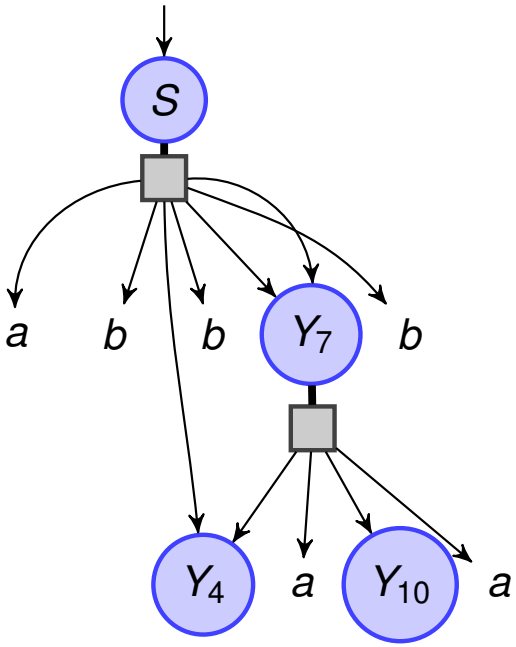
Decision procedure for regular compressed matching problem

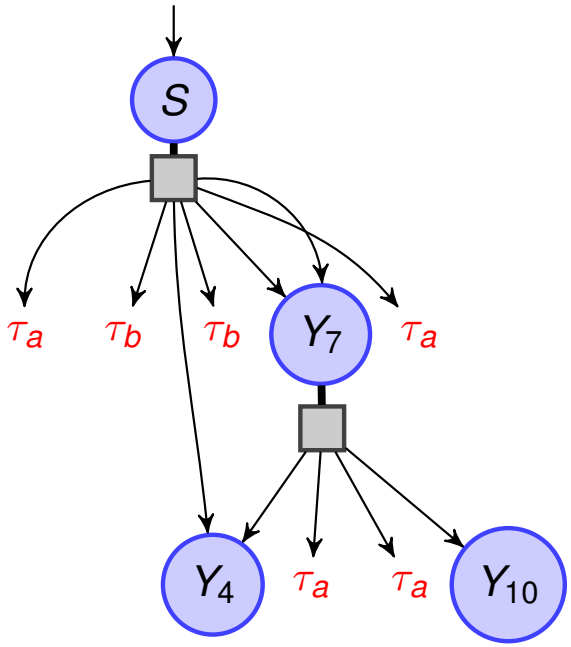
Compressed string pattern G and NFA $A = (Q, \Sigma, \delta, I, F)$.



Decision procedure for regular compressed matching problem

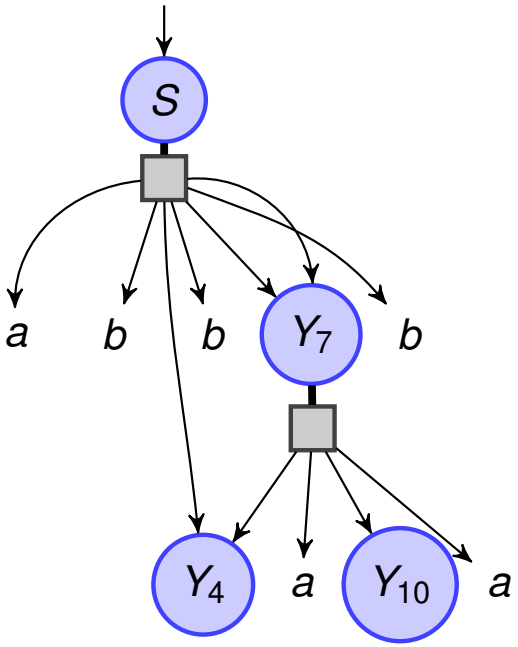
Compressed string pattern G and NFA $A = (Q, \Sigma, \delta, I, F)$.

