Jumping Finite Automata: Characterizations and Complexity

Henning Fernau, Meenakshi Paramasivan & Markus L. Schmid

{fernau, paramasivan, mschmid}@uni-trier.de

CIAA, Umeå, Sweden, 20 August 2015

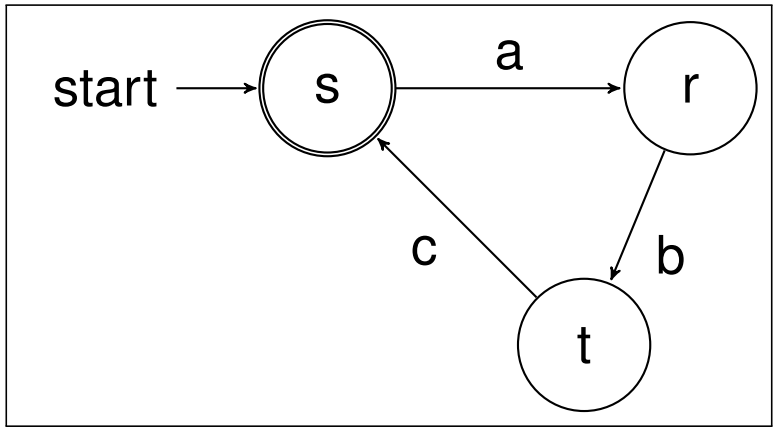
Overview on the talk

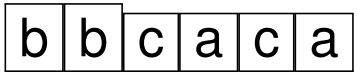
- Jumping Finite Automata
- Operations on Languages and their Properties
- Alphabetic Shuffle Expressions
- Representations and Normal Forms
- Comparing JFA and REG
- Complexity of Parsing
- Discussions

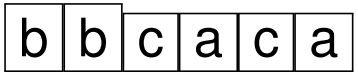
Jumping Finite Automata An Introduction . . .

- Jumping Finite Automata - [Meduna and Zemek 2012]
- Reads the input by making a jump at any position of the input
- Different jumping sequences are possible

