Automated generation of BSP Automata

Thibaut Tachon

Gaétan Hains
Supervisor in Huawei

&

Frédéric Loulergue
Academic supervisor

Frebrary 15th, 2017

Content

Regular language

Regular Expressions

$$
\begin{array}{rcl}
r & ::= & r\ r' \\
 & | & r + r' \\
 & | & r^* \\
 & | & a \\
 & | & \epsilon \\
 & | & \emptyset
\end{array}
$$

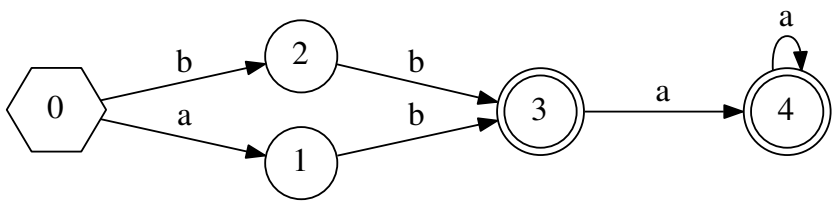
Finite Automata

$$A = (Q, \Sigma, \delta, I, F)$$

Q	State set
Σ	Alphabet
δ	Transitions
I	Initial states
F	Final states

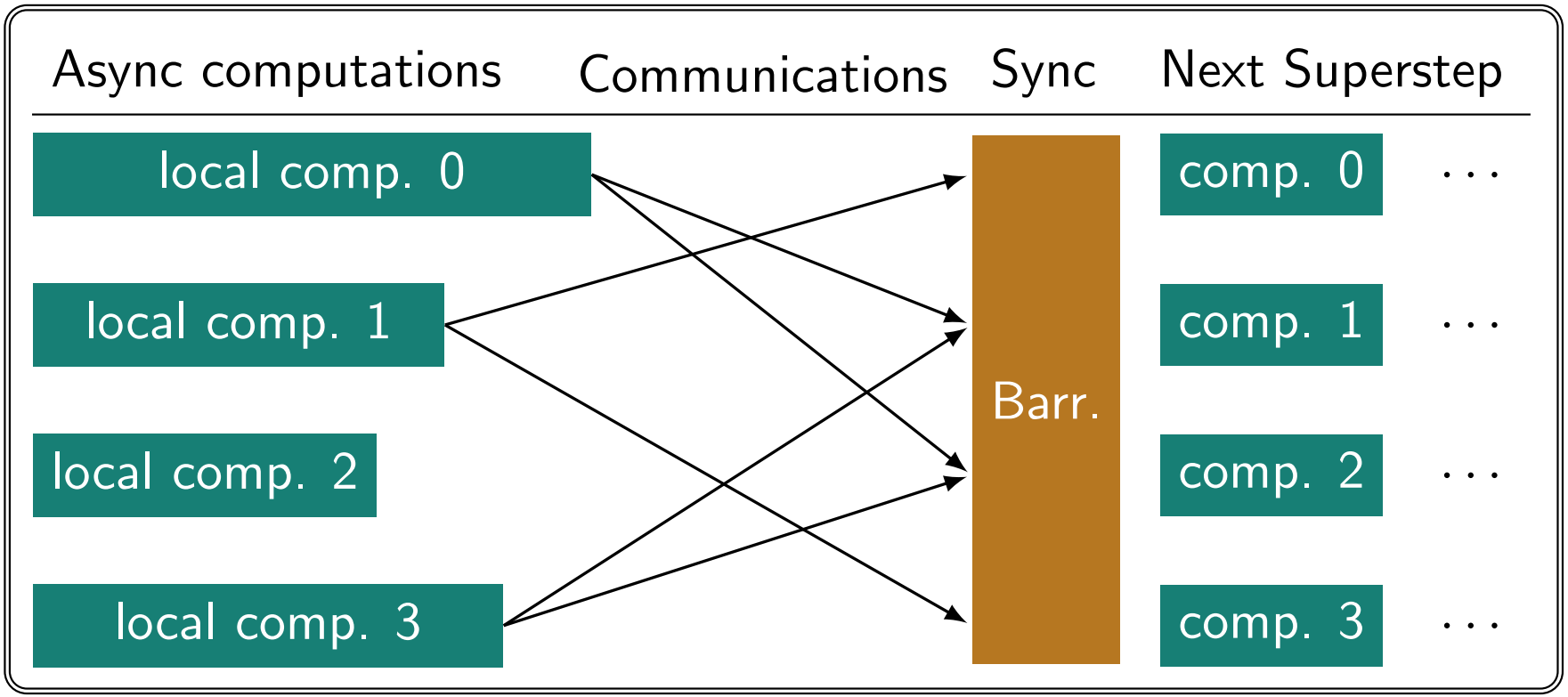
Brüggemann-Klein

$$r = (a + b)ba^* \quad \Longrightarrow$$



Bulk-Synchronous Parallel model

- Superstep sequence
- Local determinism $\rightarrow$ global determinism
- Neither deadlocks nor data races
- Simple cost model



BSP Language

Regular language

	Type	Example
Symbol :	Σ	a
Word :	Σ^*	$abccd$
Language :	$\mathcal{P}(\Sigma^*)$	$\{a, aab, ab\}$

BSP words and language

p processors

	Type	Example
Word vector :	$(\Sigma^*)^p$	$\langle ab, a, bbb \rangle$
BSP word :	$((\Sigma^*)^p)^*$	$\langle ab, a \rangle \langle bb, aba \rangle$
BSP Language :	$\mathcal{P}(((\Sigma^*)^p)^*)$	$\{\langle ab, a \rangle, \langle ab, ba \rangle \langle a, bbb \rangle\}$

BSP Regular Expressions

Grammar

$$R ::= R\ R' \mid R + R' \mid R^* \mid \emptyset \mid \epsilon \mid \langle r^0, \ldots, r^{p-1} \rangle$$

Language

r	$L(r)$
$r\ r'$	$L(r) \cdot L(r')$
$r + r'$	$L(r) \cup L(r')$
r^*	$\bigcup_{i=0}^{\infty} L^i(r)$
a	$\{a\}$
ϵ	$\{\epsilon\}$
$\emptyset$	$\{\}$

R	$L(R)$
$R\ R'$	$L(R) \cdot L(R')$
$R + R'$	$L(R) \cup L(R')$
R^*	$\bigcup_{i=0}^{\infty} L^i(R)$
$\langle r_0, \ldots, r_{p-1} \rangle$	$L(r_0) \times \cdots \times L(r_{p-1})$
ϵ	$\{\epsilon\}$
$\emptyset$	$\{\}$

BSP Automata : Definition

$$A = (\{Q^i\}_{i\in[p]}, \Sigma, \{\delta^i\}_{i\in[p]}, \{I^i\}_{i\in[p]}, \{F^i\}_{i\in[p]}, \Delta)$$

- p sized automata vector
- Common alphabet
- Synchronization function

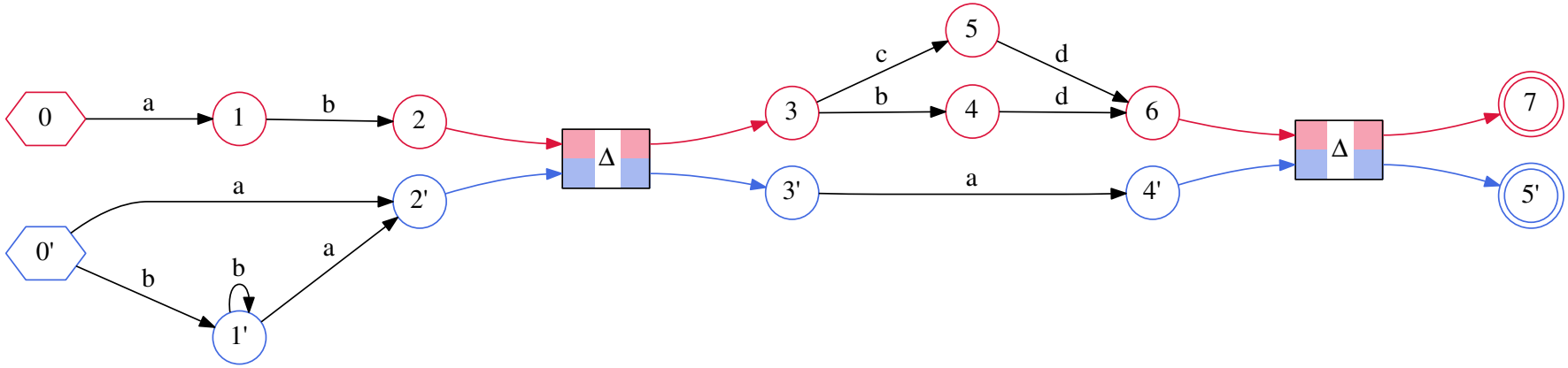
Delta

- $\Delta : Q^p \rightarrow Q^p$
- $Q^p = (Q^0 \times ... \times Q^{p-1})$

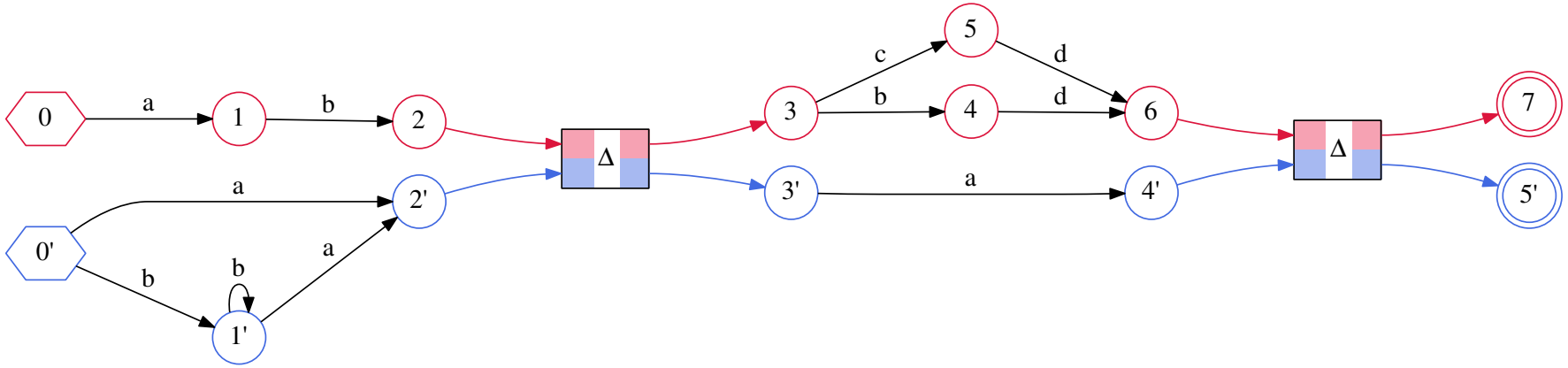
Notation

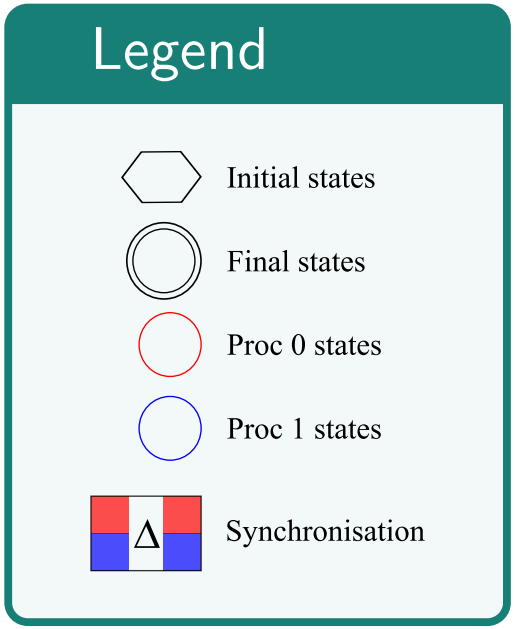
- p processors
- $[p] = \{0 \ldots p-1\}$

BSP Automata: Example ($p = 2$)

