From 2way transducers to regular function expressions

N. Baudru and P.A. Reynier

Aix-Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France

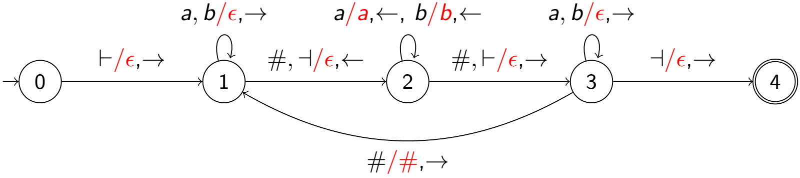
ANR DeLTA, Bordeaux, Dec. 2018

Theorem

Unambiguous 2NFTs and Reg-Expressions are equivalent.

$$\text{iteratedMirror}: \quad \textit{reverses}\#\textit{the}\#\textit{words} \quad \mapsto \quad \textit{sesrever}\#\textit{eht}\#\textit{sdrow}$$

A 2way transducer (2NFT) for iteratedMirror



A regular function expression (Reg-expr) for iteratedMirror

$$\text{iteratedMirror} = \left(\left(\text{id}_{\{a,b\}}\right)^{\overleftarrow{*}} \bullet \{\#\}/\# \right)^{*} \bullet \left(\text{id}_{\{a,b\}}\right)^{\overleftarrow{*}}$$

Our contribution

$$\begin{tikzcd}
\text{unambiguous 2NFT} \arrow[d, leftrightarrow, "\text{[EH, 2001]}"'] & & \text{Reg-Expression} \arrow[d, leftrightarrow, "\text{[AFR, LICS'14]}"] \\
\text{MSOT} \arrow[rr, leftrightarrow, "\text{[AC, FSTTCS'10]}"'] & & \text{copyless SST}
\end{tikzcd}$$

Our contribution

$$\begin{tikzcd}
\text{unambiguous 2NFT} \arrow[rr, "\text{Our construction}"] \arrow[d, leftrightarrow, "{[\text{EH, 2001}]}"'] & & \text{Reg-Expression} \arrow[d, leftrightarrow, "{[\text{AFR, LICS'14}]}"] \\
\text{MSOT} \arrow[rr, leftrightarrow, "{[\text{AC, FSTTCS'10}]}"'] & & \text{copyless SST}
\end{tikzcd}$$

An extension of the state eliminiation algorithm (Brzozowski & McCluskey) to two way transducers.

Remark: a recent work [DGK, LICS'18] also provides a transformation from det 2NFT to Reg-expr, but based on Simon factorization forest theorem.

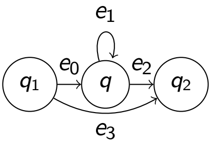
Brzozowski and McCluskey algorithm (BMC) for languages

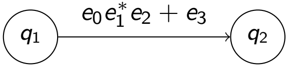
Input : a FA viewed as a generalized automaton $\mathcal{A}$

Output : a generalized automaton with a unique transition:
the RE labelling the transition is equivalent to $\mathcal{A}$.

How : For all states $q \neq \{\text{init}, \text{final}\}$, do

1. for all $q_1, q_2 \neq q$:

   replace 

   with 

2. remove q

Outline

Outline

Regular function expressions (Reg-expressions) [AFR, LICS'14]

Specify word-to-word partial functions

$$Reg \ni f, g ::= R/v \mid f \oplus g \mid f \otimes g \mid f \bullet g \mid f^* \mid f \overleftarrow{\bullet} g \mid \langle f, R\rangle^{\circledast} \mid \overleftarrow{\langle f, R\rangle^{\circledast}}$$

Sum: If $\mathrm{dom}(f)$ and $\mathrm{dom}(g)$ are disjoint, $f \oplus g(u) = \begin{cases} f(u) \text{ if } u \in \mathrm{dom}(f) \\ g(u) \text{ if } u \in \mathrm{dom}(g) \end{cases}$