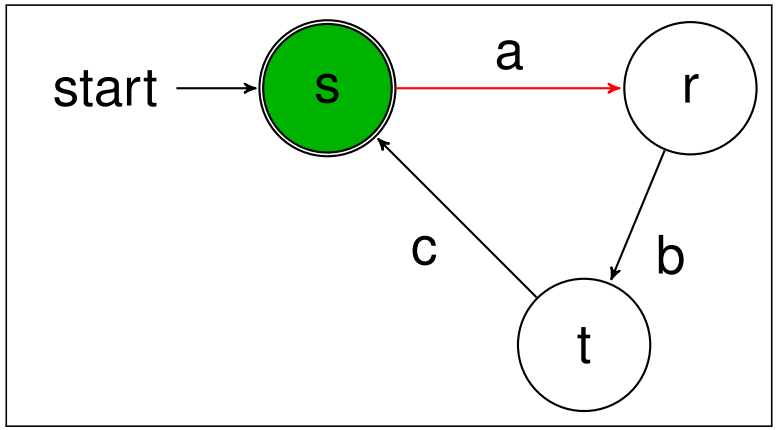
Jumping Finite Automata

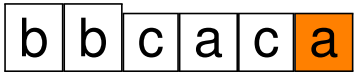




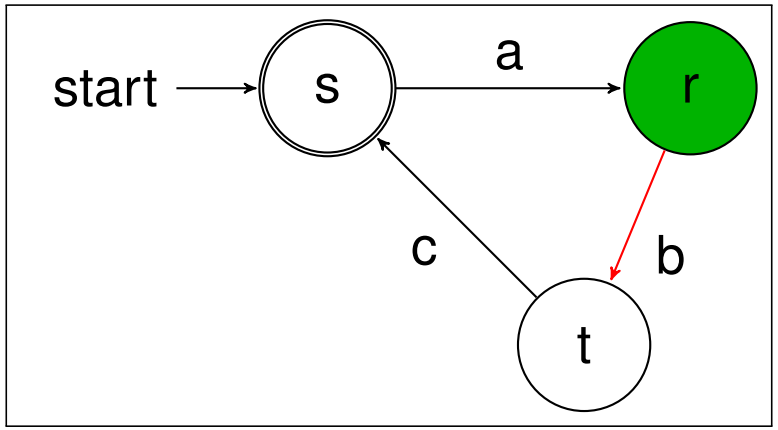


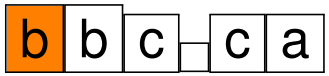
Jumping Finite Automata

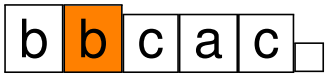




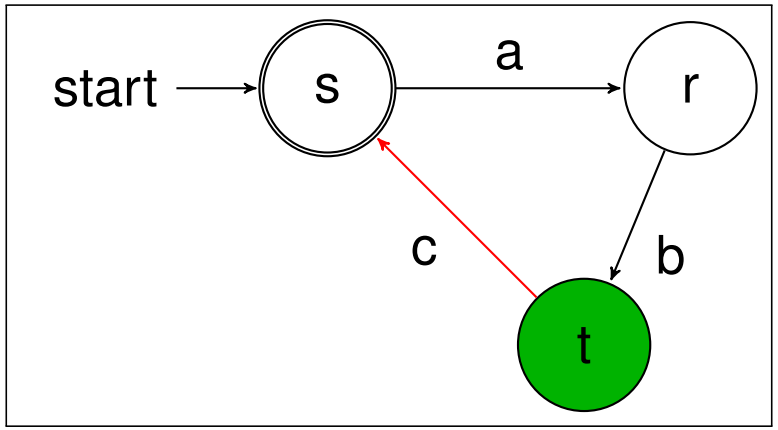
Jumping Finite Automata



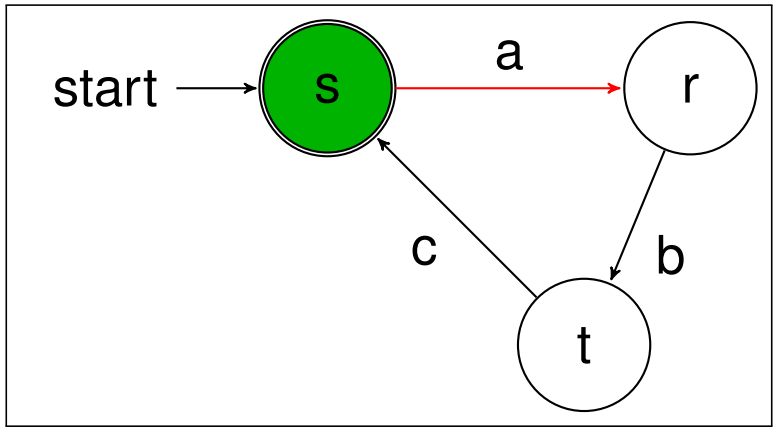




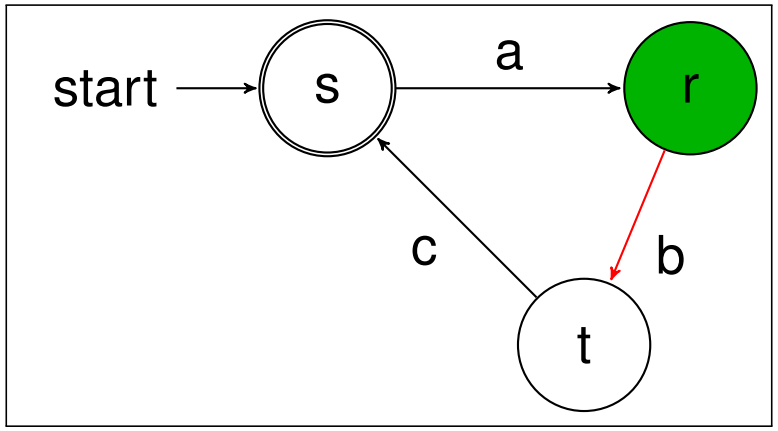
Jumping Finite Automata



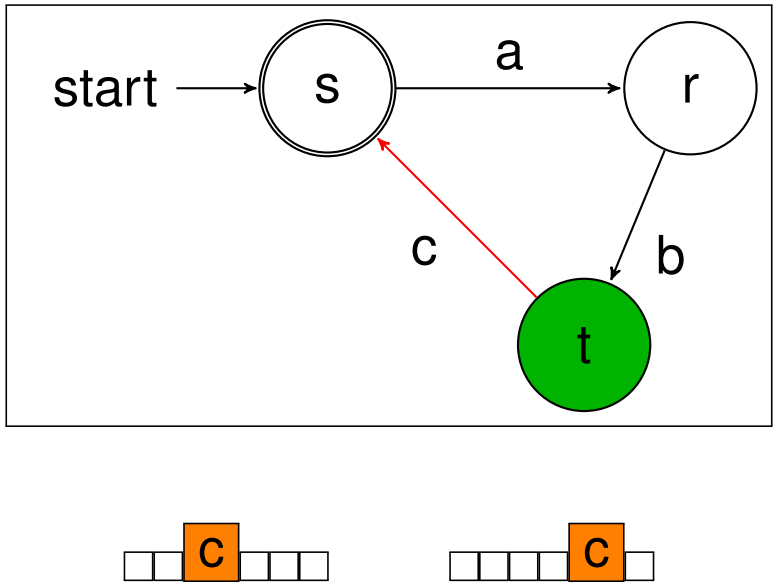
Jumping Finite Automata



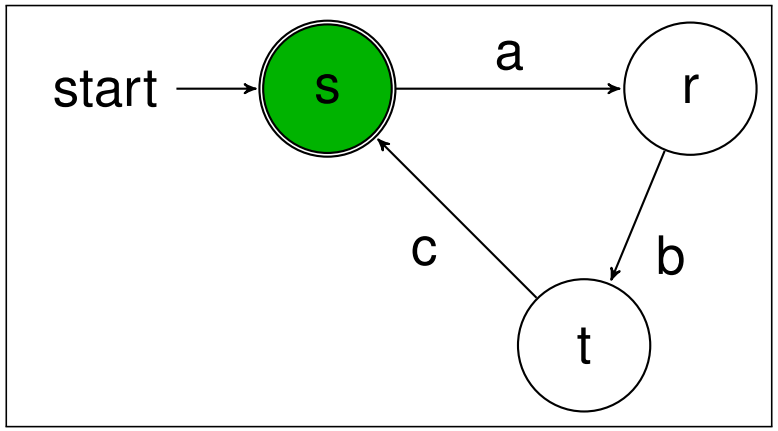
Jumping Finite Automata



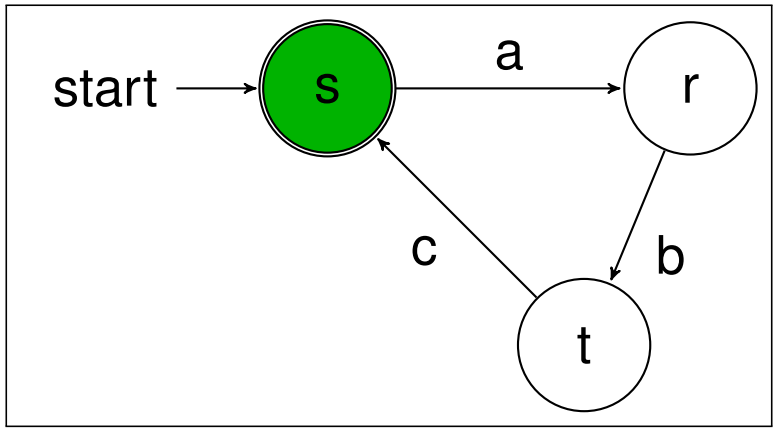
Jumping Finite Automata



Jumping Finite Automata



Jumping Finite Automata



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

Definitions

FM

A *finite machine* (FM) is a quintuple $M = (Q, \Sigma, R, s, F)$ where

- Q is a finite set of *states*