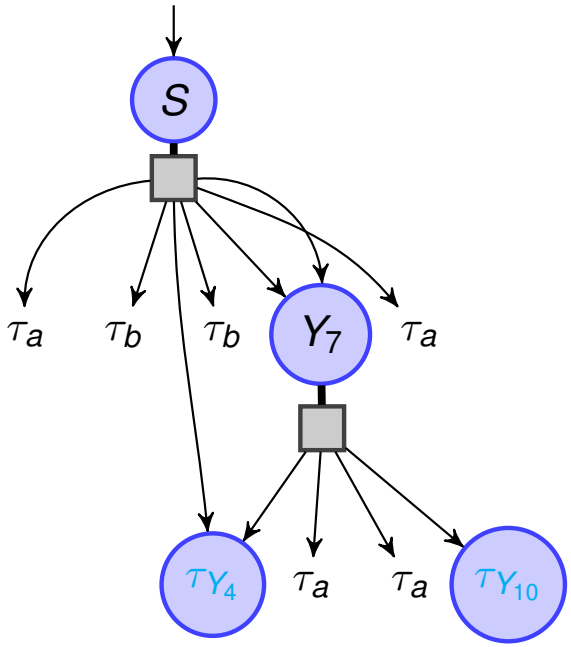




Decision procedure for regular compressed matching problem

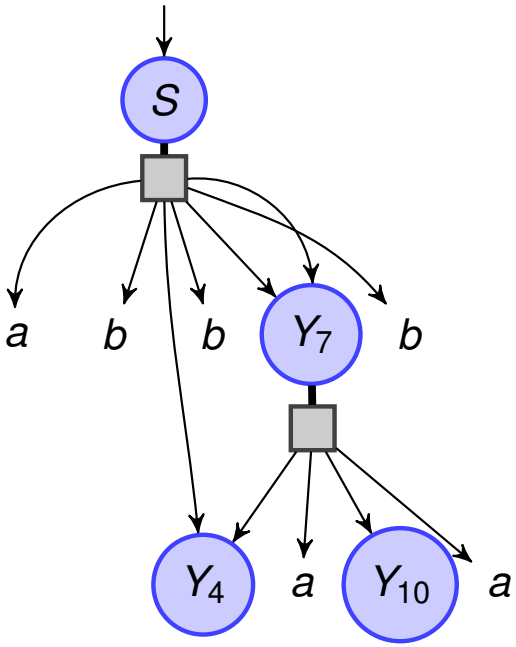
Compressed string pattern G and NFA $A = (Q, \Sigma, \delta, I, F)$.

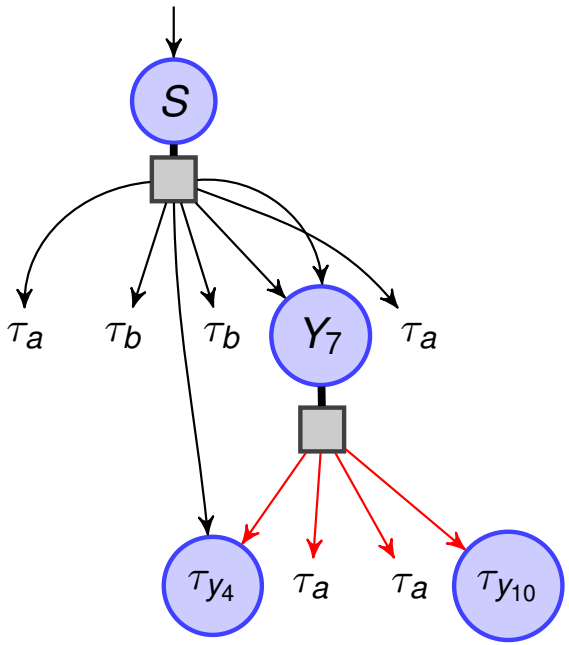




Decision procedure for regular compressed matching problem

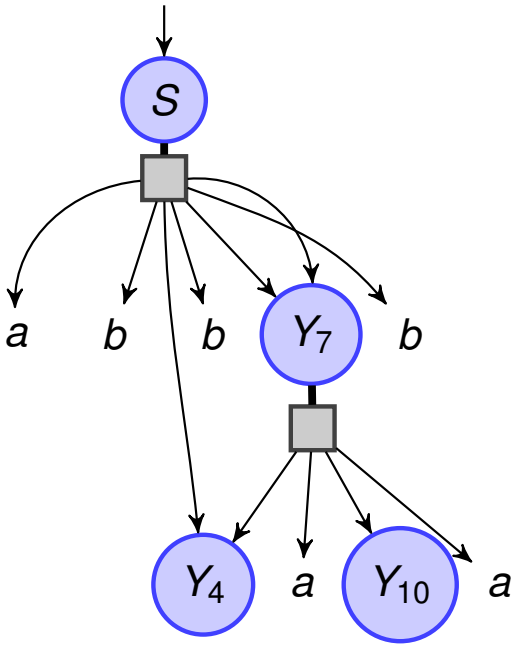
Compressed string pattern G and NFA $A = (Q, \Sigma, \delta, I, F)$.