Hadamard Product : If $u \in \mathrm{dom}(f) \cap \mathrm{dom}(g)$, then $f \otimes g(u) = f(u)g(u)$

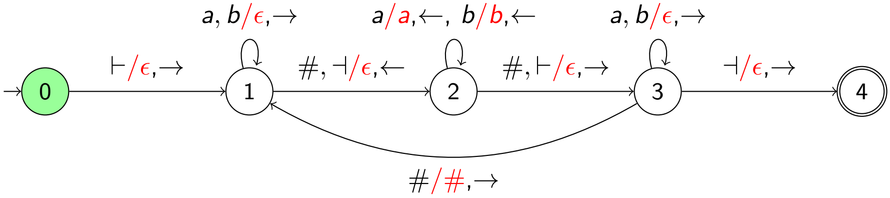
Cauchy Product: If u splits into $u_1 \in \mathrm{dom}(f)$ and $u_2 \in \mathrm{dom}(g)$,

$$f \bullet g(u) = f(u_1)g(u_2)$$

Chained star: If $R^2 \subseteq \mathrm{dom}(f)$ and u splits into $u_1 u_2 \dots u_n$ with $u_i \in R$,

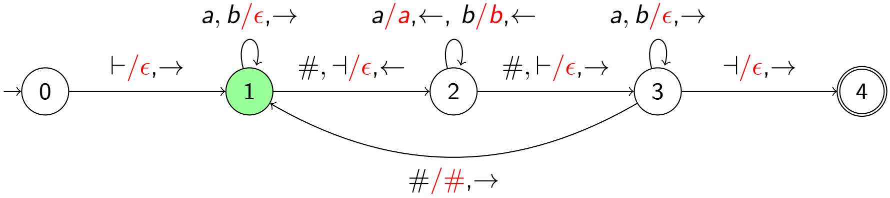
$$\langle f, R\rangle^{\circledast}(u) = f(u_1 u_2) f(u_2 u_3) \dots f(u_{n-1} u_n)$$

2-way Finite-state Transducer (2NFT)



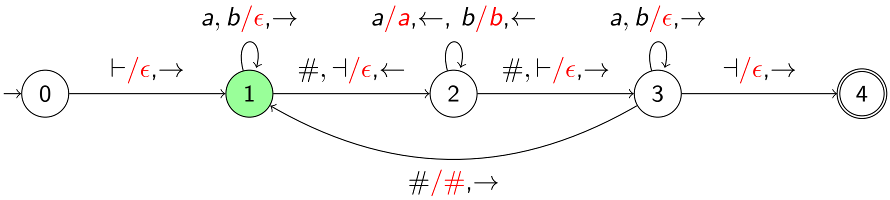
An accepting run on the input word $\vdash ab\#bba\dashv$ **with output** $ba\#abb$**:**

2-way Finite-state Transducer (2NFT)



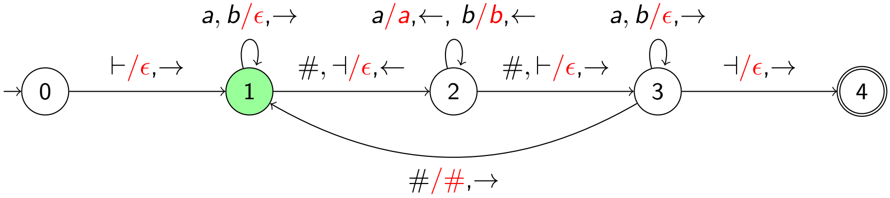
An accepting run on the input word $\vdash ab\#bba\dashv$ with output $ba\#abb$:

2-way Finite-state Transducer (2NFT)