- Σ is the *input alphabet*
- R is a finite set of rules of the form $py \to q \quad (p, q \in Q, y \in \Sigma)$
- $s \in Q$ is the *start state*
- $F \subseteq Q$ is a set of *final states*

Definitions

Let $M = (Q, \Sigma, R, s, F)$, *finite machine*, we interpret M in two ways:

- *Configuration* of a FA, M is any string in $Q\Sigma^*$.

$$qw \Rightarrow pz \iff \exists\, qy \to p \in R\, \exists\, z \in \Sigma^* : w = yz$$

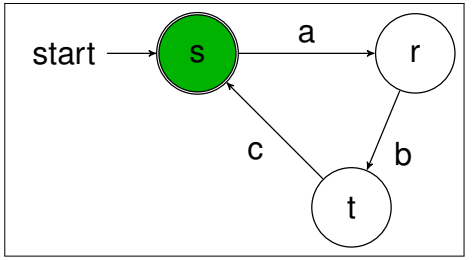
- *Configuration* of a JFA, M is any string in $\Sigma^* Q \Sigma^*$.

$$vqw \curvearrowright v'pz' \iff \exists\, qy \to p \in R\, \exists\, z \in \Sigma^* : w = yz \wedge vz = v'z'$$

$L_{FA}(M) = \{w \in \Sigma^* : \exists f \in F : sw \Rightarrow^* f\}$ and
$L_{JFA}(M) = \{w \in \Sigma^* : \exists u, v \in \Sigma^*\, \exists f \in F : w = uv \wedge usv \curvearrowright^* f\}$.

