Jumping Finite Automata: New Results

Part One: Solved Questions

Radim Kocman and Zbyněk Křivka

Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno Czech Republic

{ikocman,krivka}@fit.vutbr.cz

LTA 2016 (December 5, 2016)
Table of Contents

- Introduction

- Definitions and Examples

- Results
  Power of JFAs and GJFAs
  Closure Properties
  Decidability and Complexity

- Concluding Remarks

Vojtěch Vorel: Two Results on Discontinuous Input Processing. DCFS 2016: 205-216


Classical Finite Automata

Accepted language: $\{a\}^* \{c\} \{b\}^*$
Classical Finite Automata

Accepted language: \( \{a\}^*\{c\}\{b\}^* \)
Classical Finite Automata

Accepted language: \{a\} ∗ \{c\} ∗ \{b\} ∗ 

Jumping Finite Automata: New Results – Part One: Solved Questions
Classical Finite Automata

Accepted language: \( \{a\} \ast \{c\} \ast \{b\} \ast \)
Classical Finite Automata

Accepted language: \(\{a\}^* \{c\} \{b\}^*\)
Classical Finite Automata

Accepted language: \( \{a\}^* \{c\}^* \{b\}^* \)
Classical Finite Automata

Accepted language: \( \{a\}^* \{c\} \{b\}^* \)
Classical Finite Automata

Accepted language: \{a \}^* \{c \} \{b \}^*
Classical Finite Automata

Accepted language: \{a\}^* \{c\} \{b\}^*
Accepted language: \{a\}^* \{c\} \{b\}^*
Jumping Finite Automata

Accepted language:
\[ \{ w \in \{ a, b, c \}^*: |w|_a = |w|_b = |w|_c \} \]
Jumping Finite Automata: New Results – Part One: Solved Questions

Accepted language: \( \{ w \in \{a, b, c\}^*: |w|_a = |w|_b = |w|_c \} \)
Jumping Finite Automata

Accepted language: $\{w \in \{a, b, c\}^* : |w|_a = |w|_b = |w|_c\}$
Jumping Finite Automata

Accepted language: $\{w \in \{a, b, c\}^* : |w|_a = |w|_b = |w|_c\}$
Accepted language: \( \{ w \in \{ a, b, c \}^* : |w|_a = |w|_b = |w|_c \} \)
Jumping Finite Automata

Accepted language: \( \{ w \in \{a, b, c\}^* : |w|_a = |w|_b = |w|_c \} \)
Accepted language: \( \{ w \in \{ a, b, c \}^*: |w|_a = |w|_b = |w|_c \} \)
Jumping Finite Automata

Accepted language: 
\{w \in \{a, b, c\}^*: |w|_a = |w|_b = |w|_c\}

Jumping Finite Automata: New Results – Part One: Solved Questions
Accepted language: \( \{ w \in \{ a, b, c \}^* : |w|_a = |w|_b = |w|_c \} \)
| Definitions |

**Definition (Meduna, Zemek (2012))**

A general jumping finite automaton (GJFA) is a quintuple

\[ M = (Q, \Sigma, R, s, F) \]

where

- \( Q \) is a finite set of states;
- \( \Sigma \) is the input alphabet;
- \( R \) is a finite set of rules of the form
  \[ py \rightarrow q \quad (p, q \in Q, y \in \Sigma^*) \]
- \( s \) is the start state;
- \( F \) is a set of final states.
**Definitions**

**Definition (Meduna, Zemek (2012))**

A general jumping finite automaton (GJFA) is a quintuple

\[ M = (Q, \Sigma, R, s, F) \]

where

- \( Q \) is a finite set of states;
- \( \Sigma \) is the input alphabet;
- \( R \) is a finite set of rules of the form
  \[ py \rightarrow q \quad (p, q \in Q, y \in \Sigma^*) \]
- \( s \) is the start state;
- \( F \) is a set of final states.