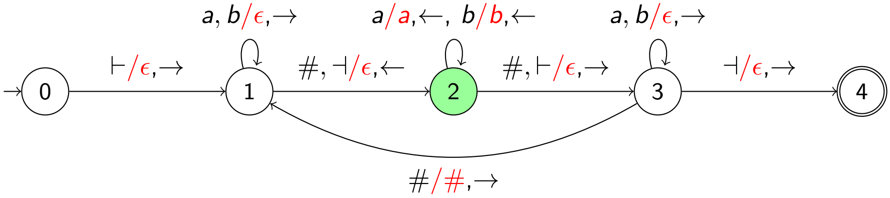
An accepting run on the input word $\vdash ab\#bba\dashv$ **with output** $ba\#abb$**:**

2-way Finite-state Transducer (2NFT)

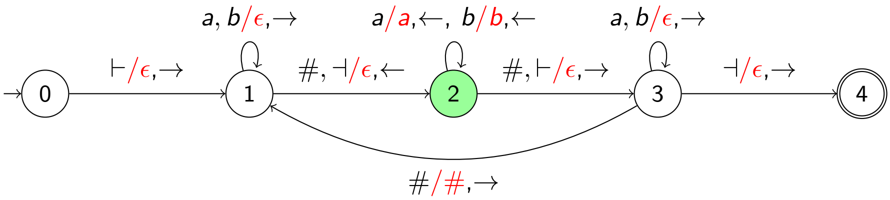


An accepting run on the input word $\vdash ab\#bba\dashv$ **with output** $ba\#abb$**:**

2-way Finite-state Transducer (2NFT)

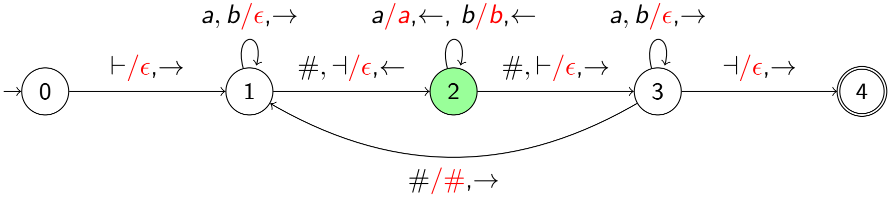


An accepting run on the input word $\vdash ab\#bba\dashv$ with output $ba\#abb$:



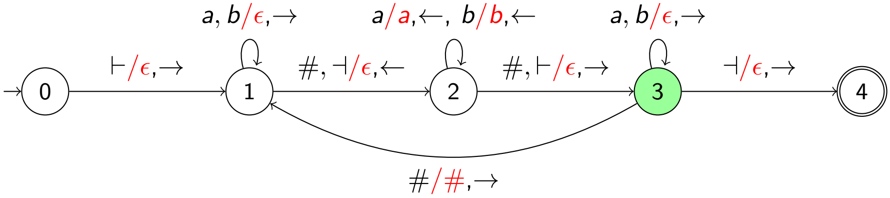
An accepting run on the input word $\vdash ab\#bba\dashv$ **with output** $ba\#abb$**:**

2-way Finite-state Transducer (2NFT)

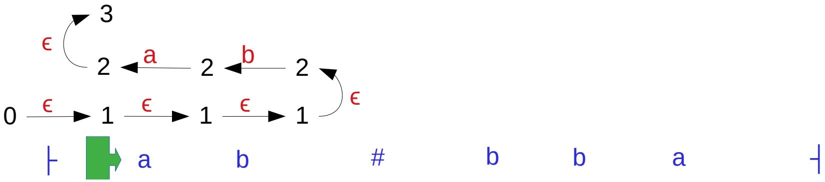


An accepting run on the input word $\vdash ab\#bba\dashv$ with output $ba\#abb$:

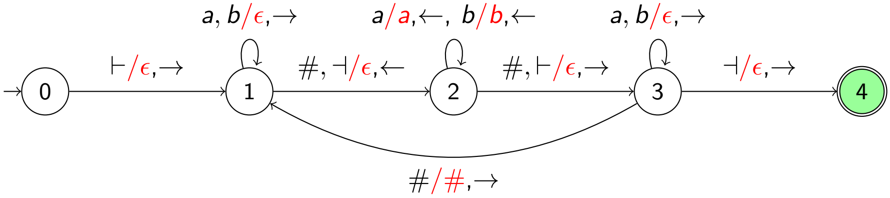
2-way Finite-state Transducer (2NFT)



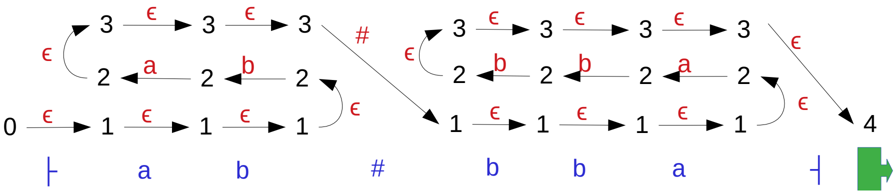
An accepting run on the input word $\vdash ab\#bba\dashv$ **with output** $ba\#abb$**:**



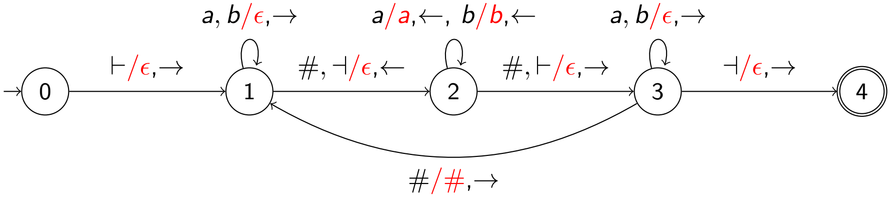
2-way Finite-state Transducer (2NFT)



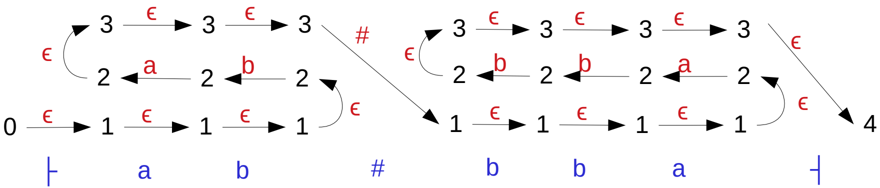
An accepting run on the input word $\vdash ab\#bba\dashv$ **with output** $ba\#abb$**:**



2-way Finite-state Transducer (2NFT)

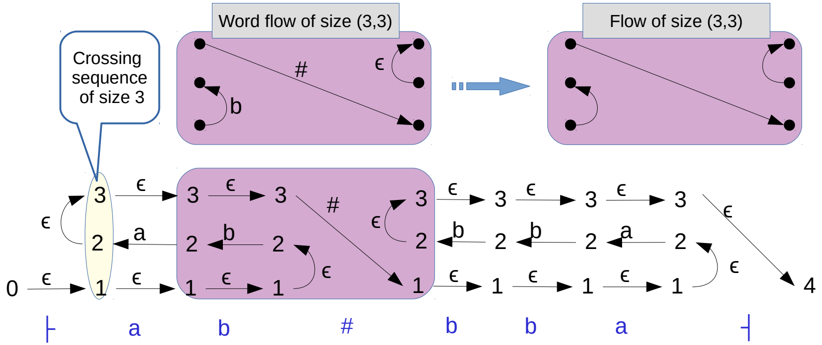


An accepting run on the input word $\vdash ab\#bba\dashv$ **with output** $ba\#abb$**:**

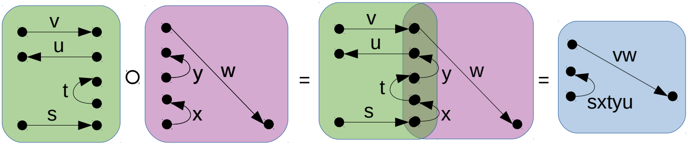


Unambiguous 2NFTs define word-to-word partial functions.