Our Example

The FM $M = (\{s, r, t\}, \{a, b, c\}, R, s, \{s\})$
with $R = \{sa \to r, rb \to t, tc \to s\}$ accepts
$L_{JFA}(M) = \{w \in \{a, b, c\}^* : |w|_a = |w|_b = |w|_c\}$
For instance:



$$
\begin{aligned}
bacbcsa &\curvearrowright bacrbc && [sa \to r] \\
&\curvearrowright bactc && [rb \to t] \\
&\curvearrowright bsac && [tc \to s] \\
&\curvearrowright rbc && [sa \to r] \\
&\curvearrowright tc && [rb \to t] \\
&\curvearrowright s && [tc \to s]
\end{aligned}
$$

Overview on the talk

- Jumping Finite Automata
- Operations on Languages and their Properties
- Alphabetic Shuffle Expressions
- Representations and Normal Forms
- Comparing JFA and REG
- Complexity of Parsing
- Discussions

Operations on Languages and their Properties

Shuffle Operation

The *shuffle operation*, denoted by $\sqcup\!\sqcup$, is defined recursively by
$(au \sqcup\!\sqcup bv) = a(u \sqcup\!\sqcup bv) \cup b(au \sqcup\!\sqcup v)$ and
$(u \sqcup\!\sqcup \varepsilon) = (\varepsilon \sqcup\!\sqcup u) = \{u\}$, where $u, v \in \Sigma^*$ and $a, b \in \Sigma$.
The shuffle operation is extended in a natural way to languages: the *shuffle* of two languages L_1 and L_2 is

$$L_1 \sqcup\!\sqcup L_2 = \bigcup_{u \in L_1, v \in L_2} u \sqcup\!\sqcup v$$

Iterated Shuffle

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

$$L^{\sqcup\!\sqcup,*} := \bigcup_{n=0}^{\infty} L^{\sqcup\!\sqcup,n} \text{ where } L^{\sqcup\!\sqcup,0} = \{\varepsilon\} \text{ and } L^{\sqcup\!\sqcup,i} := L^{\sqcup\!\sqcup,i-1} \sqcup\!\sqcup L.$$

Proposition 1

Let M_1, M_2, M_3 be arbitrary languages.

1. $M_1 \sqcup M_2 = M_2 \sqcup M_1$ (commutative law)
2. $(M_1 \sqcup M_2) \sqcup M_3 = M_1 \sqcup (M_2 \sqcup M_3)$ (associative law)
3. $M_1 \sqcup (M_2 \cup M_3) = M_1 \sqcup M_2 \cup M_1 \sqcup M_3$ (distributive law)
4. $(M_1 \cup M_2)^{\sqcup,*} = (M_1)^{\sqcup,*} \sqcup (M_2)^{\sqcup,*}$
5. $(M_1{}^{\sqcup,*})^{\sqcup,*} = (M_1)^{\sqcup,*}$
6. $(M_1 \sqcup M_2{}^{\sqcup,*})^{\sqcup,*} = (M_1 \sqcup (M_1 \cup M_2)^{\sqcup,*}) \cup \{\varepsilon\}$

perm(w)
The set of all permutations of w, perm(w), is defined as follows:

$$\text{perm}(w) = \begin{cases} \{\varepsilon\}, & |w| = 0 \\ \{a\} \sqcup\!\sqcup \text{perm}(u), & w = a \cdot u, a \in \Sigma, u \in \Sigma^* \end{cases}$$

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

Lemma 1
perm : $2^{\Sigma^*} \to 2^{\Sigma^*}$ is a hull operator, i.e., it is extensive, increasing and idempotent.

Lemma 2
The set $\{\text{perm}(w) : w \in \Sigma^*\}$ is a partition of Σ^*. $\{\text{perm}(w) : w \in \Sigma^*\} = \pi_\Sigma^-(\pi_\Sigma(w))$.