**Definition**

If all rules \( py \rightarrow q \in R \) satisfy \( |y| \leq 1 \), then \( M \) is a jumping finite automaton (JFA).
Definition

If $x, z, x', z', y \in \Sigma^*$ such that $xz = x'z'$ and $py \rightarrow q \in R$, then $M$ makes a jump from $xpyz$ to $x'qz'$, symbolically written as

$$xpyz \mathbf{↷} x'qz'$$
**Definition**

If $x, z, x', z', y \in \Sigma^*$ such that $xz = x'z'$ and $py \rightarrow q \in R$, then $M$ makes a **jump** from $xpyz$ to $x'qz'$, symbolically written as

$$xp\![yz] \leadsto x'qz'$$

\[ \leadsto^* \]

intuitively, a sequence of jumps (possibly empty); mathematically, the reflexive-transitive closure of $\leadsto$. 

---

**Note:** Hereafter, a family of languages defined by model $X$ is denoted by $L(X)$. 

---

Jumping Finite Automata: New Results – Part One: Solved Questions
Definition

If $x, z, x', z', y \in \Sigma^*$ such that $xz = x'z'$ and $py \rightarrow q \in R$, then $M$ makes a jump from $xpYZ$ to $x'qz'$, symbolically written as

$$xpYZ \sim x'qz'$$

$\sim^*$ intuitively, a sequence of jumps (possibly empty); mathematically, the reflexive-transitive closure of $\sim$

Definition

The language accepted by $M$, denoted by $L(M)$, is defined as

$$L(M) = \{uv : u, v \in \Sigma^*, usv \sim^* f, f \in F\}$$

Note: Hereafter, a family of languages defined by model $X$ is denoted by $\mathcal{L}(X)$. 
Example 1

The JFA

$$M = (\{s, r, t\}, \{a, b, c\}, R, s, \{s\})$$

with

$$R = \{sa \rightarrow r, rb \rightarrow t, tc \rightarrow s\}$$

accepts

$$L(M) = \{w \in \{a, b, c\}^* : |w|_a = |w|_b = |w|_c\}$$

For instance:

$$babcscsa \rightarrow bacrcbc \rightarrow bacbtc \rightarrow bsac \rightarrow rbc \rightarrow tc \rightarrow s$$
Example #2

The GJFA

\[ H = (\{s, f\}, \{a, b\}, R, s, \{f\}) , \]

with

\[ R = \{sba \rightarrow f, fa \rightarrow f, fb \rightarrow f\} \]

accepts

\[ L(H) = \{a, b\}^* \{ba\} \{a, b\}^* \]

For instance:

\[
\begin{align*}
  bbsbaa & \rightsquigarrow bbf a & [sba \rightarrow f] \\
  \rightsquigarrow fbb & [fa \rightarrow f] \\
  \rightsquigarrow fb & [fb \rightarrow f] \\
  \rightsquigarrow f & [fb \rightarrow f]
\end{align*}
\]
Definition

The shuffle operation, denoted by $\omega$, is defined by

$$u \omega v = \left\{ x_1 y_1 x_2 y_2 \ldots x_n y_n : \begin{array}{l} u = x_1 x_2 \ldots x_n, \ v = y_1 y_2 \ldots y_n \vspace{1mm} \end{array} \right\},$$

$$x_i, y_i \in \Sigma^*, 1 \leq i \leq n, n \geq 1,$$

$$L_1 \omega L_2 = \bigcup_{u \in L_1, v \in L_2} (u \omega v),$$

for $u, v \in \Sigma^*$ and $L_1, L_2 \subseteq \Sigma^*$. 
**Definition**

The **shuffle operation**, denoted by $\shuffle$, is defined by

\[
 u \shuffle v = \left\{ x_1 y_1 x_2 y_2 \ldots x_n y_n : u = x_1 x_2 \ldots x_n, v = y_1 y_2 \ldots y_n \right\},
\]

where $x_i, y_i \in \Sigma^*$, $1 \leq i \leq n$, $n \geq 1$.

\[
 L_1 \shuffle L_2 = \bigcup_{u \in L_1, v \in L_2} (u \shuffle v),
\]

for $u, v \in \Sigma^*$ and $L_1, L_2 \subseteq \Sigma^*$.