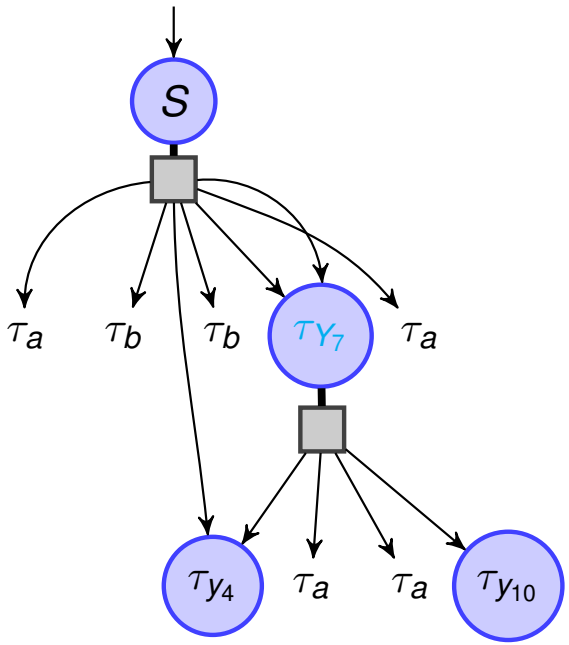




Decision procedure for regular compressed matching problem

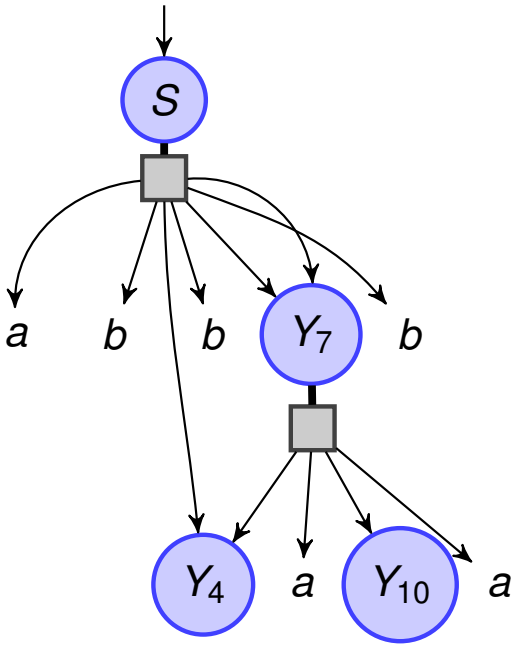
Compressed string pattern G and NFA $A = (Q, \Sigma, \delta, I, F)$.

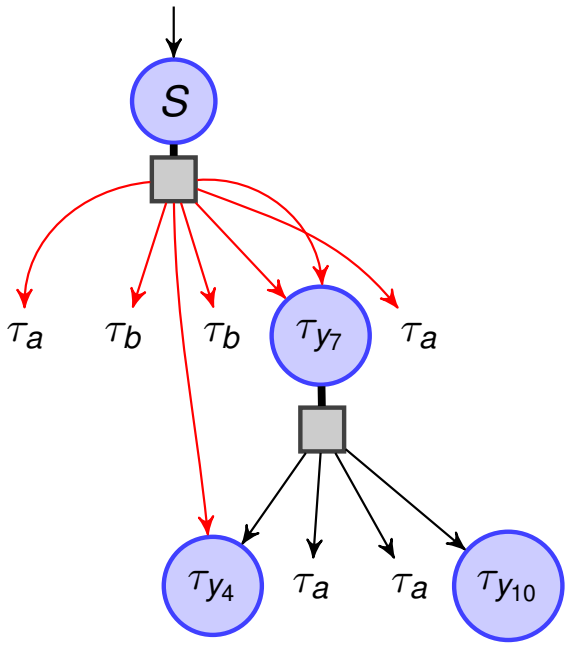




Decision procedure for regular compressed matching problem

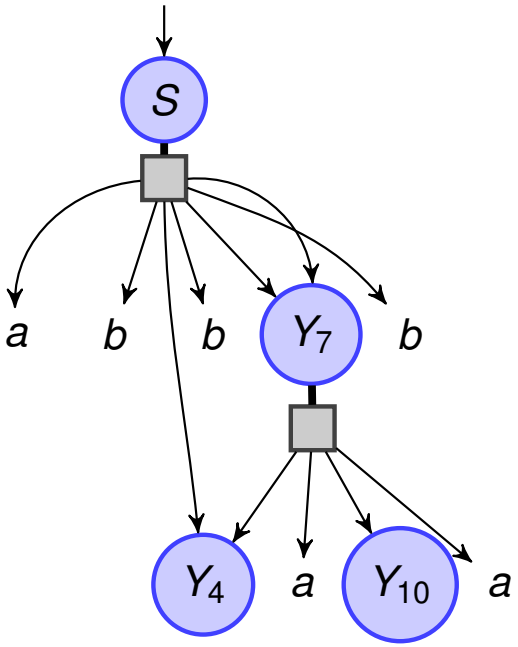
Compressed string pattern G and NFA $A = (Q, \Sigma, \delta, I, F)$.