Note: Parikh mapping $\pi_\Sigma : \Sigma^* \to \mathbb{N}^\Sigma, w \mapsto (a \mapsto |w|_a)$

Proposition 3

For $L_1, L_2 \subseteq \Sigma^*$, $\mathsf{perm}(L_1) = \mathsf{perm}(L_2)$ iff $\pi_\Sigma(L_1) = \pi_\Sigma(L_2)$.

Corollary 1

If $L \in \mathscr{JFA}$, then L is perm-closed.

Theorem 1 (Meduna, Zemek & Parikh)

$\mathscr{JFA} = \mathsf{perm}(\mathscr{REG}) = \mathsf{perm}(\mathscr{CFL})$.

Proposition 2

$(2^{\Sigma^*}, \cup, \sqcup\!\sqcup, \emptyset, \{\varepsilon\})$ is a commutative semiring.

Theorem 2

$\mathsf{perm} : (2^{\Sigma^*}, \cup, \cdot, \emptyset, \{\varepsilon\}) \to (2^{\Sigma^*}, \cup, \sqcup\!\sqcup, \emptyset, \{\varepsilon\})$ is a semiring morphism that also respects the iterated catenation resp. shuffle operation.

Note: perm cannot be an isomorphism, as the catenation semiring is not commutative, while the shuffle semiring is.

Overview on the talk

- Jumping Finite Automata
- Operations on Languages and their Properties
- Alphabetic Shuffle Expressions
- Representations and Normal Forms
- Comparing JFA and REG
- Complexity of Parsing
- Discussions

Alphabetic Shuffle Expressions

α-SHUF expressions

- $\emptyset$, ϵ and each $a \in \Sigma$ are α-SHUF expressions,
- If S_1, S_2 are α-SHUF expressions,
  then $(S_1 + S_2)$, $(S_1 \sqcup\!\sqcup S_2)$ and ${S_1}^{\sqcup\!\sqcup, *}$ are α-SHUF expressions.

Semantics: $L((a+b)^{\sqcup\!\sqcup, *}) = \{a, b\}^{\sqcup\!\sqcup, *}$

SHUF expressions

- $\emptyset$, and each $w \in \Sigma^*$ are SHUF expressions,
- If S_1, S_2 are SHUF expressions,
  then $(S_1 + S_2)$, $(S_1 \sqcup\!\sqcup S_2)$ and ${S_1}^{\sqcup\!\sqcup, *}$ are SHUF expressions.

Note: More generally, in shuffle expressions also concatenation and Kleene star are admitted as operators.

Alphabetic Shuffle Expressions

Lemma 3

Let R' be a regular expression. Let the α-SHUF expression R be obtained from R' by consequently replacing all $\cdot$ by $\sqcup\!\sqcup$, and all * by $^{\sqcup\!\sqcup,*}$ in R'. Then, $\mathrm{perm}(L(R')) = L(R)$.

Theorem 3

A language $L \subseteq \Sigma^*$ is in $\mathscr{JFA}$ iff there is some α-SHUF expression R such that $L = L(R)$.

Corollary 3

$\mathscr{JFA}$ is closed under iterated shuffle.

Overview on the talk

- Jumping Finite Automata
- Operations on Languages and their Properties
- Alphabetic Shuffle Expressions
- Representations and Normal Forms
- Comparing JFA and REG
- Complexity of Parsing
- Discussions

Representations and Normal Forms

Theorem 4 (Representation Theorem)

Let $L \in \mathscr{JFA}$. Then $\exists\, n \geq 1$ and finite sets M_i, N_i for $1 \leq i \leq n$, so that the following representation is valid.

$$L = \bigcup_{i=1}^{n} \mathsf{perm}(M_i) \sqcup\!\sqcup (\mathsf{perm}(N_i))^{\sqcup\!\sqcup,*} \tag{1}$$

Proof sketch: We will prove this representation theorem on the level of α-SHUF expressions, so that we actually get a normal form theorem for these.

A central tool in the proof of the normal form theorem is the following notion that corresponds to the well-known star-height of regular expressions.

height of α-SHUF expression

We can inductively associate the (*shuffle iteration*) *height* h to any α-SHUF expression S as follows.

- If S is a base case, then $h(S) = 0$.
- If $S = (S_1 + S_2)$ or $S = (S_1 \sqcup\!\sqcup S_2)$, then $h(S) = \max\{h(S_1), h(S_2)\}$.
- If $S = S_1^{\sqcup\!\sqcup,*}$, then $h(S) = h(S_1) + 1$.