**Example**

\[
 ab \shuffle cd = \{ abcd, acdb, cdab, acbd, cadb, cabd \}
\]
Definition

For $L \subseteq \Sigma^*$, the iterated shuffle of $L$ is

$$L^{\omega,*} = \bigcup_{n=0}^{\infty} L^{\omega,n},$$

where

$$L^{\omega,0} = \{\varepsilon\}$$

and

$$L^{\omega,i} = L^{\omega,i-1} \shuffle L,$$

where $i \geq 1$. 
Definition

All permutations of $w$, denoted by $\text{perm}(w)$, is defined as

\[
\text{perm}(\varepsilon) = \{\varepsilon\}
\]

\[
\text{perm}(au) = \{a\} \cup \text{perm}(u)
\]

where $a \in \Sigma$ and $u \in \Sigma^*$. 

For $L \subseteq \Sigma^*$, $\text{perm}(L) = \bigcup_{w \in L} \text{perm}(w)$. 
## Definition

All permutations of $w$, denoted by $\text{perm}(w)$, is defined as

$$
\text{perm}(\varepsilon) = \{\varepsilon\}
$$

$$
\text{perm}(au) = \{a\} \cup \text{perm}(u)
$$

where $a \in \Sigma$ and $u \in \Sigma^*$.

For $L \subseteq \Sigma^*$, $\text{perm}(L) = \bigcup_{w \in L} \text{perm}(w)$.

## Example

$$
\text{perm}(abc) = \{abc, acb, cba, bac, bca, cab\}
$$
Definition

All permutations of $w$, denoted by $\text{perm}(w)$, is defined as

\[
\text{perm}(\varepsilon) = \{\varepsilon\}
\]

\[
\text{perm}(au) = \{a\} \cup \text{perm}(u)
\]

where $a \in \Sigma$ and $u \in \Sigma^*$.

For $L \subseteq \Sigma^*$, $\text{perm}(L) = \bigcup_{w \in L} \text{perm}(w)$.

Example

\[
\text{perm}(abc) = \{abc, acb, cba, bac, bca, cab\}
\]

Proposition

For $u, v \in \Sigma^*$, $\text{perm}(u) = \text{perm}(v)$ if and only if $\psi_\Sigma(u) = \psi_\Sigma(v)$. 
### Definition (Jantzen (1979))

Let $\Sigma$ be an alphabet. The (atomic) SHUF expressions are:

- $\emptyset$
- $\varepsilon$
- $w \in \Sigma^+$

If $r$, $s$ are SHUF expressions, then

- $(r + s)$
- $(r \sqcup s)$
- $r^*$

are SHUF expressions. They denote the corresponding languages as expected.

### Definition (Fernau et al. (2016))

A SHUF expression is an $\alpha$-SHUF expression, if its atoms are only $\emptyset$, $\varepsilon$, or single symbols $a \in \Sigma$. 

---

Jumping Finite Automata: New Results – Part One: Solved Questions
Example

The language from Example #1 can be denoted by the following $\alpha$-SHUF expression

$$(a \lor b \lor c)^{\lor,\ast}$$
Relations Between Language Families I

\{w : |w|_a = |w|_b = |w|_c\}

\{a^n b^n c^n : n \geq 0\}

\{a^n b^n : n \geq 0\}

\{a, b\}^* \{ba\} \{a, b\}^*

\{a\}^* \{b\}^*

Jumping Finite Automata: New Results – Part One: Solved Questions
**Power of JFAs**