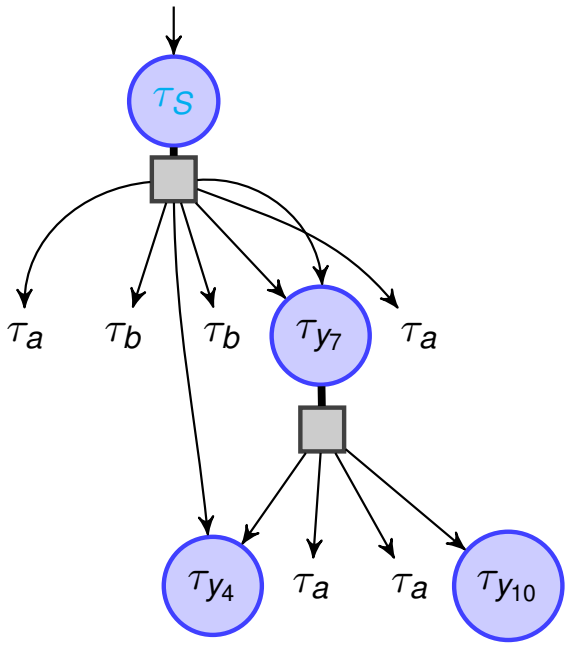




Decision procedure for regular compressed matching problem

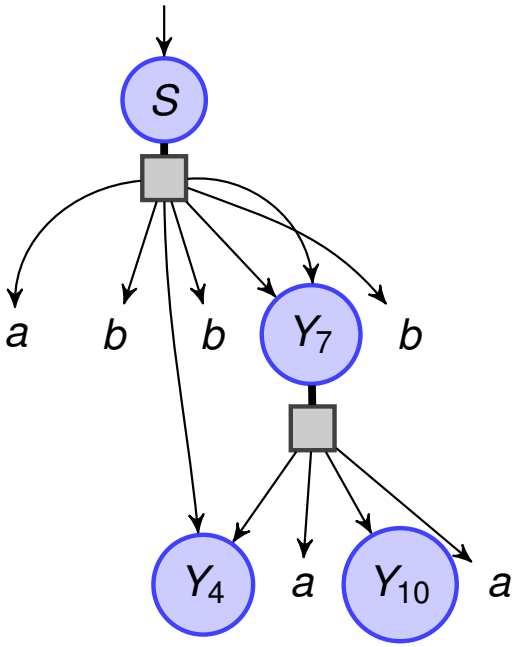
Compressed string pattern G and NFA $A = (Q, \Sigma, \delta, I, F)$.

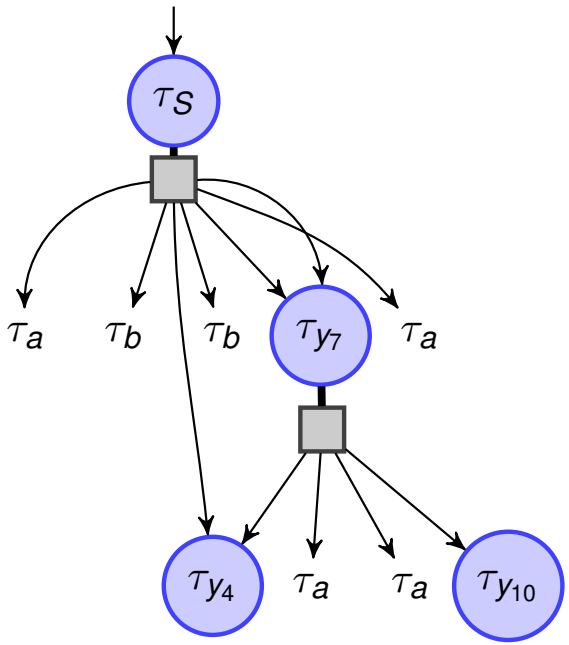




Decision procedure for regular compressed matching problem

Compressed string pattern G and NFA $A = (Q, \Sigma, \delta, I, F)$.