Monoid of (word) flows [Shepherdson'59]



Concatenation of size compatible flows:

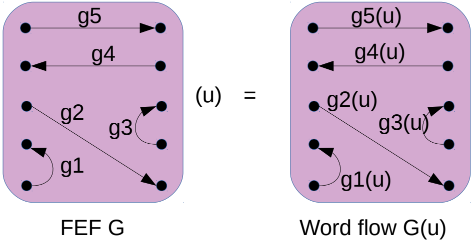


Outline

Function expression flows (FEF) and labels

FEF on domain D : flow labelled by Reg-expressions on domain D.

An FEF G defines a function from the words u of D to word flows.

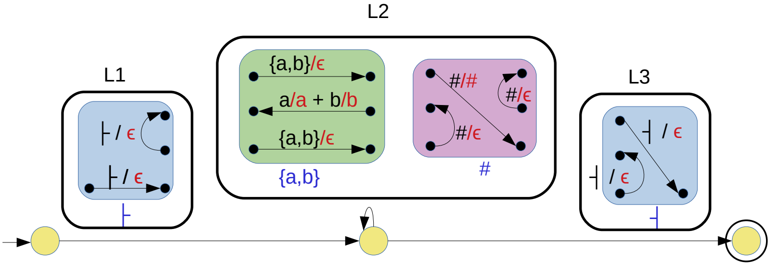


Label: finite set of FEFs of same size with pairwise disjoint domains

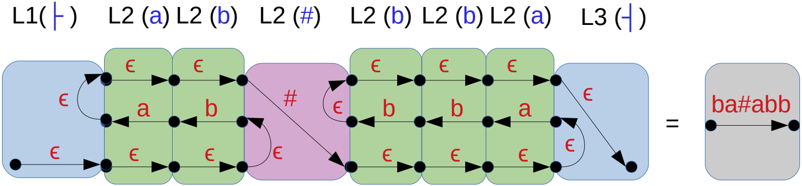
A label L defines a function from $\biguplus_{G \in L} \mathrm{dom}(G)$ to word flows:

$$L(u) = G(u) \text{ for the unique } G \in L \text{ s.t. } u \in \mathrm{dom}(G)$$

FA $\mathcal{A}$ over labels with a structural property

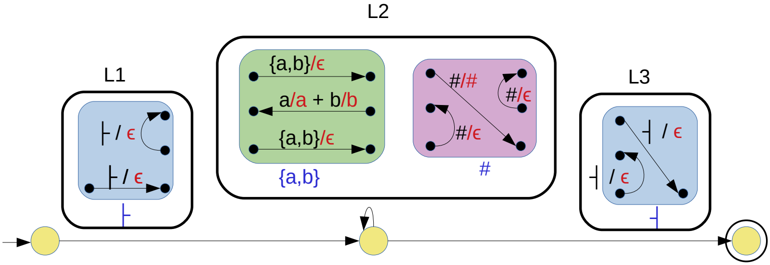


Example: $\mathcal{A}(\vdash ab\#bba \dashv) =$



Function expression flow automata (FFA)

FA $\mathcal{A}$ over labels with a structural property



Using the crossing sequences construction [Sheph. 1959], we have:

Proposition

For all unambiguous 2NFT, we can build an equivalent unambiguous FFA.

From now on, we consider unambiguous FFA.

Tailoring the BMC algorithm to FFA

Input: An unambiguous FFA $\mathcal{A}$

Output: An FFA with a unique transition:

- The label L of the transition is equivalent to $\mathcal{A}$.

Tailoring the BMC algorithm to FFA

Input: An unambiguous FFA $\mathcal{A} \leftrightsquigarrow$ 2NFT $\mathcal{B}$ (Sheph.'s construction)

Output: An FFA with a unique transition:

- The label L of the transition is equivalent to $\mathcal{A}$.
- $L = \{F\}$ of size $(1, 1)$. The unique reg-expression is equivalent to $\mathcal{B}$.