**Theorem (Meduna, Zemek (2012) & Fernau et al. (2016))**

\[ L(JFA) = \text{perm}(\text{REG}) = \text{perm}(\text{CF}) = \text{perm}(\text{PSL}) \]
Theorem (Meduna, Zemek (2012) & Fernau et al. (2016))

\[ \mathcal{L}(\text{JFA}) = \text{perm}(\text{REG}) = \text{perm}(\text{CF}) = \text{perm}(\text{PSL}) \]

Corollary (Fernau et al. (2016))

\[ \mathcal{L}(\text{JFA}) \text{ is closed under intersection and under complementation.} \]
**Theorem (Meduna, Zemek (2012) & Fernau et al. (2016))**

\[ \mathcal{L}(JFA) = \text{perm}(\text{REG}) = \text{perm}(\text{CF}) = \text{perm}(\text{PSL}) \]

**Corollary (Fernau et al. (2016))**

\[ \mathcal{L}(JFA) \text{ is closed under intersection and under complementation.} \]

**Example**

Standard complementation technique does not work for JFAs.

- For \( F = \{r\} \), it accepts all words that contain at least one \( a \).
- If \( F = \{s, t\} \), it accepts all words that contain at least one \( b \).
Theorem (Fernau et al. (2016))

\[ L(\alpha\text{-SHUF}) = L(JFA). \]

Proof Idea

\( \supseteq \): If \( L \in L(JFA) \), there exists regular \( L' \) such that \( L = \text{perm}(L') \). Then, \( \text{RE } R' \) denotes \( L' \). Then, we find an \( \alpha \)-SHUF expression \( R \) with \( L = \text{perm}(L(R')) = L(R) \).

\( \subseteq \): Let \( \alpha \)-SHUF expression \( R \) describes \( L \). Construct \( \text{RE } R' \) by replacing all \( \sqcup \) by \( \cdot \) and \( \sqcup, * \) by \( * \), so \( L(R) = \text{perm}(L(R')) \). As \( \text{perm}(L(R')) \in \text{REG} \), \( L \in L(JFA) \).
Theorem (Fernau et al. (2016))

\[ L(\alpha\text{-SHUF}) = L(JFA). \]

Proof Idea

\(\supseteq\): If \( L \in L(JFA) \), there exists regular \( L' \) such that \( L = \text{perm}(L') \). Then, \( \text{RE } R' \) denotes \( L' \). Then, we find an \( \alpha \)-SHUF expression \( R \) with \( L = \text{perm}(L(R')) = L(R) \).

\(\subseteq\): Let \( \alpha \)-SHUF expression \( R \) describes \( L \). Construct \( \text{RE } R' \) by replacing all \( \cdot\) by \( \cdot \) and \( \cdot,* \) by \( * \), so \( L(R) = \text{perm}(L(R')) \). As \( \text{perm}(L(R')) \in \text{REG} \), \( L \in L(JFA) \).

Corollary

\( L(JFA) \) is closed under iterated shuffle.
### Theorem (Fernau et al. (2016))

$L(GJFA)$ and $L(SHUF)$ are incomparable.

### Proof Idea

- Let $M = (\{s\}, \Sigma, \{sab \rightarrow s, scd \rightarrow s\}, s, \{s\})$. $L(M) \notin L(SHUF)$.
- $L(ac \uplus (bd)^\omega,*)$ is not accepted by any GJFA.
Theorem (Fernau et al. (2016))

\[ L(GJFA) \text{ and } L(SHUF) \text{ are incomparable.} \]

Proof Idea

- Let \( M = (\{s\}, \Sigma, \{sab \rightarrow s, scd \rightarrow s\}, s, \{s\}) \). \( L(M) \notin L(SHUF) \).
- \( L(ac \uplus (bd)^\omega,*) \) is not accepted by any GJFA.

Lemma (Fernau et al. (2016))

\[ \{ab\}^{\omega,*} \in (L(GJFA) \cap L(SHUF)) - L(JFA). \]
Theorem (Fernau et al. (2016))

\[ \mathcal{L}(GJFA) \text{ and } \mathcal{L}(SHUF) \text{ are incomparable.} \]

Proof Idea

- Let \( M = (\{s\}, \Sigma, \{sab \rightarrow s, scd \rightarrow s\}, s, \{s\}) \). \( L(M) \notin \mathcal{L}(SHUF) \).
- \( L(ac \uplus (bd)^\omega,*) \) is not accepted by any GJFA.

