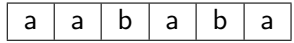
Algebraic Characterizations of Classes of Regular Languages in DynFO

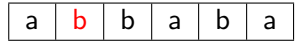
Corentin Barloy, Felix Tschirbs, Nils Vortmeier, Thomas Zeume

Incremental maintenance and DynFO



$\in (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

Incremental maintenance and DynFO

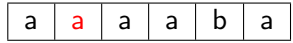


$\notin (aa)^*b(a+b)^*$

Incremental maintenance and DynFO

$\notin (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

Incremental maintenance and DynFO



$\in (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

Incremental maintenance and DynFO

a	a	a	a	b	a

$\in (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

$\rightarrow$ Descriptive approach DynFO: use tables updated by first-order formulas

Incremental maintenance and DynFO

a	a	a	a	b	a

$\in (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

$\rightarrow$ Descriptive approach DynFO: use tables updated by first-order formulas

$\rightarrow$ Table T: store whether of the parity of the number of as before every position is even.

Incremental maintenance and DynFO

a	a	a	a	b	a

$\in (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

$\rightarrow$ Descriptive approach DynFO: use tables updated by first-order formulas

$\rightarrow$ Table T: store whether of the parity of the number of as before every position is even.

0	1	0	1	1	0

Incremental maintenance and DynFO

a	a	a	a	b	a

$\in (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

$\rightarrow$ Descriptive approach DynFO: use tables updated by first-order formulas

$\rightarrow$ Table T: store whether of the parity of the number of as before every position is even.

0	1	0	1	1	0

$\rightarrow$ Update of j on change at i: $(j \geq i \Leftrightarrow \neg T(j))$

Incremental maintenance and DynFO

a	a	a	a	b	a

$\in (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

→ Descriptive approach DynFO: use tables updated by first-order formulas

→ Table T: store whether of the parity of the number of as before every position is even.

0	1	0	1	1	0

→ Update of j on change at i: $(j \geq i \Leftrightarrow \neg T(j))$

→ Output: $\exists j, \text{First-b}(j) \wedge T(j)$

Incremental maintenance and DynFO

a	b	a	a	b	a

$\notin (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

→ Descriptive approach DynFO: use tables updated by first-order formulas

→ Table T: store whether of the parity of the number of as before every position is even.

0	0	1	0	0	1

→ Update of j on change at i: $(j \geq i \Leftrightarrow \neg T(j))$

→ Output: $\exists j, \text{First-b}(j) \wedge T(j)$

Incremental maintenance and DynFO

a	b	a	a	b	a

$\notin (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

→ Descriptive approach DynFO: use tables updated by first-order formulas

→ Table T: store whether of the parity of the number of as before every position is even.

0	0	1	0	0	1

→ Update of j on change at i: $(j \geq i \Leftrightarrow \neg T(j))$

→ Output: $\exists j, \text{First-b}(j) \wedge T(j)$

→ This language is in DynFO but not in FO.

Incremental maintenance and DynFO

a	b	a	a	b	a

$\notin (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

→ Descriptive approach DynFO: use tables updated by first-order formulas
→ Table T: store whether of the parity of the number of as before every position is even.

0	0	1	0	0	1

→ Update of j on change at i: $(j \geq i \Leftrightarrow \neg T(j))$
→ Output: $\exists j, \text{First-b}(j) \wedge T(j)$
→ This language is in DynFO but not in FO.

Fine-grained analysis:

Incremental maintenance and DynFO

a	b	a	a	b	a

$\notin (aa)^*b(a+b)^*$

→ Descriptive approach DynFO: use tables updated by first-order formulas
→ Table T: store whether of the parity of the number of as before every position is even.

0	0	1	0	0	1

→ Update of j on change at i: $(j \geq i \Leftrightarrow \neg T(j))$
→ Output: $\exists j, \text{First-b}(j) \wedge T(j)$
→ This language is in DynFO but not in FO.

Fine-grained analysis:
→ Restricting alternations: DynProp, DynΣ_1, DynΣ_2, ...

Incremental maintenance and DynFO

a	b	a	a	b	a

$\notin (\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$

→ Descriptive approach DynFO: use tables updated by first-order formulas
→ Table T: store whether of the parity of the number of as before every position is even.