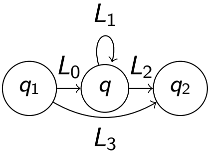
Tailoring the BMC algorithm to FFA

Input: An unambiguous FFA $\mathcal{A} \leftsquigarrow$ 2NFT $\mathcal{B}$ (Sheph.'s construction)

Output: An FFA with a unique transition:

- The label L of the transition is equivalent to $\mathcal{A}$.
- $L = \{F\}$ of size $(1, 1)$. The unique reg-expression is equivalent to $\mathcal{B}$.

Algo:

replace


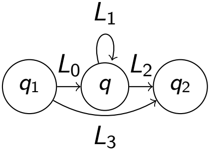
with


Tailoring the BMC algorithm to FFA

Input: An unambiguous FFA $\mathcal{A} \leftrightsquigarrow$ 2NFT $\mathcal{B}$ (Sheph.'s construction)

Output: An FFA with a unique transition:

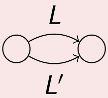
- The label L of the transition is equivalent to $\mathcal{A}$.
- $L = \{F\}$ of size $(1,1)$. The unique reg-expression is equivalent to $\mathcal{B}$.

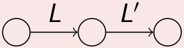
Algo: replace 

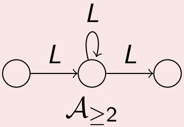
with 

Proposition

We can effectively build labels $L \oplus L'$, $L \bullet L'$ and $L^{\geq 2}$ such that

$L \oplus L' \equiv$ 

$L \bullet L' \equiv$ 

$L^{\geq 2} \equiv$ 

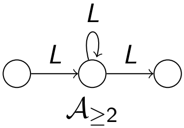
Construction of $L^{\geq 2}$ - overview

If L satisfies the left-absorbing property (key case) :

$$\text{for all } F, F' \in L, \ \mathsf{flow}(F) \circ \mathsf{flow}(F') = \mathsf{flow}(F).$$

The construction $L^{\geq 2}$ follows from

- an analysis of the sequences of flows induced by $\mathcal{A}_{\geq 2}$:
- the use of the chained star combinator.



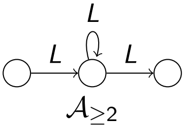
Construction of $L^{\geq 2}$ - overview

If L satisfies the left-absorbing property (key case) :

$$\text{for all } F, F' \in L, \ \text{flow}(F) \circ \text{flow}(F') = \text{flow}(F).$$

The construction $L^{\geq 2}$ follows from

- an analysis of the sequences of flows induced by $\mathcal{A}_{\geq 2}$:
- the use of the chained star combinator.

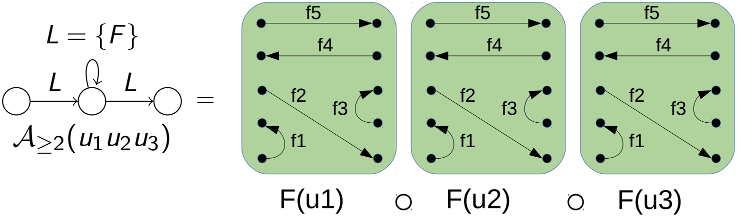


Otherwise:

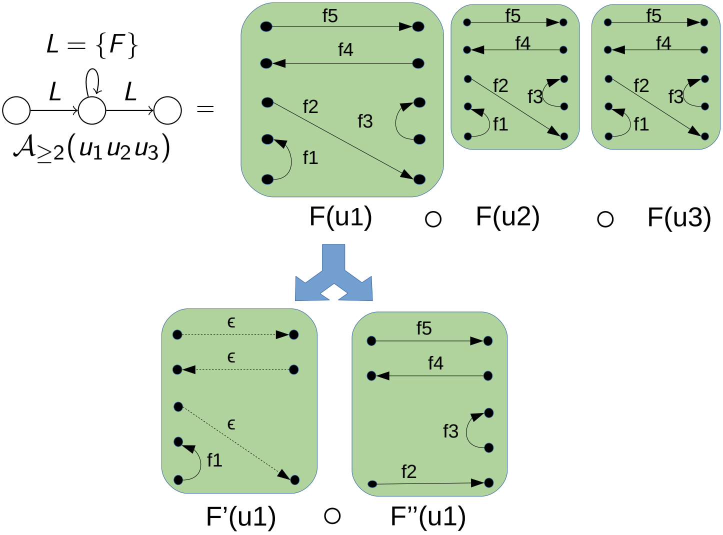
1. use a finite unfolding of $\mathcal{A}_{\geq 2}$
2. apply again our BMC algorithm with a state elimination strategy

$\Longrightarrow$ reduction to the previous case

Construction of $L^{\geq 2}$ when L has the left-absorbing property: main ideas...

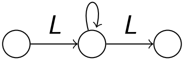

$L = \{F\}$
L
L
$\mathcal{A}_{\geq 2}(u_1 u_2 u_3)$
=
f5
f4
f2
f3
f1
f5
f4
f2
f3
f1
f5
f4
f2
f3
f1
F(u1)
∘
F(u2)
∘
F(u3)

$L = \{F\}$

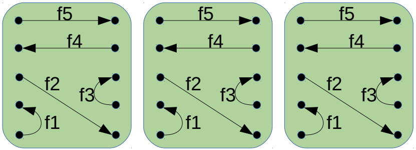


Decompose F**:** For all $u \in \mathrm{dom}(F)$, $F(u) = F'(u) \circ F''(u)$.