Lemma (Fernau et al. (2016))

\[ \{ab\}^\omega,* \in (\mathcal{L}(GJFA) \cap \mathcal{L}(SHUF)) - \mathcal{L}(JFA). \]

Theorem (Fernau et al. (2016))

\[ \mathcal{L}(JFA) = \text{perm}(\mathcal{L}(REG)) = \text{perm}(\mathcal{L}(CF)) = \text{perm}(\mathcal{L}(PSL)) = \text{perm}(\mathcal{L}(GJFA)) = \text{perm}(\mathcal{L}(SHUF)) \]
Relations Between Language Families II (Fernau et al., 2016)

\[
\begin{align*}
\text{SHUF} & \quad \uparrow \quad \text{PSL} \\
\text{SHUF} \cap \text{GJFA} & \quad \uparrow \\
\text{perm}(\text{GJFA}) & = \text{perm}(\text{SHUF}) = \\
\text{perm}(\text{PSL}) & = \text{JFA} = \alpha-\text{SHUF} \\
\text{REG} & \quad \uparrow \\
\text{REG} \cap \text{JFA} & \\
\text{CFL} & \quad \uparrow \quad \text{GJFA}
\end{align*}
\]
Closure Properties

Theorem (Vorel (2015), Theorem 2)

\( \mathcal{L}(\text{GJFA}) \) is not closed under Kleene star, Kleene plus, \( \varepsilon \)-free and general homomorphism and finite substitution.

Proof

• We have \( \{ab\} \in \mathcal{L}(\text{GJFA}) \), but \( \{ab\}^* \notin \mathcal{L}(\text{GJFA}) \).
• Since \( \mathcal{L}(\text{GJFA}) \) is closed under union, \( \{ab\}^+ \notin \mathcal{L}(\text{GJFA}) \).
• Consider \( \varepsilon \)-free homomorphism \( \varphi: \{a\}^* \to \{a, b\}^* \) with \( \varphi(a) = ab \).
• For \( L = \{a\}^* \in \mathcal{L}(\text{GJFA}) \), \( \varphi(L) = \{ab\}^* \notin \mathcal{L}(\text{GJFA}) \).
• In addition, \( \varphi \) is a general homomorphism and finite substitution as well.
## Closure Properties – Summary

<table>
<thead>
<tr>
<th>Operation</th>
<th>$\mathcal{L}(\text{GJFA})$</th>
<th>$\mathcal{L}(\text{JFA})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>union</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>intersection</td>
<td>- *(Vorel, 2015)</td>
<td>+</td>
</tr>
<tr>
<td>concatenation</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>intersection with reg. lang.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>complement</td>
<td>-</td>
<td>+ *(Fernau et al., 2016)</td>
</tr>
<tr>
<td>shuffle</td>
<td>- *(Vorel, 2015)</td>
<td>+</td>
</tr>
<tr>
<td>iterated shuffle</td>
<td>?</td>
<td>+ *(Fernau et al., 2016)</td>
</tr>
<tr>
<td>mirror image</td>
<td>+ *(Vorel, 2015)</td>
<td>+</td>
</tr>
<tr>
<td>Kleene star</td>
<td>- *(Vorel, 2015)</td>
<td>-</td>
</tr>
<tr>
<td>Kleene plus</td>
<td>- *(Vorel, 2015)</td>
<td>-</td>
</tr>
<tr>
<td>substitution</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>regular substitution</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>finite substitution</td>
<td>- *(Vorel, 2015)</td>
<td>-</td>
</tr>
<tr>
<td>homomorphism</td>
<td>- *(Vorel, 2015)</td>
<td>-</td>
</tr>
<tr>
<td>$\varepsilon$-free homomorphism</td>
<td>- *(Vorel, 2015)</td>
<td>-</td>
</tr>
<tr>
<td>inverse homomorphism</td>
<td>- *(Vorel, 2015)</td>
<td>+</td>
</tr>
</tbody>
</table>