0	0	1	0	0	1

→ Update of j on change at i: $(j \geq i \Leftrightarrow \neg T(j))$
→ Output: $\exists j, \text{First-b}(j) \wedge T(j)$
→ This language is in DynFO but not in FO.

Fine-grained analysis:
→ Restricting alternations: DynProp, DynΣ_1, DynΣ_2, ...
→ Restricting tables arity: UDynProp, UDynΣ_1, UDynΣ_2, ...

Regular languages

Already studied: DynProp = Reg [Gelade, Marquardt, Schwentick 2012]

Regular languages

Already studied: DynProp = Reg [Gelade, Marquardt, Schwentick 2012]

The proof only uses binary tables for a DFA A:

Regular languages

Already studied: DynProp = Reg [Gelade, Marquardt, Schwentick 2012]

The proof only uses binary tables for a DFA A:
$\rightarrow$ for every pair of states p, q, $T_{p,q}$ stores all couples $j \leq k$ such that $w[j, k]$ ends in q starting from p.

Regular languages

Already studied: DynProp = Reg [Gelade, Marquardt, Schwentick 2012]

The proof only uses binary tables for a DFA A:
$\rightarrow$ for every pair of states p, q, $T_{p,q}$ stores all couples $j \leq k$ such that $w[j, k]$ ends in q starting from p.
$\rightarrow$ Output: disjunction of $T_{i,f}(\min, \max)$ for i initial and f final.

Regular languages

Already studied: DynProp = Reg [Gelade, Marquardt, Schwentick 2012]

The proof only uses binary tables for a DFA A:
→ for every pair of states p, q, $T_{p,q}$ stores all couples $j \leq k$ such that $w[j, k]$ ends in q starting from p.
→ Output: disjunction of $T_{i,f}(\min, \max)$ for i initial and f final.
→ Update of (j, k) for change a at i:
- → if $j \leq i \leq k$: disjunction over $r \xrightarrow{a} r'$ of $T_{p,r} \wedge T_{r',q}$.
- → else: do nothing.

Regular languages

Already studied: DynProp = Reg [Gelade, Marquardt, Schwentick 2012]

The proof only uses binary tables for a DFA A:
→ for every pair of states p, q, $T_{p,q}$ stores all couples $j \leq k$ such that $w[j, k]$ ends in q starting from p.
→ Output: disjunction of $T_{i,f}(\min, \max)$ for i initial and f final.
→ Update of (j, k) for change a at i:
- → if $j \leq i \leq k$: disjunction over $r \xrightarrow{a} r'$ of $T_{p,r} \wedge T_{r',q}$.
- → else: do nothing.

We restrict our attention to unary DynFO

Regular languages

Already studied: DynProp = Reg [Gelade, Marquardt, Schwentick 2012]

The proof only uses binary tables for a DFA A:
→ for every pair of states p, q, $T_{p,q}$ stores all couples $j \leq k$ such that $w[j, k]$ ends in q starting from p.
→ Output: disjunction of $T_{i,f}(\min, \max)$ for i initial and f final.
→ Update of (j, k) for change a at i:
- → if $j \leq i \leq k$: disjunction over $r \xrightarrow{a} r'$ of $T_{p,r} \wedge T_{r',q}$.
- → else: do nothing.

We restrict our attention to unary DynFO
Already known: Reg $\subseteq$ UDynFO [Hesse 2003]

Regular languages

Already studied: DynProp = Reg [Gelade, Marquardt, Schwentick 2012]

The proof only uses binary tables for a DFA A:
→ for every pair of states p, q, $T_{p,q}$ stores all couples $j \leq k$ such that $w[j, k]$ ends in q starting from p.
→ Output: disjunction of $T_{i,f}(\min, \max)$ for i initial and f final.
→ Update of (j, k) for change a at i:
- → if $j \leq i \leq k$: disjunction over $r \xrightarrow{a} r'$ of $T_{p,r} \wedge T_{r',q}$.
- → else: do nothing.

We restrict our attention to unary DynFO
Already known: Reg $\subseteq$ UDynFO [Hesse 2003]
→ We refine this with algebra!