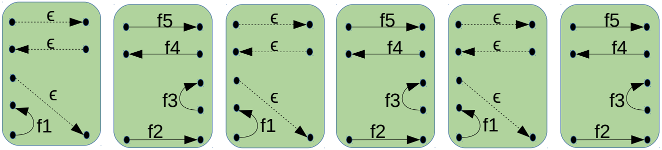
$L = \{F\}$



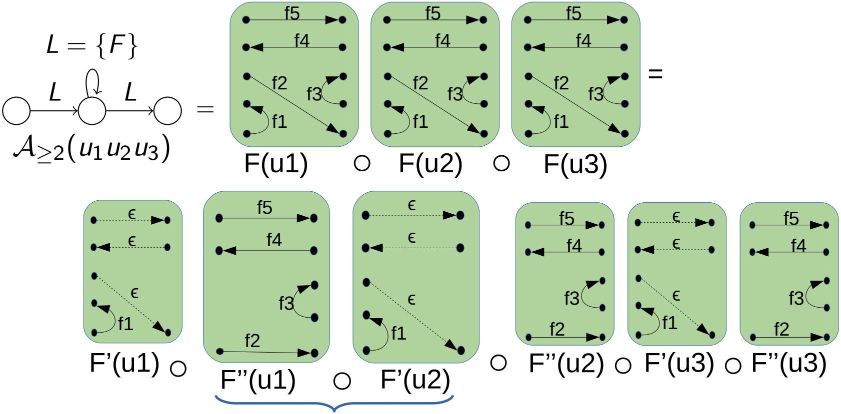
$\mathcal{A}_{\geq 2}(u_1 u_2 u_3) =$

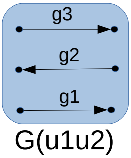


F(u1) ∘ F(u2) ∘ F(u3) =



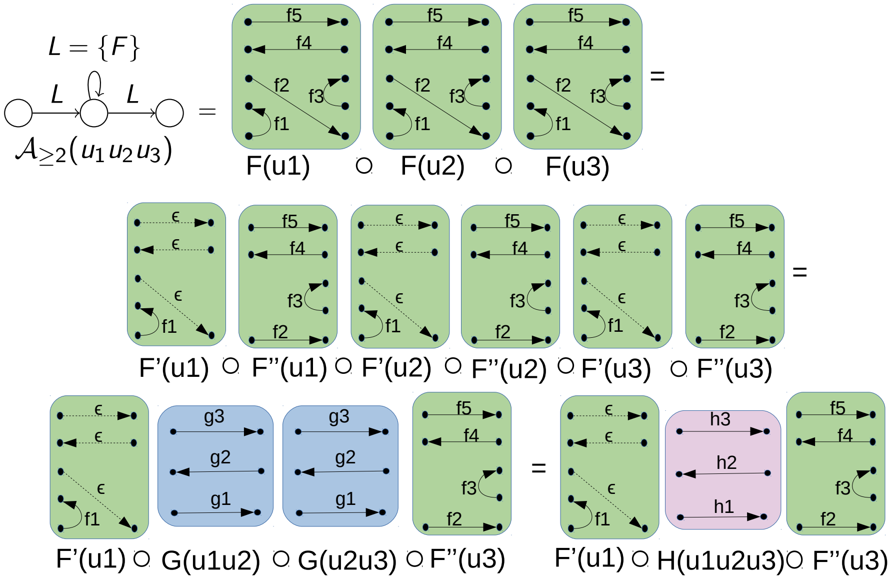
F'(u1) ∘ F''(u1) ∘ F'(u2) ∘ F''(u2) ∘ F'(u3) ∘ F''(u3) =



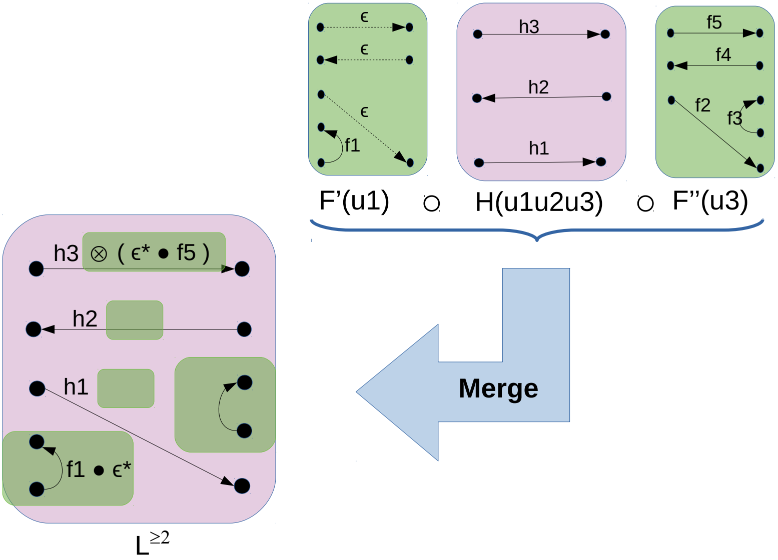


Thanks to left-absorbing property :
G has crossing edges only !

Combine F'' **and** F'**:** For all $u, v \in \mathrm{dom}(F)$, $G(uv) = F''(u) \circ F'(v)$.



Chaine the G's: $h_1(u_1 u_2 u_3) = \langle g_1, \mathrm{dom}(L)\rangle^{\circledast}(u_1 u_2 u_3) = g_1(u_1 u_2) g_1(u_2 u_3)$



ϵ
ϵ
ϵ
f1
h3
h2
h1
f5
f4
f2
f3
F'(u1) ∘ H(u1u2u3) ∘ F''(u3)
Merge
h3 ⊗ (ϵ* • f5)
h2
h1
f1 • ϵ*
L≥2

Conclusion and future works

In this talk: we extended BMC algorithm to unambiguous 2NFT.

Not presented: sweeping transducer.

Future works:

- Complexity analysis
- Simon's Theorem vs our construction
- Expressiveness of non functional FFA
- Infinite words

Thank you for your attention