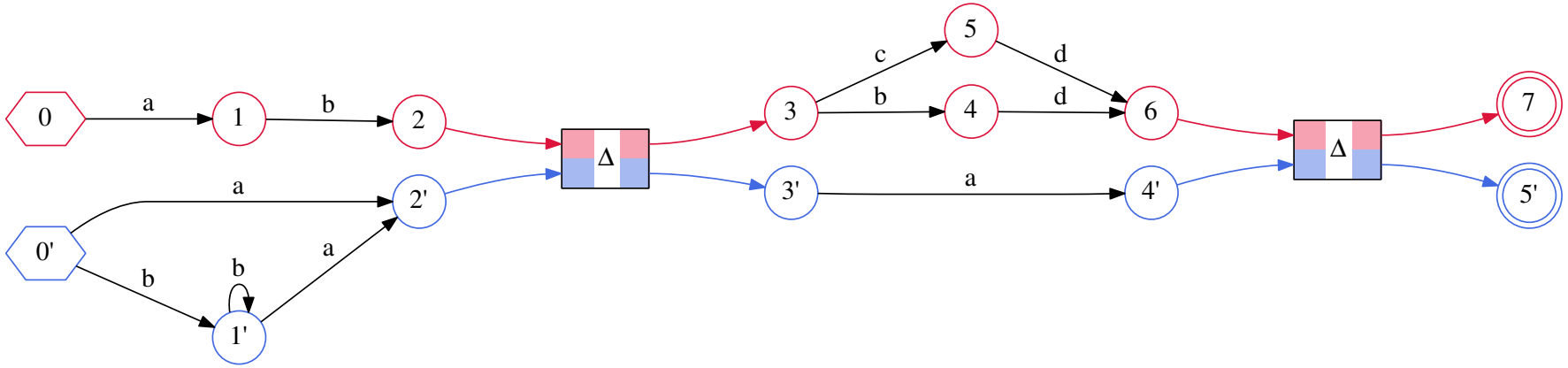


BSP Automata: Example ($p = 2$)

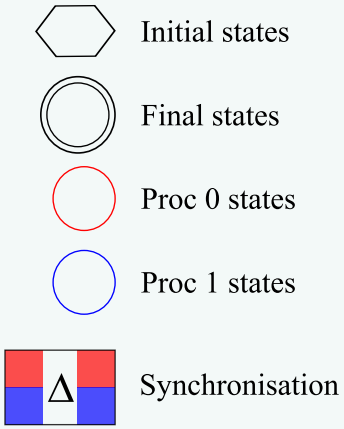




BSP Automata: Example ($p = 2$)



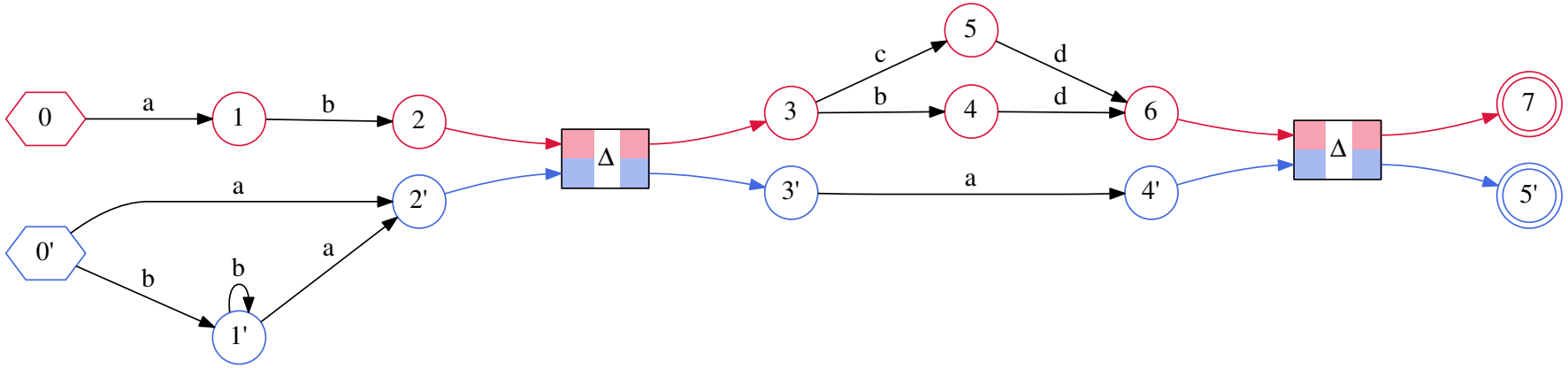
Legend



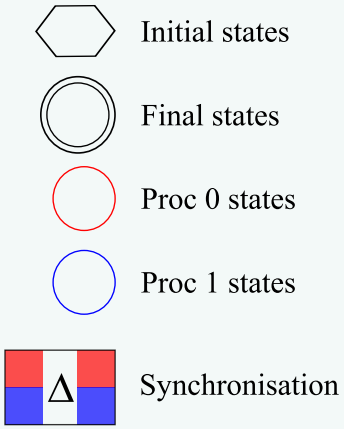
Delta

$$\Delta = \{ \ \langle 2, 2' \rangle \rightarrow \langle 3, 3' \rangle \ , \\ \langle 6, 4' \rangle \rightarrow \langle 7, 5' \rangle \}$$

BSP Automata: Example ($p = 2$)



Legend



Delta

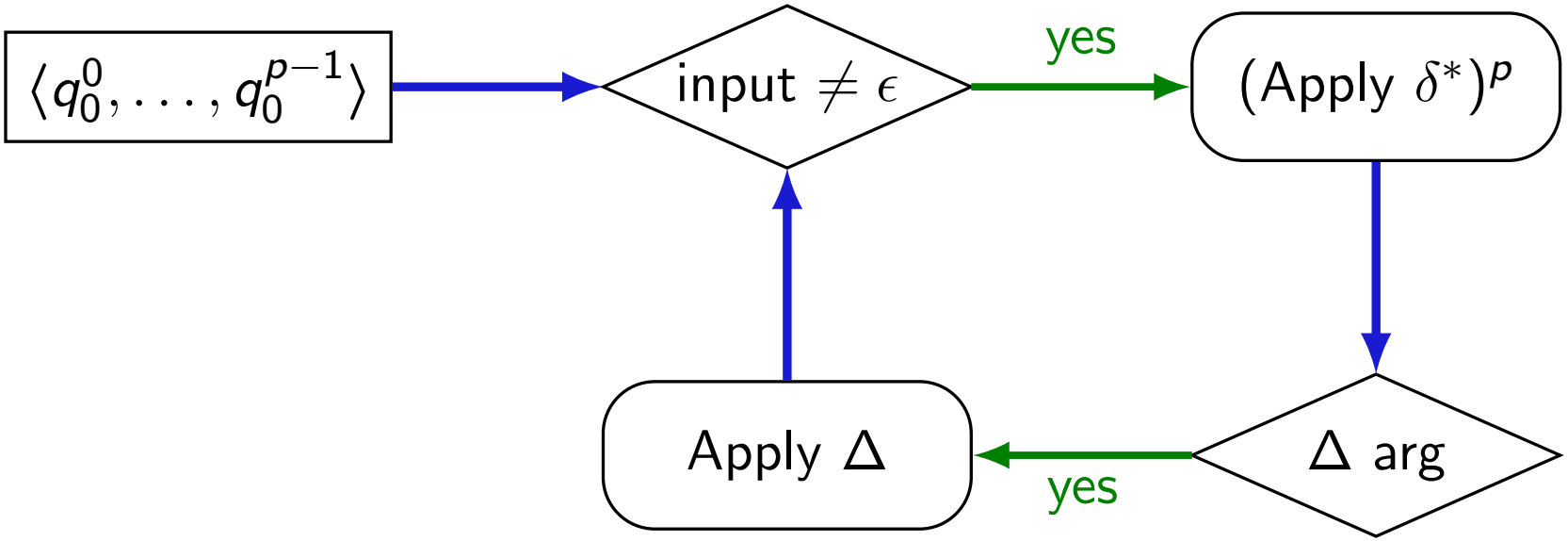
$$\Delta = \{ \ \langle 2, 2' \rangle \rightarrow \langle 3, 3' \rangle \ , \\ \langle 6, 4' \rangle \rightarrow \langle 7, 5' \rangle \}$$

Accept : $\langle ab, a \rangle \langle bd, a \rangle$?