The algebraic theory

finite automaton $\approx$ finite monoid $(M, \cdot)$

The algebraic theory

finite automaton $\approx$ finite monoid $(M, \cdot)$

finite set → M

associative operation → $\cdot$

The algebraic theory

finite set — associative operation

finite automaton $\approx$ finite monoid $(M, \cdot)$

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to L$ recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

The algebraic theory

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$

$\to L$ recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

The algebraic theory

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to L$ recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

$(\mathrm{aa})^*\mathrm{b}(\mathrm{a} + \mathrm{b})^*$ is recognized by $M = \{a, aa, aab, ab\}$.

The algebraic theory

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

$(\mathrm{aa})^*\mathrm{b}(\mathrm{a} + \mathrm{b})^*$ is recognized by $M = \{a, aa, aab, ab\}$.
$\to$ records if there is a b and the parity of the number of as before the first b.

The algebraic theory

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

$(\mathrm{aa})^*\mathrm{b}(\mathrm{a} + \mathrm{b})^*$ is recognized by $M = \{a, aa, aab, ab\}$.
$\to$ records if there is a b and the parity of the number of as before the first b.
$\to$ The operation is defined accordingly. (ex: $a \cdot aab = ab$ and $aab \cdot a = aab$)

The algebraic theory

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

$(\mathrm{aa})^*\mathrm{b}(\mathrm{a} + \mathrm{b})^*$ is recognized by $M = \{a, aa, aab, ab\}$.
$\to$ records if there is a b and the parity of the number of as before the first b.
$\to$ The operation is defined accordingly. (ex: $a \cdot aab = ab$ and $aab \cdot a = aab$)

Syntactic monoid of a language: smallest monoid recognizing it.

The algebraic theory

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

$(\mathrm{aa})^*\mathrm{b}(\mathrm{a} + \mathrm{b})^*$ is recognized by $M = \{a, aa, aab, ab\}$.
$\to$ records if there is a b and the parity of the number of as before the first b.
$\to$ The operation is defined accordingly. (ex: $a \cdot aab = ab$ and $aab \cdot a = aab$)

Syntactic monoid of a language: smallest monoid recognizing it.
$\to$ Given its minimal automaton, it corresponds to the set of transition functions with composition.

The algebraic theory

finite set — associative operation

finite automaton $\approx$ finite monoid $(M, \cdot)$

recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

$(\mathrm{aa})^*\mathrm{b}(\mathrm{a} + \mathrm{b})^*$ is recognized by $M = \{a, aa, aab, ab\}$.
$\to$ records if there is a b and the parity of the number of as before the first b.
$\to$ The operation is defined accordingly. (ex: $a \cdot aab = ab$ and $aab \cdot a = aab$)

Syntactic monoid of a language: smallest monoid recognizing it.
$\to$ Given its minimal automaton, it corresponds to the set of transition functions with composition.
$\to$ Exhibit algebraic properties instead of combinatorial ones.

The algebraic theory

finite set — associative operation

finite automaton $\approx$ finite monoid $(M, \cdot)$

recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

$(\mathrm{aa})^*\mathrm{b}(\mathrm{a} + \mathrm{b})^*$ is recognized by $M = \{a, aa, aab, ab\}$.
$\to$ records if there is a b and the parity of the number of as before the first b.
$\to$ The operation is defined accordingly. (ex: $a \cdot aab = ab$ and $aab \cdot a = aab$)

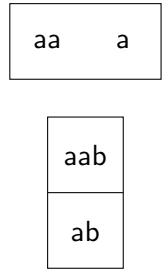
Syntactic monoid of a language: smallest monoid recognizing it.
$\to$ Given its minimal automaton, it corresponds to the set of transition functions with composition.
$\to$ Exhibit algebraic properties instead of combinatorial ones.

Previous proof of Reg $\subseteq$ DynFO: Maintain the evaluation of infixes in a monoid.