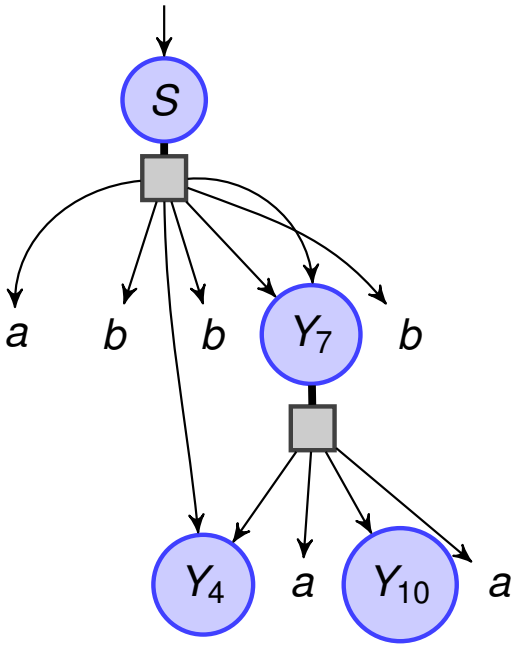


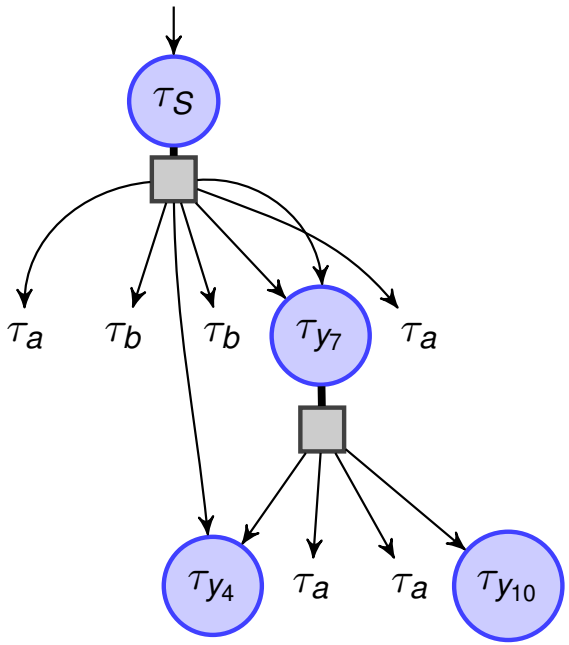


$$\tau_S \cap (I \times F) \neq \emptyset \ ?$$

Decision procedure for regular compressed matching problem

Compressed string pattern G and NFA $A = (Q, \Sigma, \delta, I, F)$.





$\tau_S \cap (I \times F) \neq \emptyset$?

Similar algorithm for Inclusion

Contribution (Complexity of CQA on **Hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-hard
Non-answers (*coREGMATCH*)	-	-

Contribution (Complexity of CQA on **linear hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-hard
Non-answers (*coREGMATCH*)	-	-

Contribution (Complexity of CQA on **Hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-c
Non-answers (*coREGMATCH*)	PSPACE-c	PSPACE-c

Contribution (Complexity of CQA on **linear hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-c
Non-answers (*coREGMATCH*)	-	-

Complexity of *REGINCL* for DFAs

Proposition

$REGINCL(\_, \text{DFA}s) = coREGMATCH(\_, \text{DFA}s)$

Complexity of *REGINCL* for DFAs

Proposition

$REGINCL(\_, \text{DFA}s) = coREGMATCH(\_, \text{DFA}s)$

Proof idea

For hyperstream G and NFA A, $Inst(G) \subseteq L(A)$ iff $Inst(G) \cap L(\bar{A}) = \emptyset$.
PTIME complementation for DFAs.

Contribution (Complexity of CQA on **Hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-c
Non-answers (*coREGMATCH*)	PSPACE-c	PSPACE-c

Contribution (Complexity of CQA on **linear hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-c
Non-answers (*coREGMATCH*)	-	-

Contribution (Complexity of CQA on **Hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	PSPACE-c	PSPACE-c
Non-answers (*coREGMATCH*)	PSPACE-c	PSPACE-c

Contribution (Complexity of CQA on **linear hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-c
Non-answers (*coREGMATCH*)	-	-

An efficient fragment

An efficient fragment

Theorem

REGMATCH(*Linear_String_Patterns*, NFA*s*) *is in* PTIME.