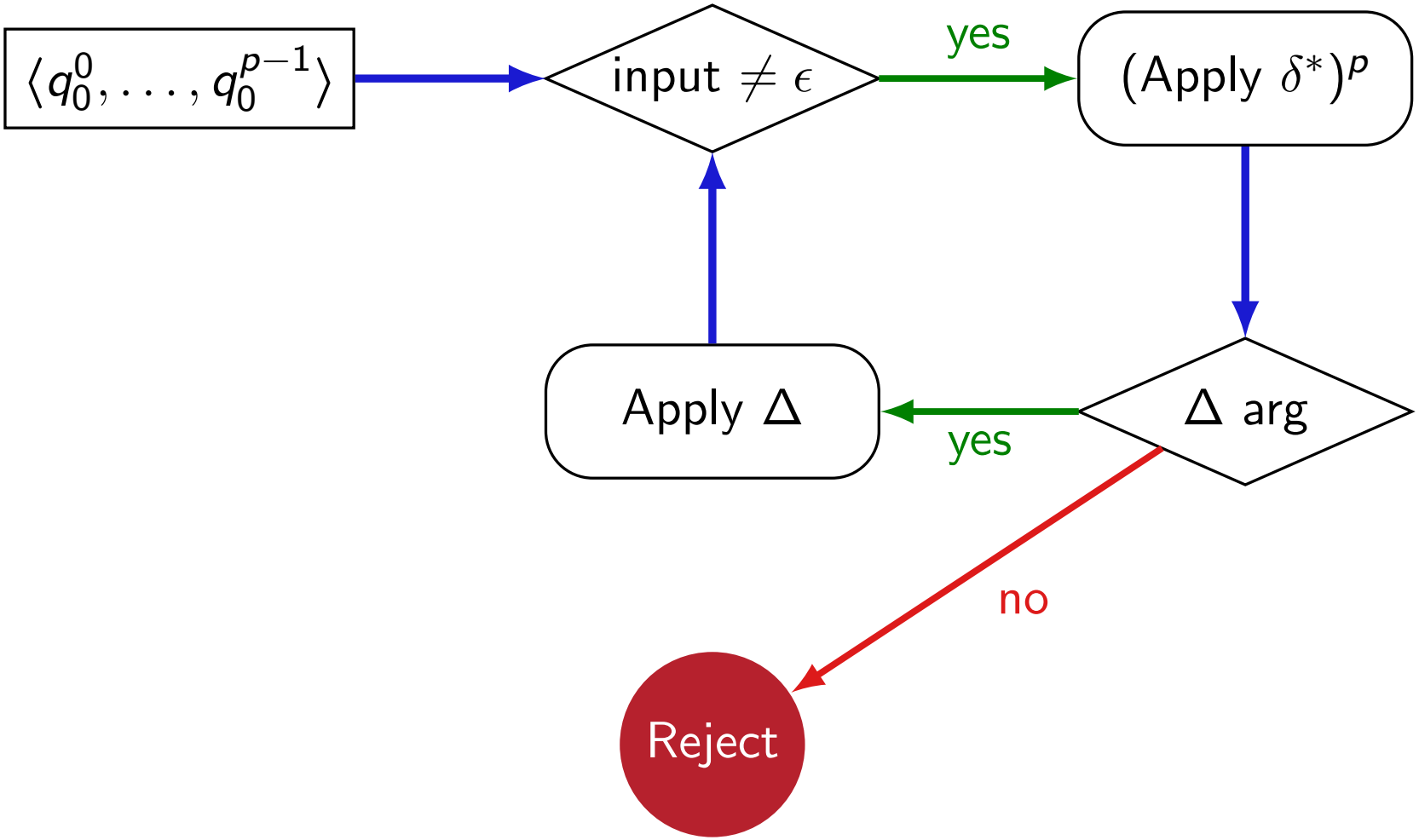
BSP Automata : Recognition

$$\boxed{\langle q_0^0, \ldots, q_0^{p-1} \rangle}$$

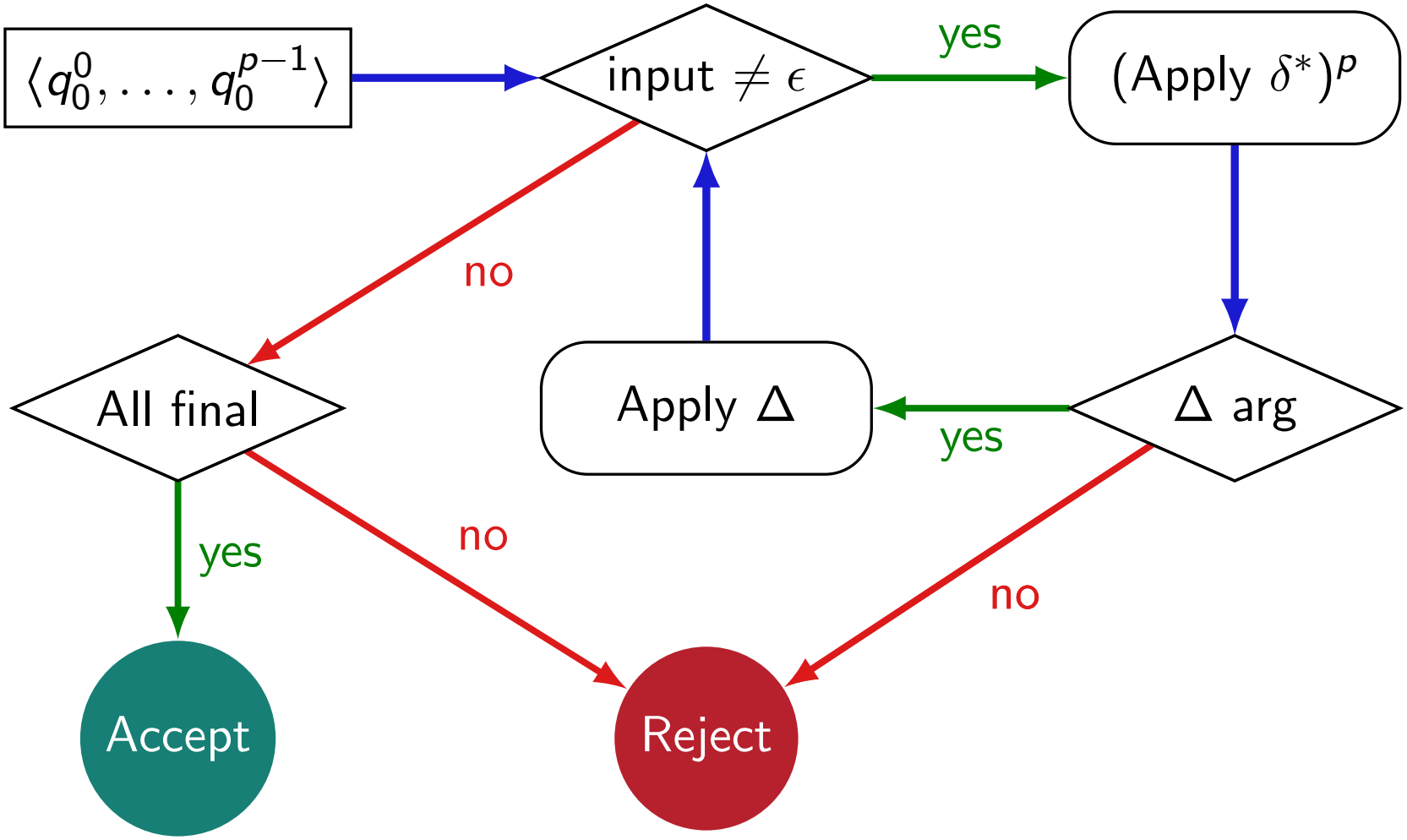
BSP Automata : Recognition



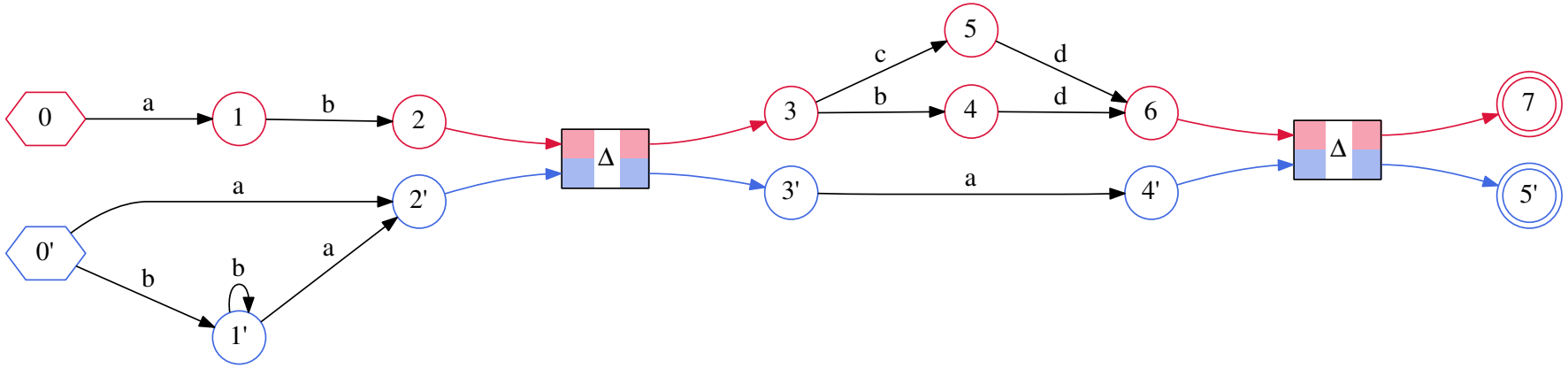
BSP Automata : Recognition



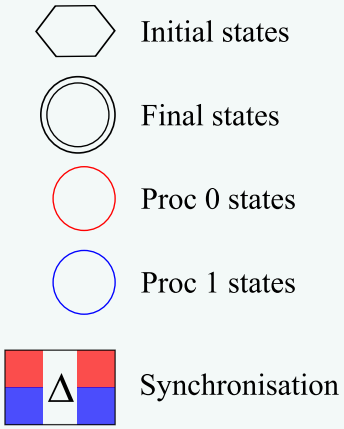
BSP Automata : Recognition



BSP Automata: Example ($p = 2$)



Legend



Delta

$$\Delta = \{ \ \langle 2, 2' \rangle \rightarrow \langle 3, 3' \rangle \ , \\ \langle 6, 4' \rangle \rightarrow \langle 7, 5' \rangle \}$$

Accept : $\langle ab, a \rangle \langle bd, a \rangle$?

Model

Regular language

	Type	model
Symbol :	Σ	an instruction
Word :	Σ^*	a program trace
Language :	$\mathcal{P}(\Sigma^*)$	all possible traces of a program

BSP words and language

p processors

	Type	model
Word vector :	$(\Sigma^*)^p$	one superstep
BSP word :	$((\Sigma^*)^p)^*$	a BSP program trace
BSP Language :	$\mathcal{P}(((\Sigma^*)^p)^*)$	all possible program traces

Query language

Which target

- BSP program trace
- BSP code (well structured)

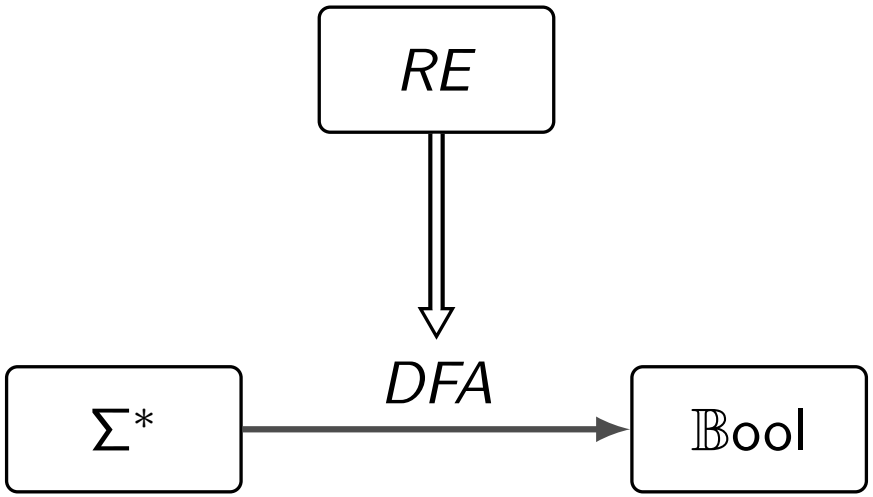
DSL for queries

- based on BSPRE
- computed by a BSPA

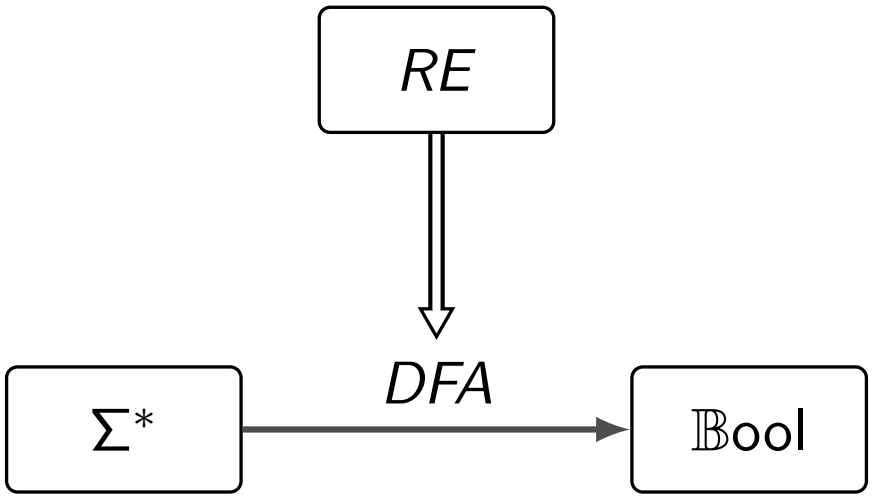
Limitations

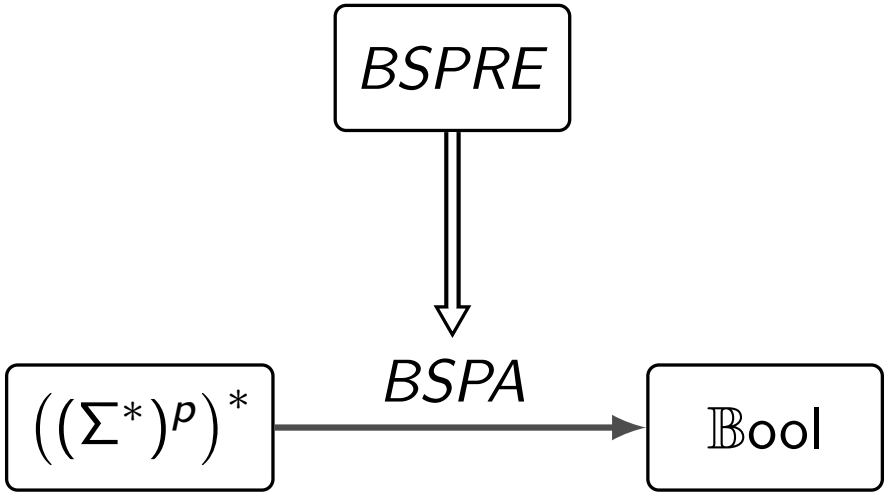
- regular expressions expressiveness
- output is a boolean