The shuffle iteration height of a $\mathscr{JFA}$-language L is then the smallest shuffle iteration height of any α-SHUF expression S describing L.

Corollary 4

$L \in \mathscr{JFA}$ iff there is a regular language R of star height at most one such that $L = \mathrm{perm}(R)$.

Corollary 2

Let L be a finite language. Then, $L \in \mathscr{JFA}$ if and only if L is perm-closed.

Proposition 4

Let L be some language. Then, L is finite and perm-closed if and only if there is an α-SHUF expression R, with $L = L(R)$, that does not contain the iterated shuffle operator.

Corollary 6

It is decidable, given some JFA and some integer k, whether or not this JFA describes a language of shuffle iteration height at most k.

Normal Form Theorem

Theorem 5

For any α-SHUF expression R, an equivalent α-SHUF expression S with $h(S) = 1$ can be constructed that is the union of n α-SHUF expressions $S_1, \ldots, S_n$ such that $S_i = F_i \sqcup\!\sqcup G_i^{\sqcup\!\sqcup,*}$, where $h(F_i) = h(G_i) = 0$, $1 \leq i \leq n$. Moreover, we can assume that $F_i = \bigcup_{j=1}^{n(i)} u_j$ and $G_i = \bigcup_{j=1}^{m(i)} v_j$, where all u_j and v_j are α-SHUF expressions with $\sqcup\!\sqcup$ as their only operators.

Proof Sketch:

1. By induction on the height of R.
2. Applying the laws from Proposition 1.

Overview on the talk

- Jumping Finite Automata
- Operations on Languages and their Properties
- Alphabetic Shuffle Expressions
- Representations and Normal Forms
- Comparing JFA and REG
- Complexity of Parsing
- Discussions

Comparing JFA and REG

Proposition 5

$L \in \mathscr{JFA} \cap \mathscr{REG}$ if and only if $L \in \mathscr{REG}$ and L is perm-closed.

Theorem 7

Let $L \subseteq \Sigma^*$. Then, $L \in \mathscr{JFA} \cap \mathscr{REG}$ if and only if there exists a number $n \geq 1$, words w_i and finite sets N_i for $1 \leq i \leq n$, where each N_i is given as $\bigcup_{a \in \Sigma_i} a^{n_i(a)}$ for some $\Sigma_i \subseteq \Sigma$ and some $n_i : \Sigma_i \to \mathbb{N}$, so that the following representation is valid.

$$L = \bigcup_{i=1}^{n} \mathsf{perm}(w_i) \sqcup\!\sqcup (\mathsf{perm}(N_i))^{\sqcup\!\sqcup,*}$$

Overview on the talk

- Jumping Finite Automata
- Operations on Languages and their Properties
- Alphabetic Shuffle Expressions
- Representations and Normal Forms
- Comparing JFA and REG
- Complexity of Parsing
- Discussions

Complexity of Parsing

Theorem 8

For any fixed input alphabet, the universal word problem for JFA is polynomial-time solvable.

Theorem 9, due to V. Vorel

The universal word problem for JFA (with unbounded input alphabet) is NP-complete.

(The CIAA proceedings contain a weaker result.)

Theorem 10

The universal word problem is polynomial-time solvable for α-SHUF expressions in normal form.

We have an NP-hardness result on the universal word problem for general α-SHUF expressions.

Overview on the talk

- Jumping Finite Automata
- Operations on Languages and their Properties
- Alphabetic Shuffle Expressions
- Representations and Normal Forms
- Comparing JFA and REG
- Complexity of Parsing
- Discussions

Discussions

- We related JFA to the area of expressions involving shuffle operators.
- Is there a characterization of the class of languages accepted by general jumping finite automata in terms of expressions?
- Several variants of jumping automata have been suggested in [Meduna & Zemek 2012] and more variants can be easily thought of.
- Automata that process the input by jumping at the begin or at the end of it, so, the input is accepted from both ends of the word. This resembles a specific form of revolving automata. [Bensch, Bordihn, Holzer & Kutrib 2009]
- Automata that process the input by jumping alternately from the left and from the right. This resembles a form of biautomata. [Klíma & Polák 2012]
- Hierarchies in [Flick & Kudlek 2012] inspires similar research for α-SHUF.
- As there is quite a number of variations and restrictions of the shuffle operation itself, it would be also interesting to study expressions that contain (some of) these. This is also something we are about to do in the near future.

Thank You!