An efficient fragment

Theorem

REGMATCH(*Linear_String_Patterns*, NFA*s*) *is in* PTIME.

Proof idea

Evaluation with the transition monoid, using the accessibility relation of the automaton as transition for variables.

An efficient fragment

Theorem

REGMATCH(*Linear_String_Patterns*, NFA*s*) *is in* PTIME.

Proof idea

Evaluation with the transition monoid, using the accessibility relation of the automaton as transition for variables.

Corollary

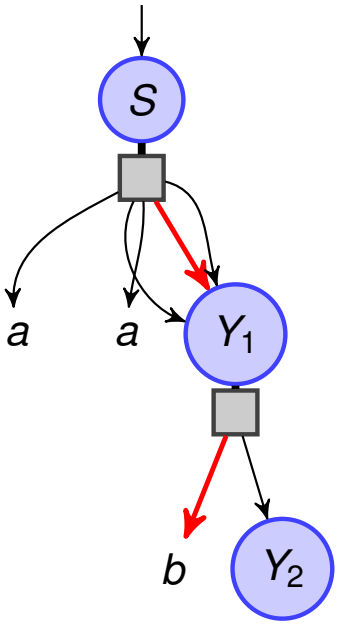
REGINCL(*Linear_String_Patterns*, DFA*s*) *is in* PTIME.

When the query is not boolean

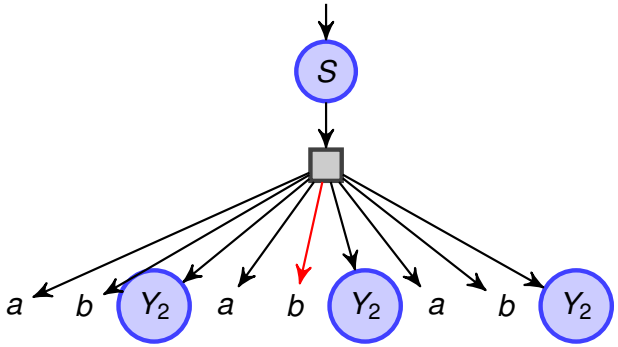
Theorem

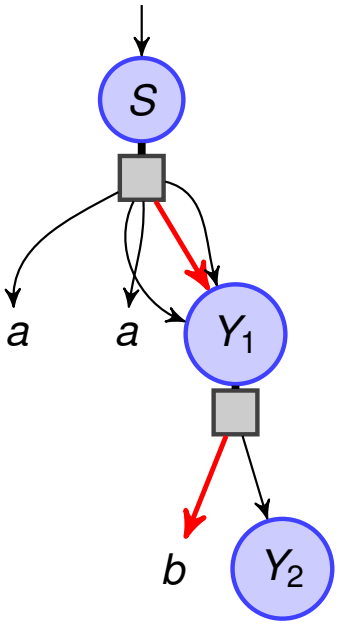
CQA(*Linear_Hyperstreams*, DFA*s*) *is in* PTIME.