Parallel recognition of RE

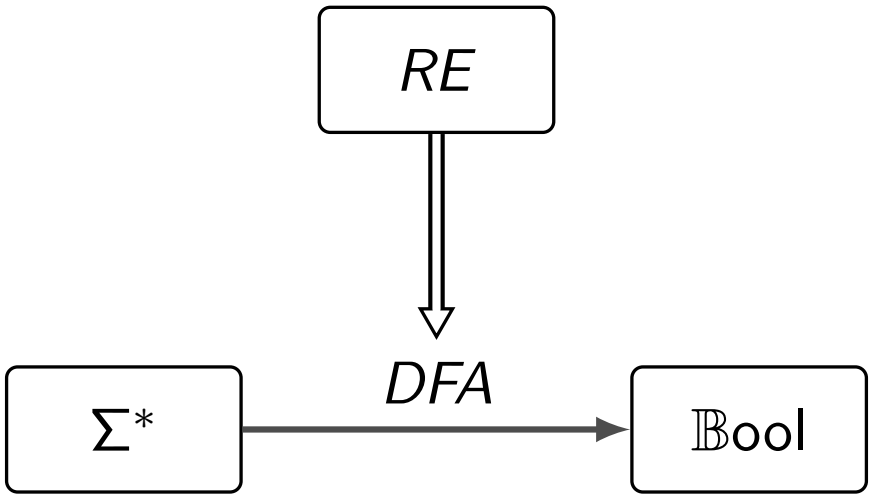


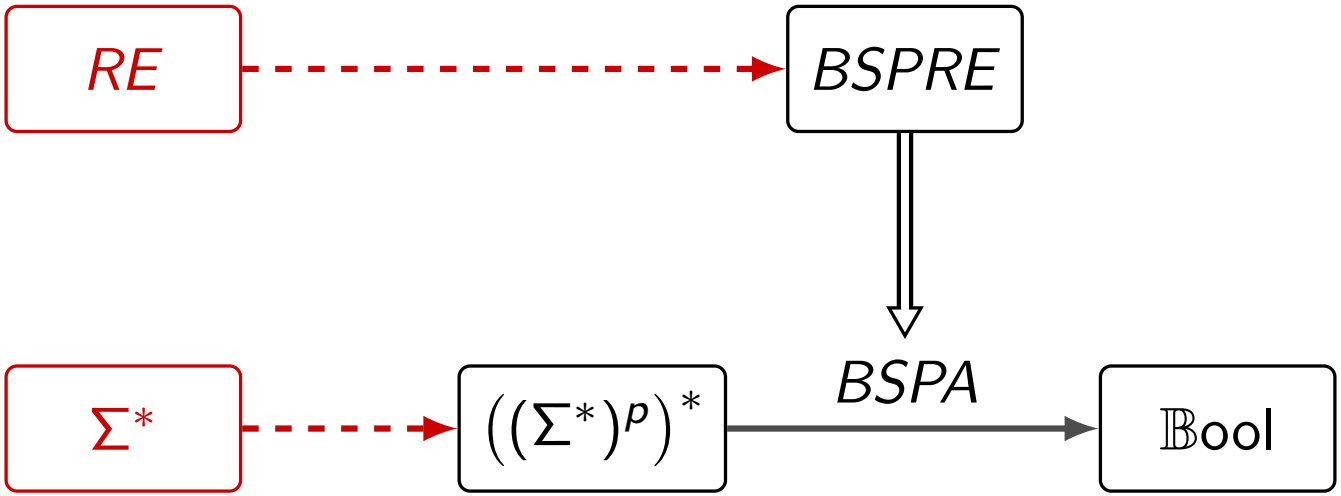
Parallel recognition of RE





Parallel recognition of RE





From RE to BSPRE

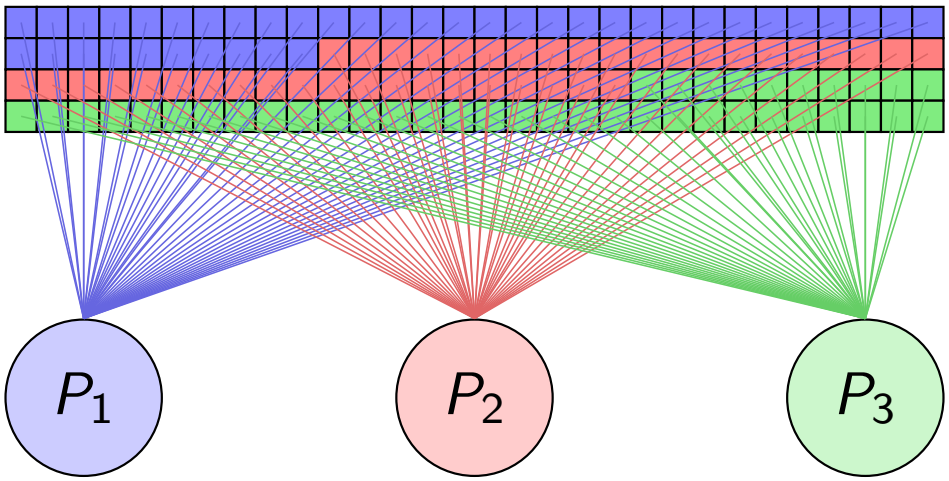
String research

$r = \sigma^* \texttt{ string } \sigma^*$ where $r \in RE$

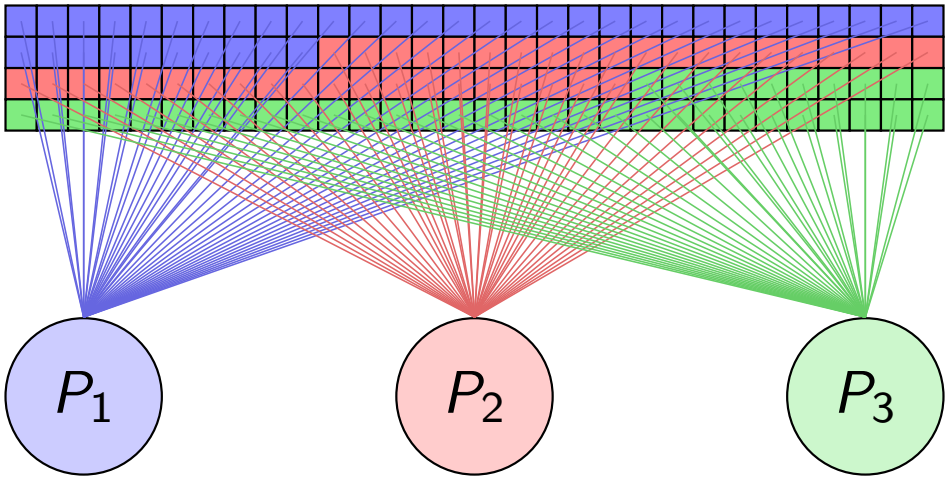
$\sigma = \forall \sigma_i \in \Sigma, (\sigma_0 + \sigma_1 + \dots)$

Distribution of input

Block



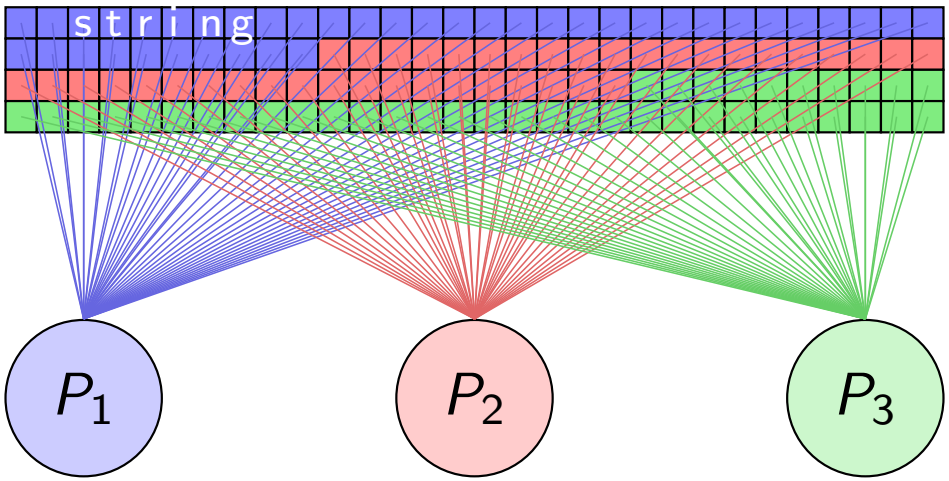
Block distribution



$R \in BSPRE$

$$R \stackrel{?}{=} \begin{cases} \langle \sigma^* , \sigma^* , \sigma^* \ \texttt{string} \ \sigma^* \rangle \\ + \langle \sigma^* , \sigma^* \ \texttt{string} \ \sigma^* , \sigma^* \rangle \\ + \langle \sigma^* \ \texttt{string} \ \sigma^* , \sigma^* , \sigma^* \rangle \end{cases}$$

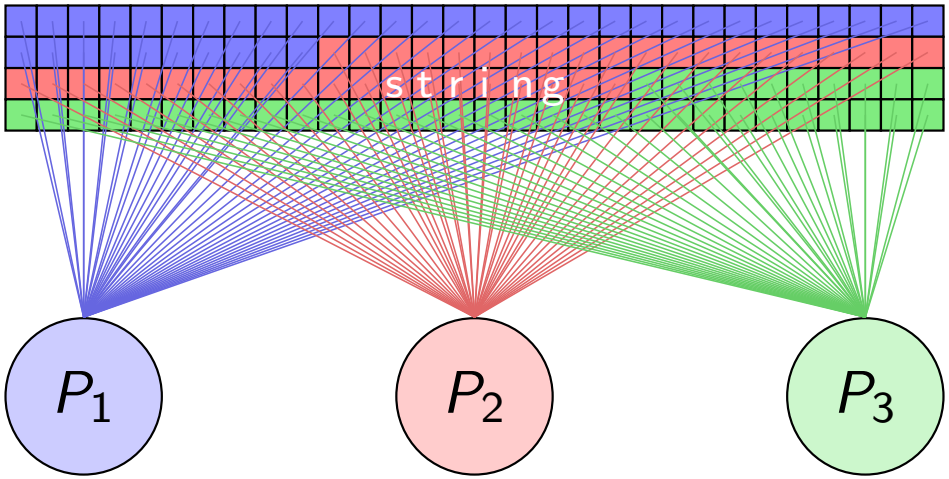
Block distribution



$R \in BSPRE$

$$R \stackrel{?}{=} \begin{cases} \langle \sigma^* , \sigma^* , \sigma^* \texttt{ string } \sigma^* \rangle \\ + \langle \sigma^* , \sigma^* \texttt{ string } \sigma^* , \sigma^* \rangle \\ + \langle \sigma^* \texttt{ string } \sigma^* , \sigma^* , \sigma^* \rangle \end{cases}$$

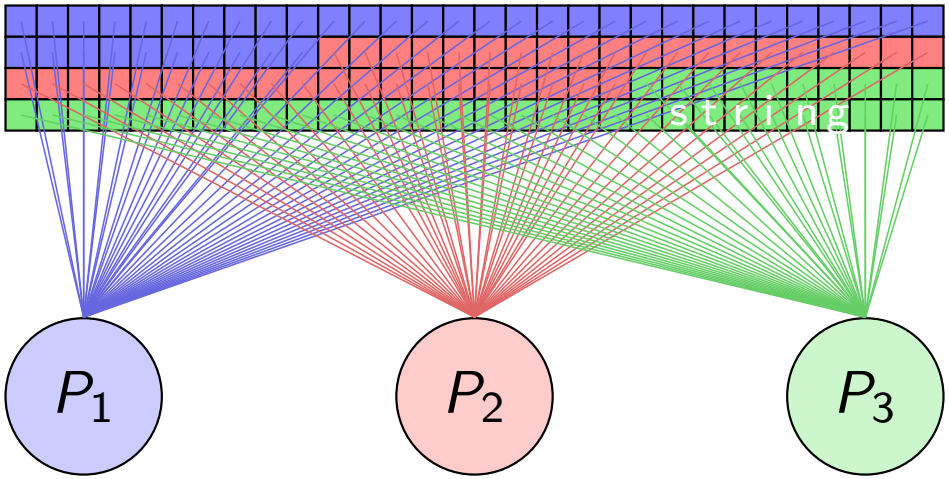
Block distribution



$R \in BSPRE$

$$R \stackrel{?}{=} \begin{cases} \langle \sigma^* , \sigma^* , \sigma^* \ \texttt{string}\ \sigma^* \rangle \\ + \langle \sigma^* , \sigma^* \ \texttt{string}\ \sigma^* , \sigma^* \rangle \\ + \langle \sigma^* \ \texttt{string}\ \sigma^* , \sigma^* , \sigma^* \rangle \end{cases}$$

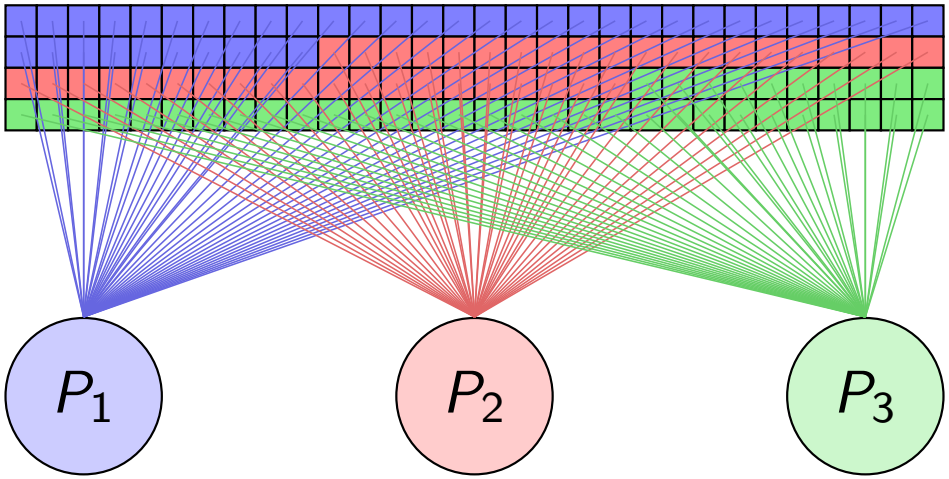
Block distribution



$R \in BSPRE$

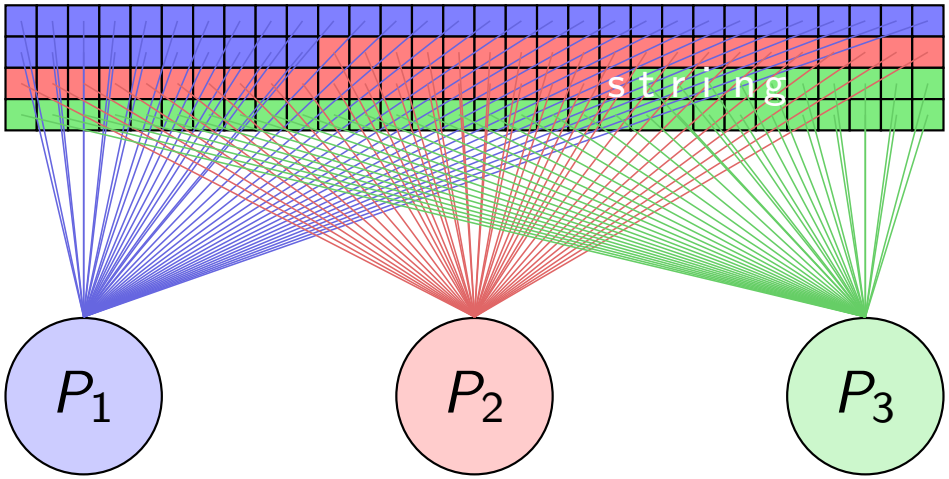
$$R \stackrel{?}{=} \begin{cases} \langle \sigma^* , \sigma^* , \sigma^* \texttt{ string } \sigma^* \rangle \\ + \langle \sigma^* , \sigma^* \texttt{ string } \sigma^* , \sigma^* \rangle \\ + \langle \sigma^* \texttt{ string } \sigma^* , \sigma^* , \sigma^* \rangle \end{cases}$$