Note: * marks corrections. (Meduna, Zemek, 2012) when the source is not specified.
### Decidability – Summary by Meduna, Zemek (2012)

<table>
<thead>
<tr>
<th></th>
<th>$L(GJFA)$</th>
<th>$L(JFA)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>membership</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>emptiness</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>finiteness</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>infiniteness</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
Theorem (Vorel [2016], Thm. 1)

*Given a GJFA \( M = (Q, \Sigma, R, s, F) \), it is *undecidable* whether \( L(M) = \Sigma^* \).*

**Proof Idea**

By reduction from universality of context-free grammar to the universality of GJFA.

Theorem (Vorel [2015], Thm. 6)

*Given GJFA \( M_1 \) and \( M_2 \) over an 8-letter alphabet, it is *undecidable* whether \( L(M_1) \cap L(M_2) = \emptyset \).*

**Proof Idea**

Using a prefix-disjoint instance of the Post correspondence problem over a range alphabet.
Decidability – Extended Summary

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{L}(\text{GJFA})$</th>
<th>$\mathcal{L}(\text{JFA})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>membership</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>emptiness</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>finiteness</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>infiniteness</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>universality</td>
<td>– (Vorel, 2016)</td>
<td>+ (Fernau et al., 2016)</td>
</tr>
<tr>
<td>disjointness</td>
<td>– (Vorel, 2016)$^1$</td>
<td>+ (Fernau et al., 2016)</td>
</tr>
</tbody>
</table>

$^1$GJFAs are over an 8-letter alphabet.
Note on Parsing of Fixed JFA

Scan over $w$ and store the current state and the Parikh mapping (as $\Sigma$ fixed, use working tape of non-det. logspace machine). Thus, $L(JFA) \subseteq NL \subseteq P$. 
Note on Parsing of Fixed JFA

Scan over $w$ and store the current state and the Parikh mapping (as $\Sigma$ fixed, use working tape of non-det. logspace machine). Thus, $L(JFA) \subseteq NL \subseteq P$.

Theorem (Fernau et al. (2016))

Unless ETH fails, there is no algorithm that, for given JFA $M$ with state set $Q$ and a given word $w$, decides whether $w \in L(M)$ and runs in time $O^*(2^{o(|Q|)})$.

Note on ETH

Often, Exponential Time Hypothesis (ETH) is used to state computational complexity results. If ETH holds, then $P \neq NP$. 
| Problem              | GJFA   | GJFA $|\Sigma| = k$ | JFA    | JFA $|\Sigma| = k$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed word</td>
<td>NP-C</td>
<td>NP-C*</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Universal word</td>
<td>NP-C</td>
<td>NP-C*</td>
<td>NP-C</td>
<td>P</td>
</tr>
<tr>
<td>Non-disjointness</td>
<td>Und.</td>
<td>Und.</td>
<td>NP-C</td>
<td>P</td>
</tr>
<tr>
<td>Non-universality</td>
<td>Und.</td>
<td>NP-H</td>
<td>NP-H</td>
<td>NP-C</td>
</tr>
</tbody>
</table>

Note: * marks results from (Fernau et al., 2016). NP-C = NP-complete; NP-H = NP-hard, membership in NP unknown; Und. = undecidable.
Open Problem Areas

- closure property of $\mathcal{L}(\text{GJFA})$ (iterated shuffle?)
- other decision problems of $\mathcal{L}(\text{GJFA})$ and $\mathcal{L}(\text{JFA})$, like equivalence and inclusion
- variants of JFA and GJFA (determinism, parallel, regulated, ...)

Jumping Finite Automata: New Results – Part One: Solved Questions
Thank you for your attention!

Part Two follows!