Example of partial decompression

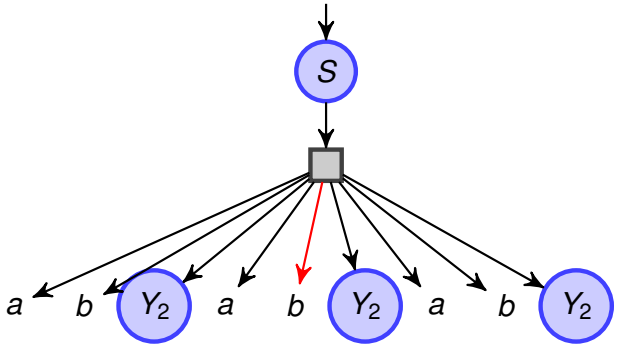


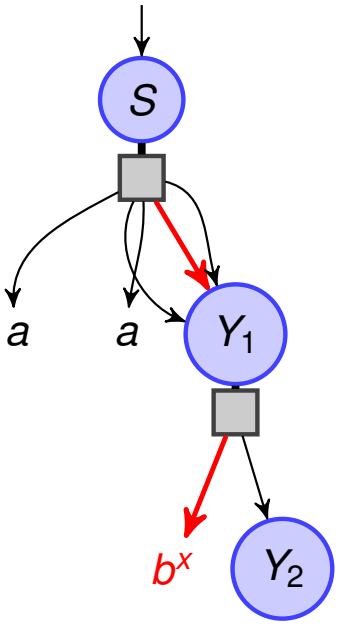
Example of partial decompression



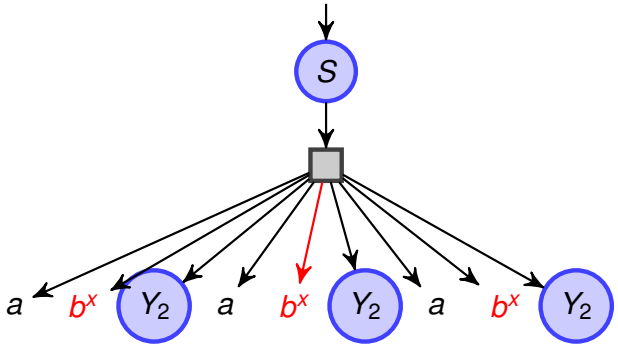


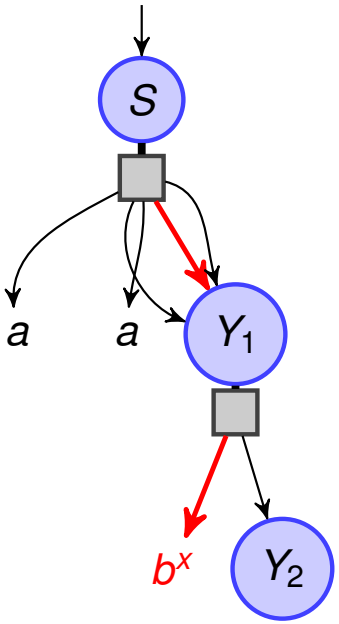
Example of partial decompression



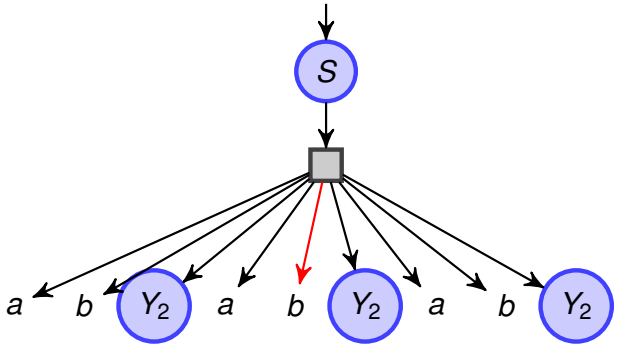


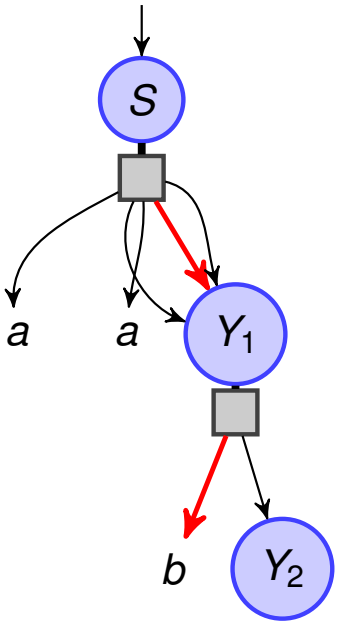
Example of partial decompression

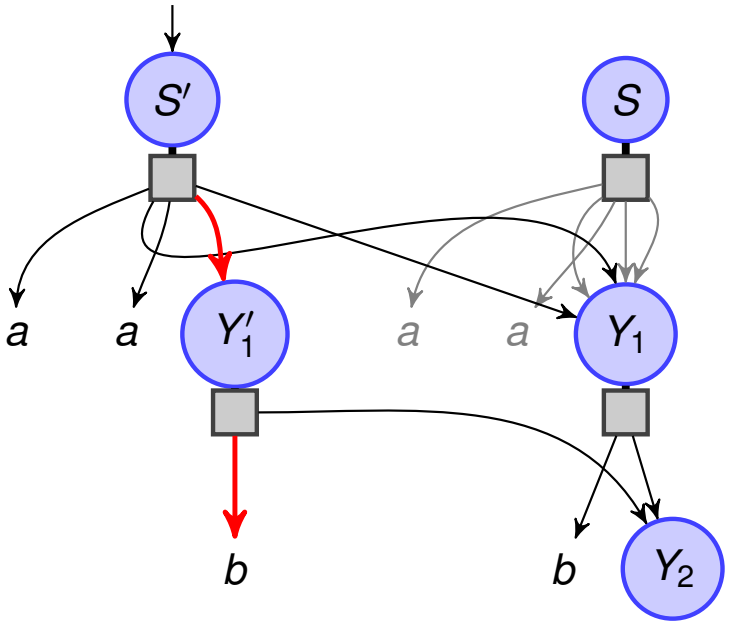




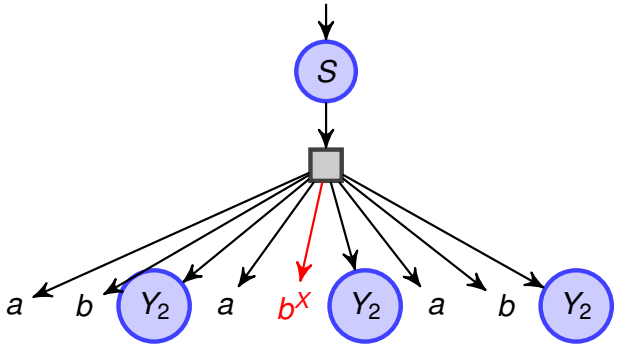
Example of partial decompression

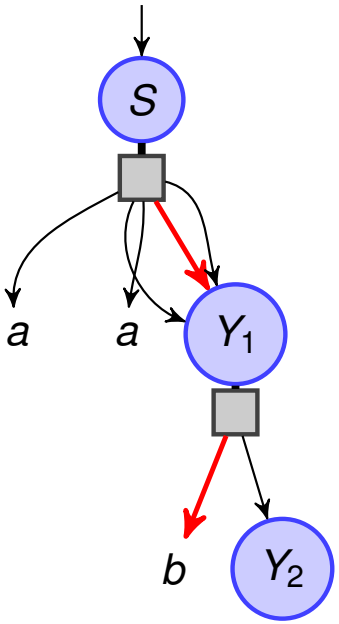


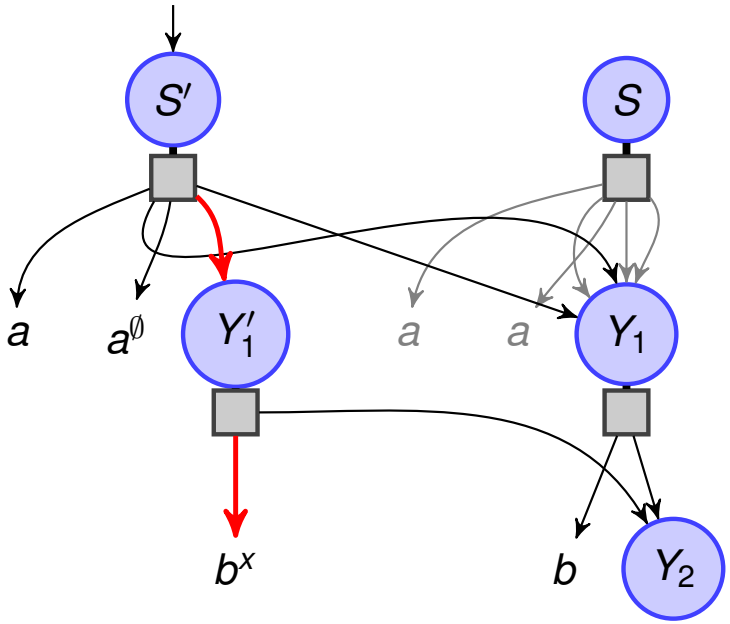




Example of partial decompression







Contribution (Complexity of CQA on **Hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	PSPACE-c	PSPACE-c
Non-answers (*coREGMATCH*)	PSPACE-c	PSPACE-c

Contribution (Complexity of CQA on **linear hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	-	PSPACE-c
Non-answers (*coREGMATCH*)	-	-

Contribution (Complexity of CQA on **Hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	PSPACE-c	PSPACE-c
Non-answers (*coREGMATCH*)	PSPACE-c	PSPACE-c

Contribution (Complexity of CQA on **linear hyperstreams**)

	DFAs	NFAs
Answers (*REGINCL*)	PTIME	PSPACE-c
Non-answers (*coREGMATCH*)	PTIME	PTIME

Outline

Future Work

- Short and middle term
  - Tractable approximations of Certain query answering on NFAs
  - Practical hyperstreaming algorithms
  - Extension to Trees

Future Work

- Short and middle term
  - Tractable approximations of Certain query answering on NFAs
  - Practical hyperstreaming algorithms
  - Extension to Trees
- Long term
  - Extension to transformations
  - Extension to Graphs