Block distribution



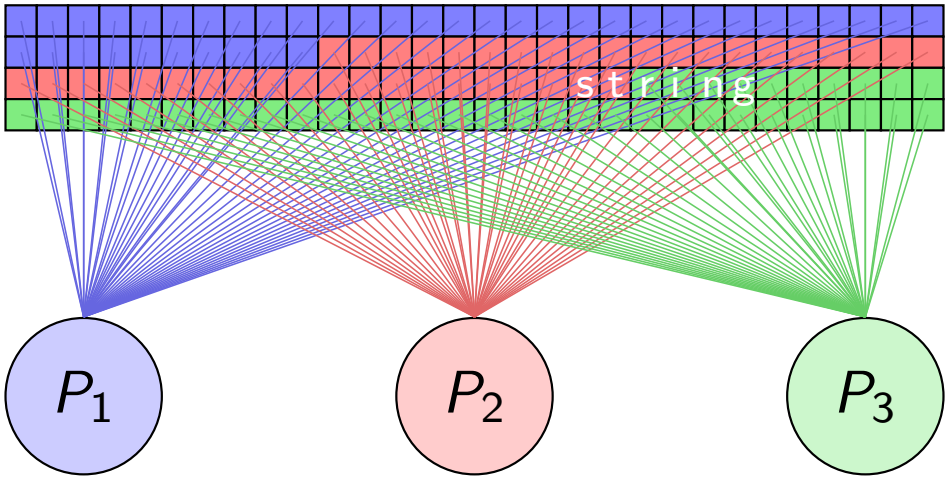
$$R = \begin{cases} \langle \sigma^* , \sigma^* , \sigma^* \texttt{string} \sigma^* \rangle \\ + \langle \sigma^* , \sigma^* \texttt{s} , \texttt{tring} \sigma^* \rangle \\ + \langle \sigma^* , \sigma^* \texttt{st} , \texttt{ring} \sigma^* \rangle \\ \vdots \\ + \langle \sigma^* , \sigma^* \texttt{string} \sigma^* , \sigma^* \rangle \\ \vdots \\ + \langle \sigma^* \texttt{string} \sigma^* , \sigma^* , \sigma^* \rangle \end{cases}$$

Block distribution



$$
R = \begin{cases}
\langle\ \sigma^* \ ,\ \sigma^* \ ,\ \sigma^*\ \texttt{string}\ \sigma^*\ \rangle \\
+\langle\ \sigma^* \ ,\ \sigma^* \ \texttt{s}\ ,\ \texttt{tring}\ \sigma^*\ \rangle \\
+\langle\ \sigma^* \ ,\ \sigma^* \ \texttt{st}\ ,\ \texttt{ring}\ \sigma^*\ \rangle \\
\qquad\qquad\vdots \\
+\langle\ \sigma^* \ ,\ \sigma^*\ \texttt{string}\ \sigma^*\ ,\ \sigma^*\ \rangle \\
\qquad\qquad\vdots \\
+\langle\ \sigma^*\ \texttt{string}\ \sigma^*\ ,\ \sigma^* \ ,\ \sigma^*\ \rangle
\end{cases}
$$

Block distribution



$$
R = \begin{cases}
\langle\ \sigma^*\ ,\ \sigma^*\ ,\ \sigma^*\ \texttt{string}\ \sigma^*\ \rangle \\
+\langle\ \sigma^*\ ,\ \sigma^*\ \texttt{s}\ ,\ \texttt{tring}\ \sigma^*\ \rangle \\
+\langle\ \sigma^*\ ,\ \sigma^*\ \texttt{st}\ ,\ \texttt{ring}\ \sigma^*\ \rangle \\
\qquad\qquad\vdots \\
+\langle\ \sigma^*\ ,\ \sigma^*\ \texttt{string}\ \sigma^*\ ,\ \sigma^*\ \rangle \\
\qquad\qquad\vdots \\
+\langle\ \sigma^*\ \texttt{string}\ \sigma^*\ ,\ \sigma^*\ ,\ \sigma^*\ \rangle
\end{cases}
$$

Block distribution

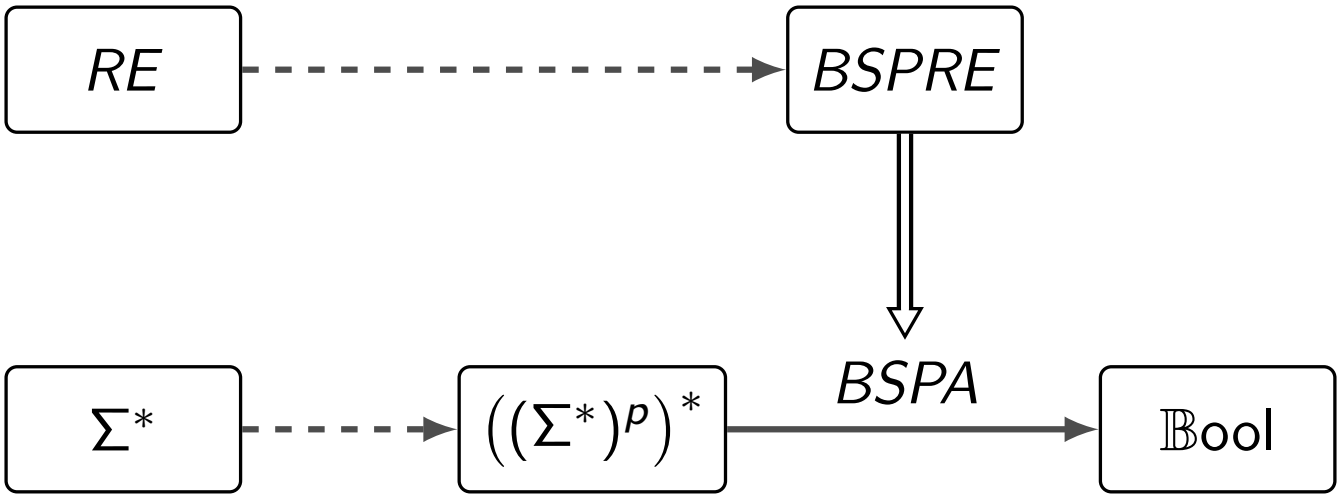
let $m = |string|$

$$R = \begin{cases} \langle \sigma^* , \sigma^* , \sigma^* \texttt{string} \sigma^* \rangle \\ + \langle \sigma^* , \sigma^* \texttt{s} , \texttt{tring} \sigma^* \rangle \\ + \langle \sigma^* , \sigma^* \texttt{st} , \texttt{ring} \sigma^* \rangle \\ \vdots \\ + \langle \sigma^* , \sigma^* \texttt{string} \sigma^* , \sigma^* \rangle \\ \vdots \\ + \langle \sigma^* \texttt{string} \sigma^* , \sigma^* , \sigma^* \rangle \end{cases}$$

Possibilities number

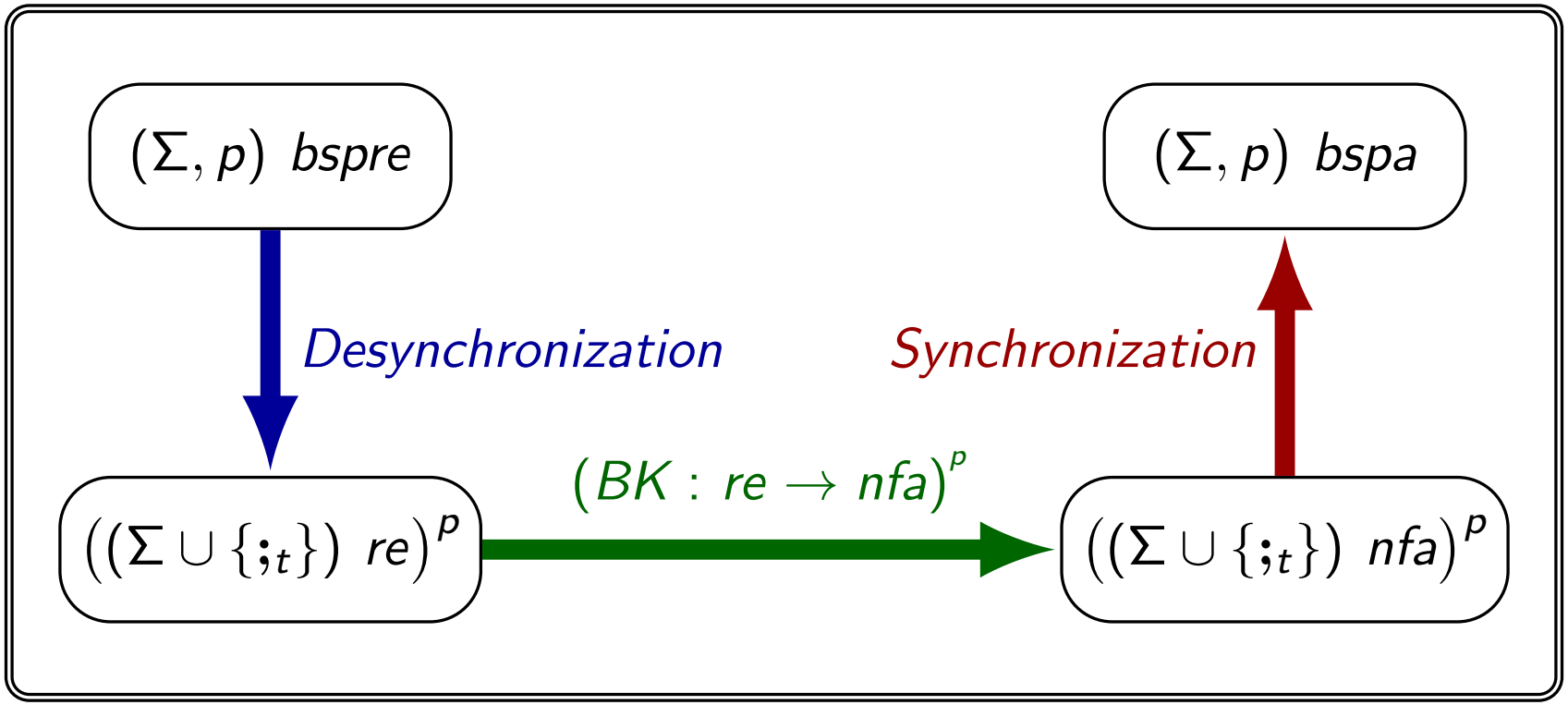
$$(p-1) \times m + 1$$

Parallel recognition of RE



From BSPRE to BSPA

Algorithm global scheme



Pre-processing

- Vectors annotated with their position
- *Depth First Search* of syntactic tree
- Vector number : S

Pre-processing

- Vectors annotated with their position
- *Depth First Search* of syntactic tree
- Vector number : S

Annotation example

$$\left(\left\langle \begin{smallmatrix} a, \\ a+b \end{smallmatrix} \right\rangle + \left\langle \begin{smallmatrix} bb, \\ a^* \end{smallmatrix} \right\rangle\right) \left\langle \begin{smallmatrix} (a+b)^*, \\ aa \end{smallmatrix} \right\rangle$$

Pre-processing

- Vectors annotated with their position
- *Depth First Search* of syntactic tree
- Vector number : S

Annotation example

$$\left(\left\langle \begin{smallmatrix} a, \\ a+b \end{smallmatrix} \right\rangle + \left\langle \begin{smallmatrix} bb, \\ a^* \end{smallmatrix} \right\rangle\right) \left\langle \begin{smallmatrix} (a+b)^*, \\ aa \end{smallmatrix} \right\rangle$$

$$\Longrightarrow \left(\left\langle \begin{smallmatrix} a, \\ a+b \end{smallmatrix} \right\rangle_0 + \left\langle \begin{smallmatrix} bb, \\ a^* \end{smallmatrix} \right\rangle_1\right) \left\langle \begin{smallmatrix} (a+b)^*, \\ aa \end{smallmatrix} \right\rangle_2$$

Desynchronization $(\Sigma, p)\ bspre \rightarrow \left((\Sigma \cup \{;_t\})\ re\right)^p$

Auxiliary recursive function

$$
\begin{aligned}
dsn^i(F \cdot_{BSP} G) &= dsn^i(F) \cdot dsn^i(G) \\
dsn^i(F +_{BSP} G) &= dsn^i(F) + dsn^i(G) \\
dsn^i(F^{*_{BSP}}) &= dsn^i(F)^* \\
dsn^i(\langle r_0, \ldots, r_{p-1} \rangle_{\boldsymbol{t}}) &= r_i \cdot ;_{\boldsymbol{t}} \\
dsn^i(\epsilon_{BSP}) &= \epsilon \\
dsn^i(\emptyset_{BSP}) &= \emptyset
\end{aligned}
$$

Main function

$$Dsn(R) = \langle dsn^0(R), \ldots, dsn^{p-1}(R) \rangle$$

Desynchronization $(\Sigma, p)\ bspre \rightarrow \left((\Sigma \cup \{;_t\})\ re\right)^p$

Auxiliary recursive function

$$
\begin{aligned}
dsn^i(F \cdot_{BSP} G) &= dsn^i(F) \cdot dsn^i(G) \\
dsn^i(F +_{BSP} G) &= dsn^i(F) + dsn^i(G) \\
dsn^i(F^{*_{BSP}}) &= dsn^i(F)^* \\
dsn^i(\langle r_0, \ldots, r_{p-1} \rangle_{\boldsymbol{t}}) &= r_i \cdot ;_{\boldsymbol{t}} \\
dsn^i(\epsilon_{BSP}) &= \epsilon \\
dsn^i(\emptyset_{BSP}) &= \emptyset
\end{aligned}
$$

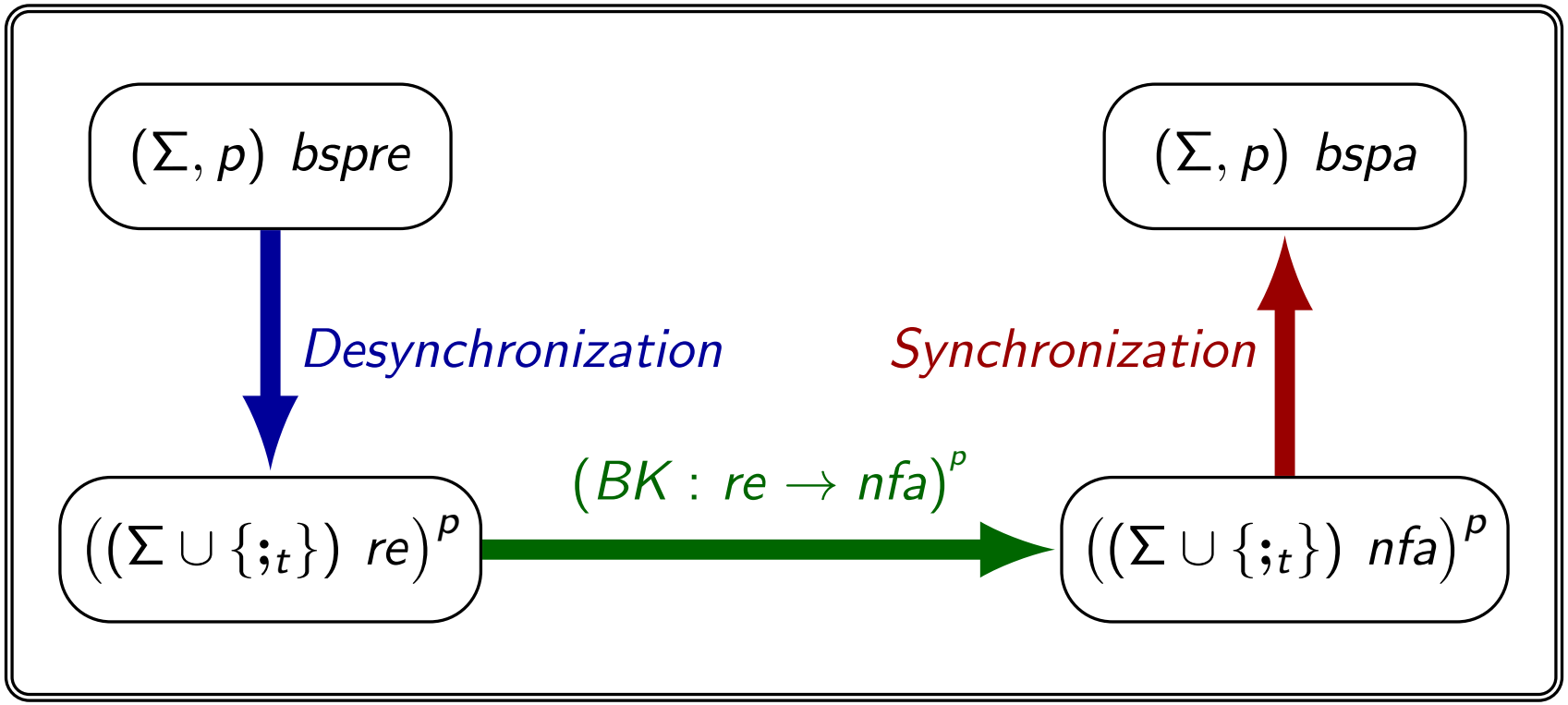
Main function

$$Dsn(R) = \langle dsn^0(R), \ldots, dsn^{p-1}(R) \rangle$$

$$\boxed{\Sigma_{dsync} = \Sigma \cup \left(\bigcup_{t \in [S]} ;_t \right)}$$

From BSPRE to BSPA

Algorithm global scheme



Brüggemann-Klein $((\Sigma \cup \{;_t\})\ re) \rightarrow ((\Sigma \cup \{;_t\})\ nfa)$

Principles

- Improvement of Glushkov $O(n^3)$
- Pre-processing : *Star Normal From*

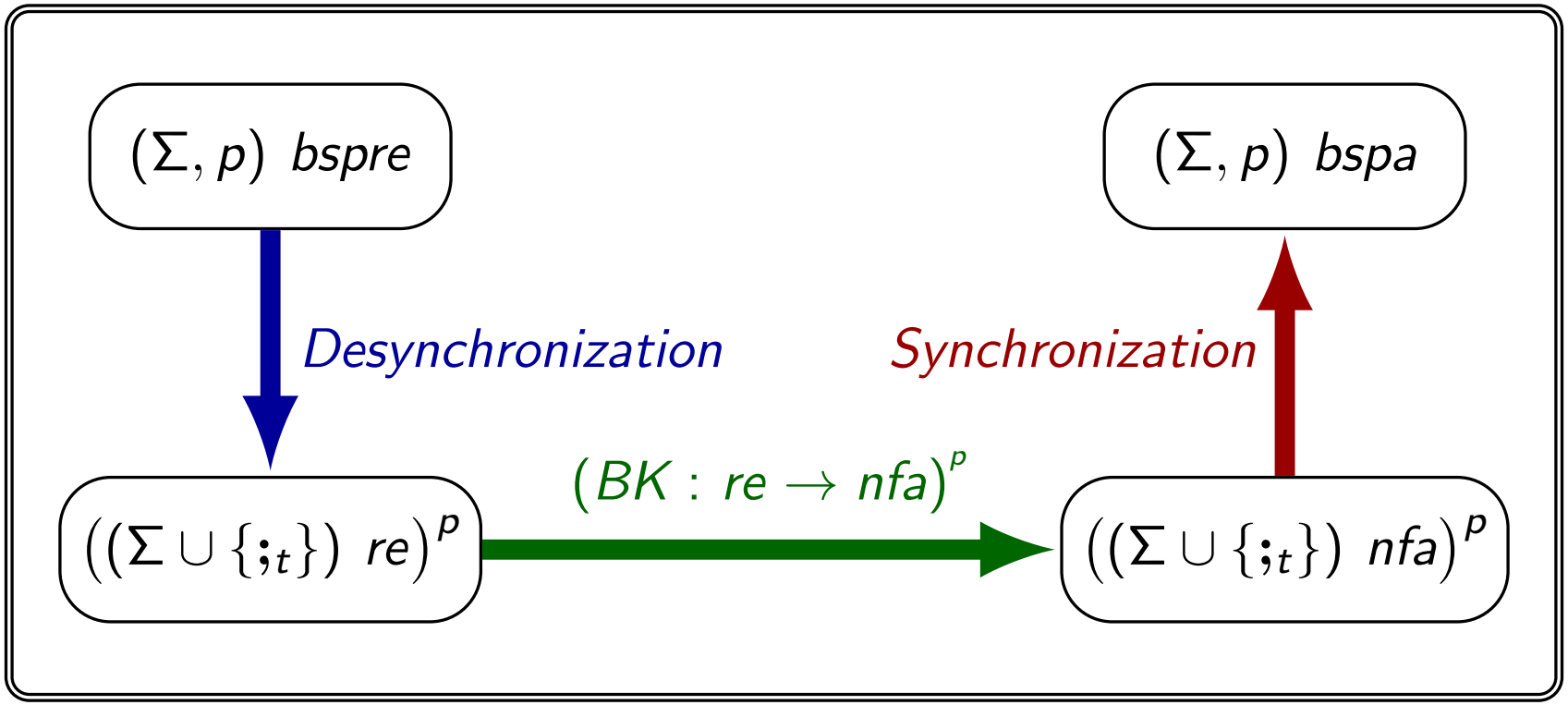
Features

- Time complexity : $O(n^2)$
- Space complexity : $O(n)$
- No ϵ-transitions.

$$\forall i \in [p],\ A^i = (Q^i, \Sigma_{dsync}, \delta^i_{dsync}, I^i, F^i)$$

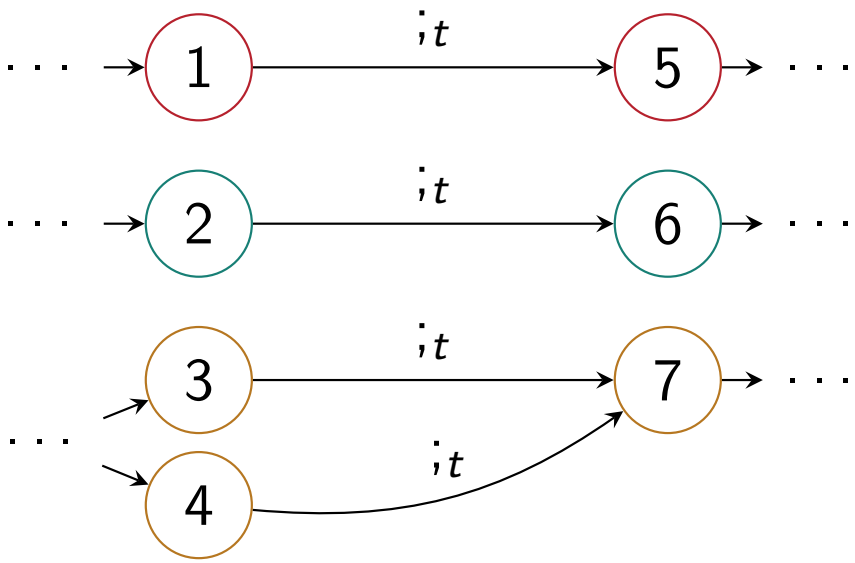
From BSPRE to BSPA

Algorithm global scheme



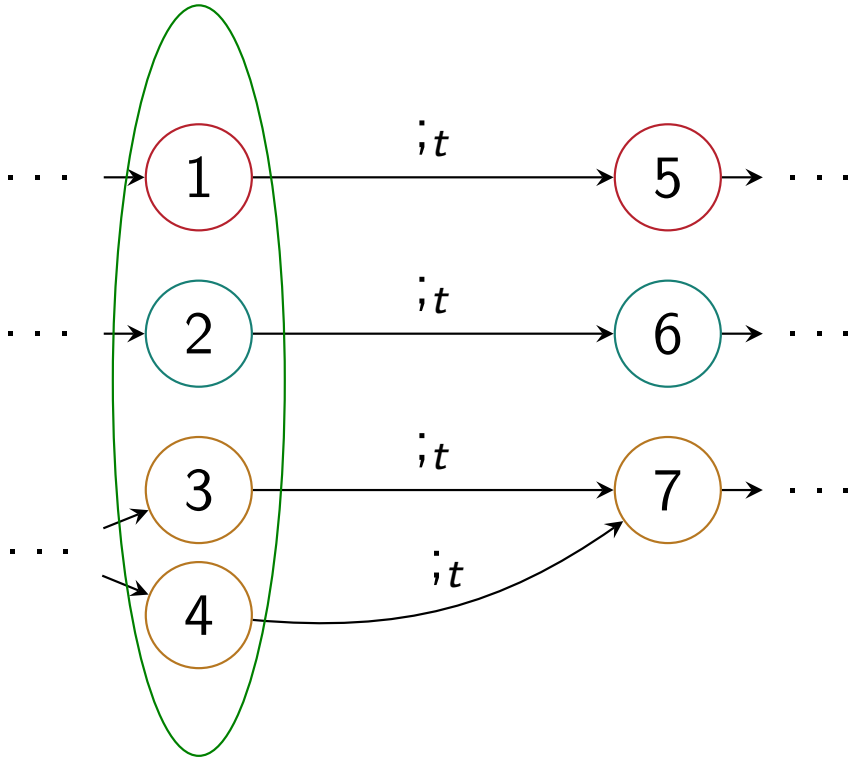
Synchronization : Auxiliary functions

For each $;_t$



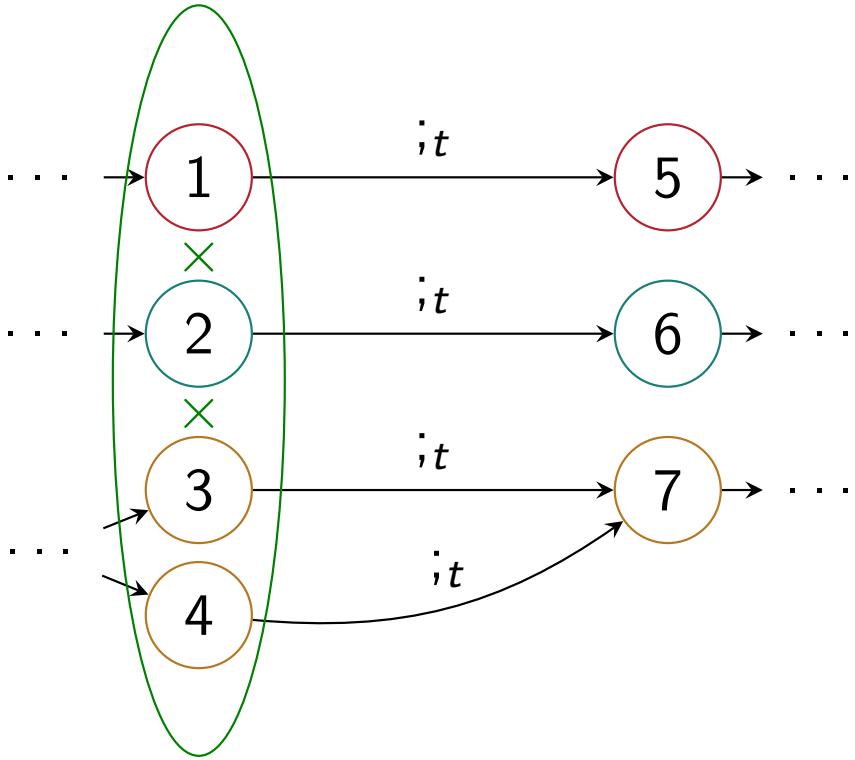
Synchronization : Auxiliary functions

For each $;_t$



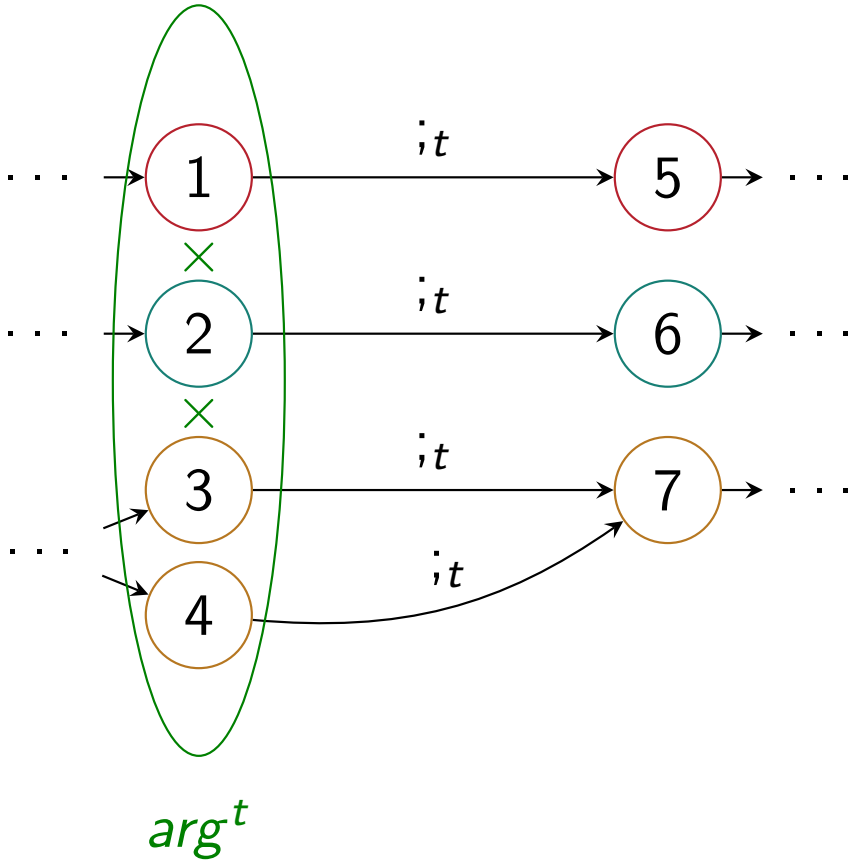
Synchronization : Auxiliary functions

For each $;_t$



Synchronization : Auxiliary functions

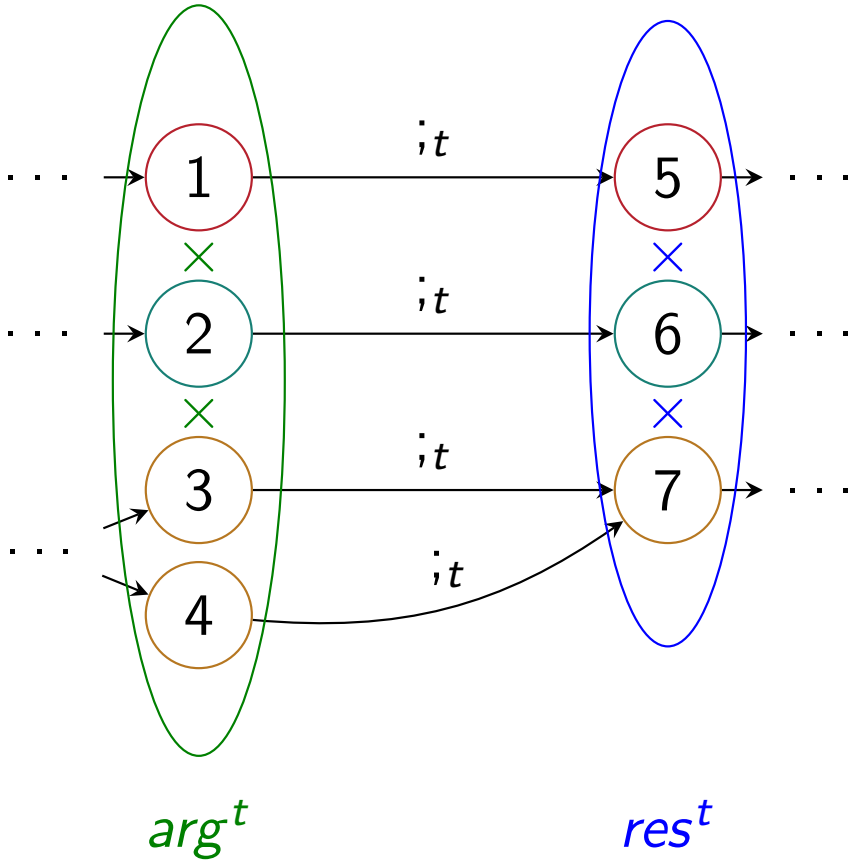
For each $;_t$



arg^t : state vector set

Synchronization : Auxiliary functions

For each $;_t$

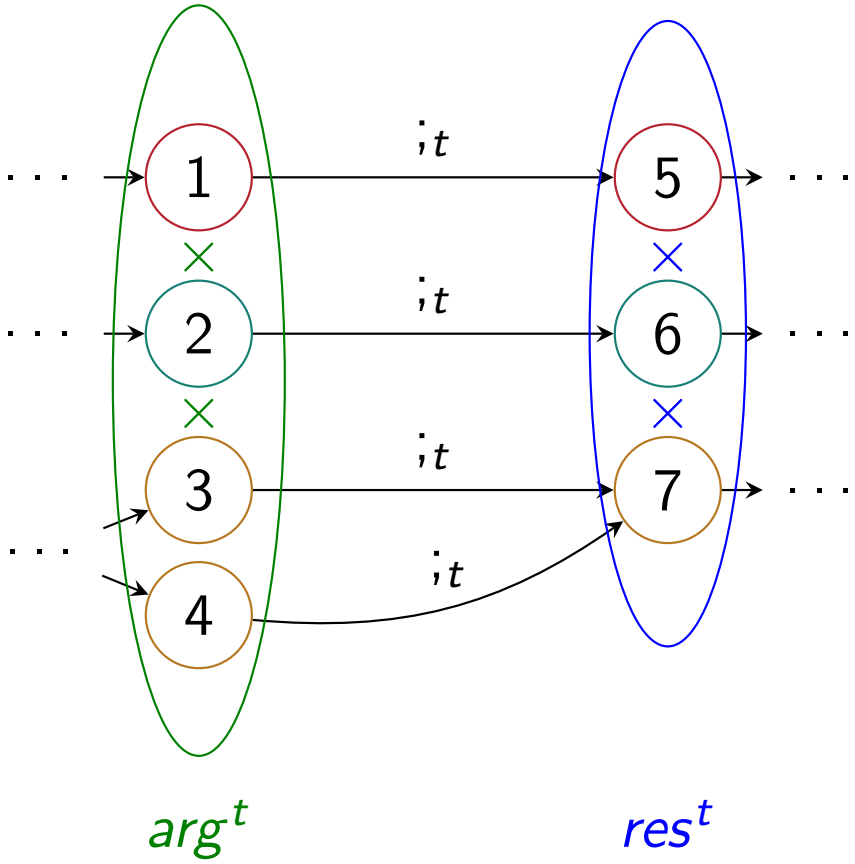


arg^t : state vector set

res^t : state vector set

Synchronization : Auxiliary functions

For each $;_t$



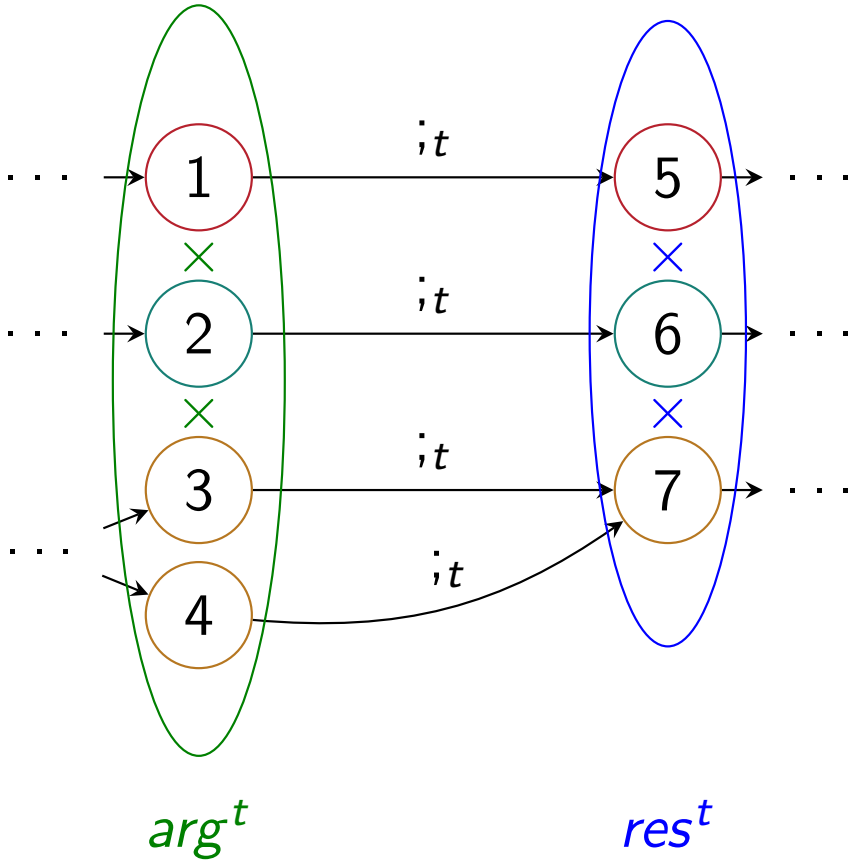
arg^t : state vector set

res^t : state vector set

$$\Delta = \bigcup_{\vec{q} \in arg^t} \bigcup_{\vec{q}' \in res^t} \left(\vec{q} \rightarrow \vec{q}' \right)$$

Synchronization : Auxiliary functions

For each $;_t$



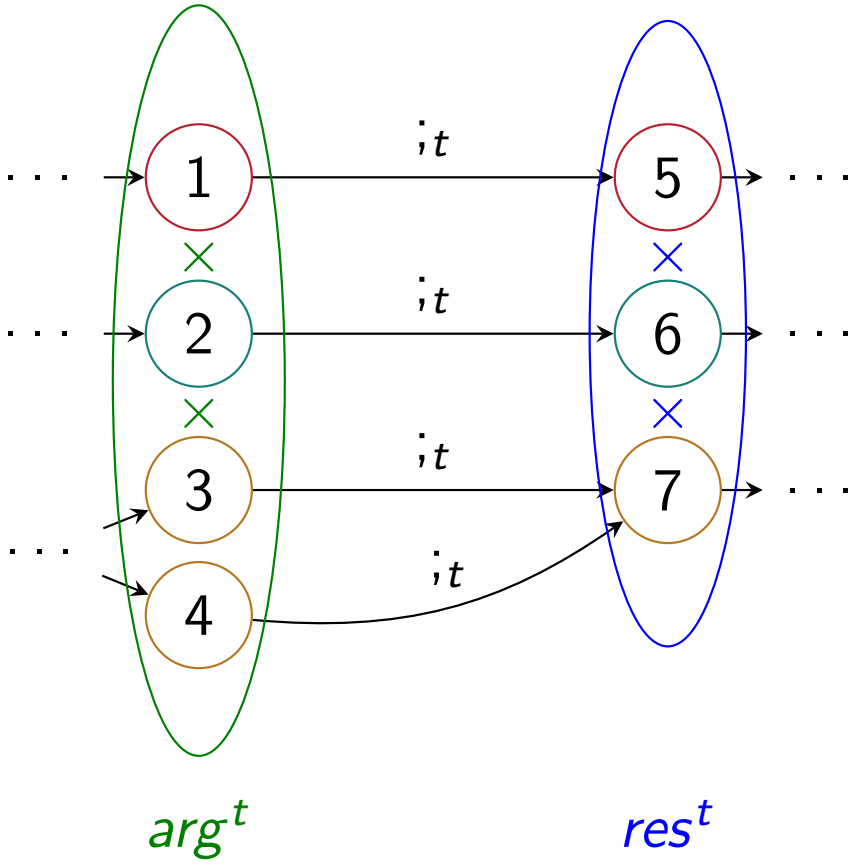
arg^t : state vector set

res^t : state vector set

$$\Delta = \bigcup_{t \in [S]} \left(\bigcup_{\vec{q} \in arg^t} \bigcup_{\vec{q}' \in res^t} \left(\vec{q} \rightarrow \vec{q}' \right) \right)$$

Synchronization : Auxiliary functions

For each $;_t$



arg^t : state vector set

res^t : state vector set

$$\Delta = \bigcup_{t \in [S]} \left(\bigcup_{\vec{q} \in arg^t} \bigcup_{\vec{q}' \in res^t} \left(\vec{q} \rightarrow \vec{q}' \right) \right)$$

$$\Delta = \{ \ \langle 1,2,3 \rangle \rightarrow \langle 5,6,7 \rangle \ , \quad \langle 1,2,4 \rangle \rightarrow \langle 5,6,7 \rangle \ , \quad \ldots \quad \}$$

Synchronization : Auxiliary functions

Input vectors of Δ

$$
\begin{aligned}
src^t(\delta_{dsync}) &= \left\{ q \mid \left((q, ;_{\boldsymbol{t}}) \to q' \right) \in \delta_{dsync} \right\} \\
arg^t(\{\delta^i\}_{i \in [p]}) &= src^t(\delta^0_{dsync}) \times \cdots \times src^t(\delta^{p-1}_{dsync})
\end{aligned}
$$

Output vectors of Δ

$$
\begin{aligned}
dst^t(\delta_{dsync}) &= \left\{ q' \mid \left((q, ;_{\boldsymbol{t}}) \to q' \right) \in \delta_{dsync} \right\} \\
res^t(\{\delta^i\}_{i \in [p]}) &= dst^t(\delta^0_{dsync}) \times \cdots \times dst^t(\delta^{p-1}_{dsync})
\end{aligned}
$$

Synchronization $\quad ((\Sigma \cup \{;_t\})\ nfa)^p \to (\Sigma, p)\ bspa$

$$Syn(\ (\{Q^i\}_{i\in[p]}, \Sigma_{dsync}, \{\delta^i_{dsync}\}_{i\in[p]}, \{I^i\}_{i\in[p]}, \{F^i\}_{i\in[p]}))$$

$$= (\{Q^i\}_{i\in[p]}, \Sigma, \{\delta^i\}_{i\in[p]}, \{I^i\}_{i\in[p]}, \{F^i\}_{i\in[p]}, \Delta)$$

where

$$\Delta = \bigcup_{t\in[S]} \left(\bigcup_{\vec{q}\in arg^t} \bigcup_{\vec{q}'\in res^t} \left(\vec{q} \to \vec{q}'\right)\right)$$

$$\delta^i = \delta^i_{dsync} \setminus \bigcup_{t\in[S]} \left((q, ;_{\boldsymbol{t}}\ ) \to q'\right) \in \delta^i_{dsync}$$

$$\Sigma = \Sigma_{dsync} \setminus \bigcup_{t\in[S]} ;_{\boldsymbol{t}}$$

Example

BSPRE

$$\left\langle {(a+b)a, \atop ab} \right\rangle_0 \left(\left\langle {(\epsilon+b)b, \atop ba} \right\rangle_1 + \left\langle {ab, \atop a} \right\rangle_2 \right)$$

Example

BSPRE

$$\left\langle \begin{matrix} (a+b)a, \\ ab \end{matrix} \right\rangle_0 \left(\left\langle \begin{matrix} (\epsilon+b)b, \\ ba \end{matrix} \right\rangle_1 + \left\langle \begin{matrix} ab, \\ a \end{matrix} \right\rangle_2 \right)$$

Desynchronization

$$\left\langle \begin{matrix} (a+b)\ a\ ;_0 \Big((\epsilon+b)\ b\ ;_1) + (a\ b\ ;_2) \Big), \\ a\ b\ ;_0 \Big((b\ a\ ;_1) + (a\ ;_2) \Big) \end{matrix} \right\rangle$$

Example

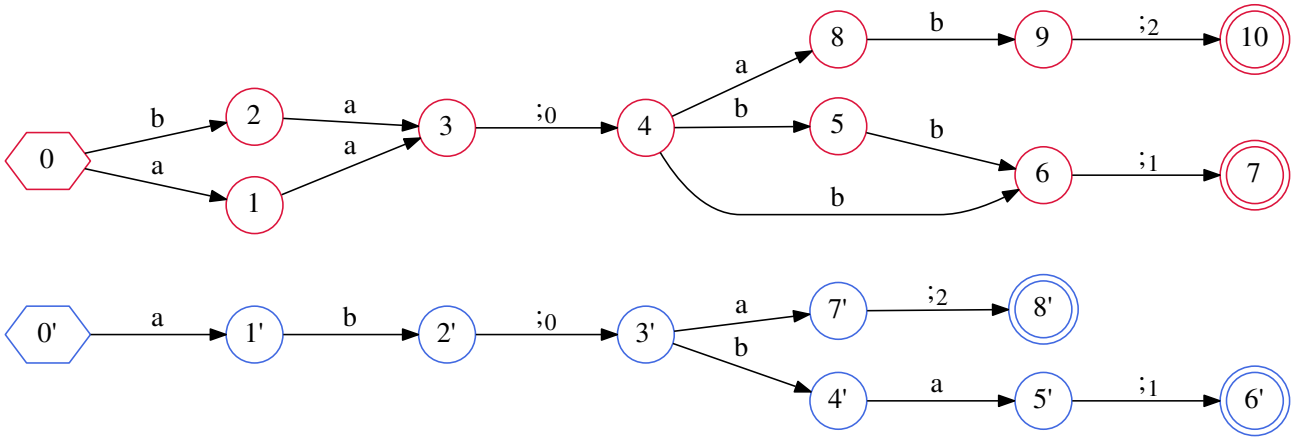
BSPRE

$$\left\langle \begin{matrix} (a+b)a, \\ ab \end{matrix} \right\rangle_0 \left(\left\langle \begin{matrix} (\epsilon+b)b, \\ ba \end{matrix} \right\rangle_1 + \left\langle \begin{matrix} ab, \\ a \end{matrix} \right\rangle_2 \right)$$

Desynchronization

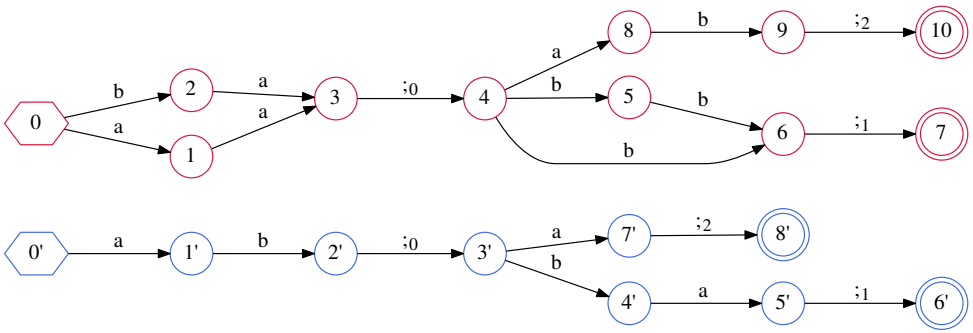
$$\left\langle \begin{matrix} (a+b)\ a\ ;_0 \Big((\epsilon+b)\ b\ ;_1) + (a\ b\ ;_2) \Big), \\ a\ b\ ;_0 \Big((b\ a\ ;_1) + (a\ ;_2) \Big) \end{matrix} \right\rangle$$

BK

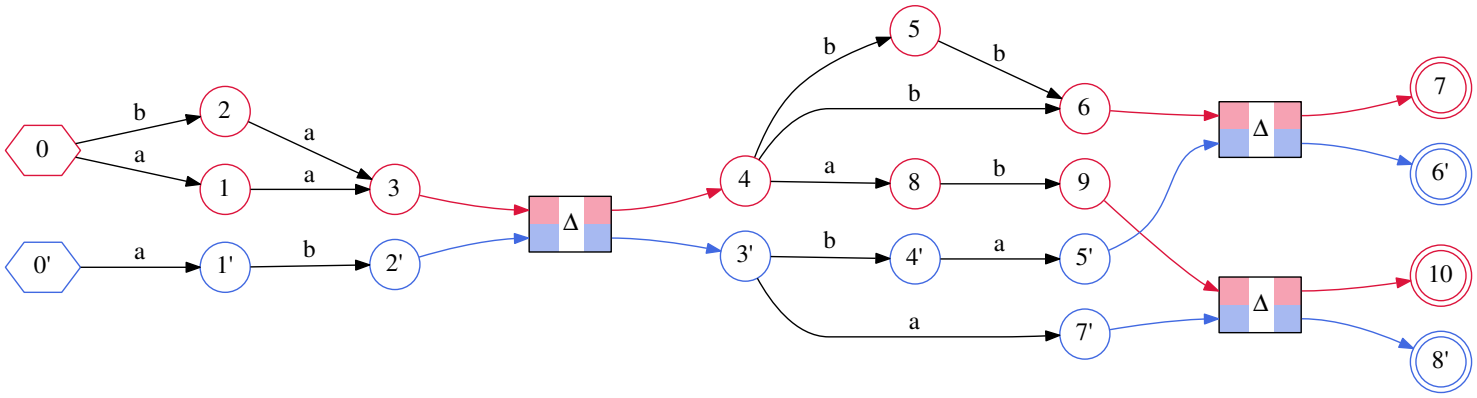


Example

Desynchronized automata



Synchronization



Conclusion

What is done

- Software for RE and Automata transformation
- BSPRE $\rightarrow$ BSPA

What is being done

- Determinize and minimize BSPA
- Automatize RE $\rightarrow$ BSPRE
- Implement BSPA recognition