The regular languages of UDynΣ_2

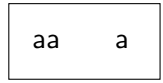
Ideal structure

A division-based representation of the syntactic monoid of $(\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$:

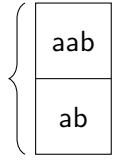


Ideal structure

A division-based representation of the syntactic monoid of $(\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$:

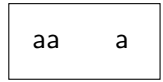


$\mathcal{J}$-classes: set of x with same MxM

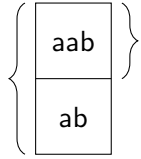


Ideal structure

A division-based representation of the syntactic monoid of $(\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$:



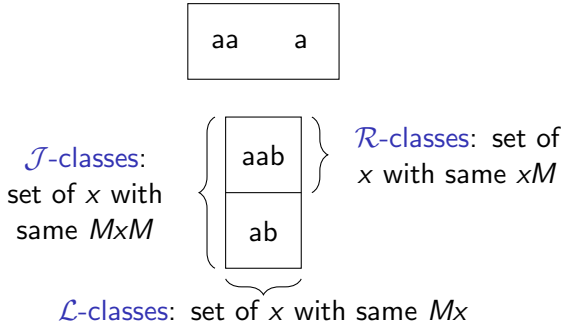
$\mathcal{J}$-classes: set of x with same MxM



$\mathcal{R}$-classes: set of x with same xM

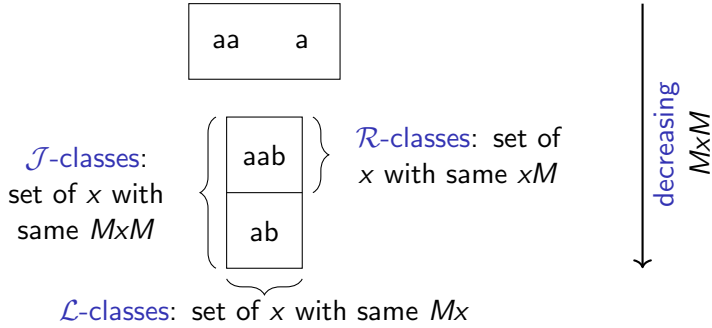
Ideal structure

A division-based representation of the syntactic monoid of $(\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$:



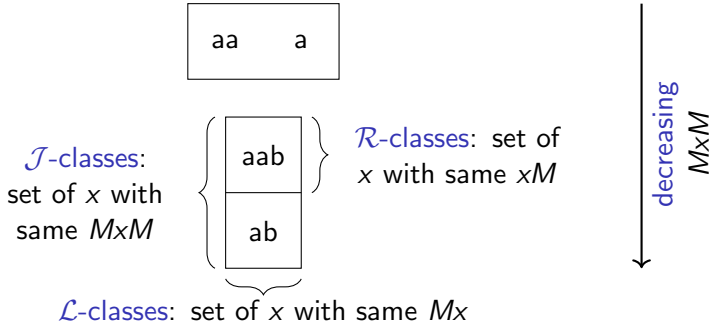
Ideal structure

A division-based representation of the syntactic monoid of $(\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$:



Ideal structure

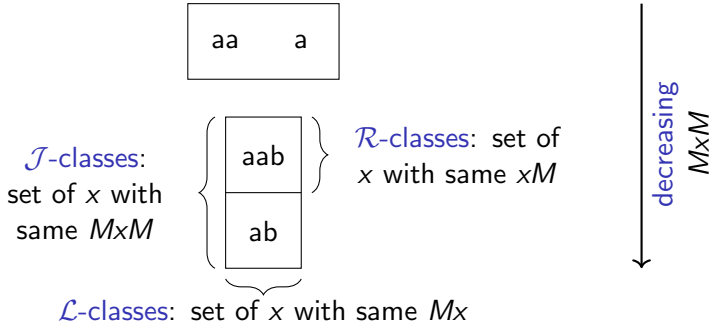
A division-based representation of the syntactic monoid of $(\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$:



$\rightarrow$ If an infix evaluates in J, then the whole word evaluates $\geq J$.

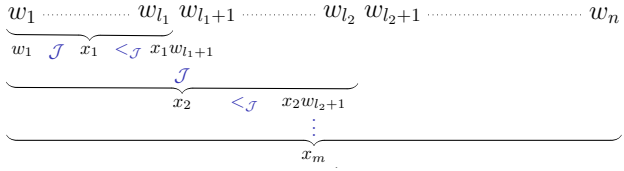
Ideal structure

A division-based representation of the syntactic monoid of $(\mathtt{aa})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*$:



$\rightarrow$ If an infix evaluates in J, then the whole word evaluates $\geq J$.

$\rightarrow$ Every word has a decomposition of the form:



The power of UDynΣ_2

Theorem
All regular languages are in UDynΣ_2.

The power of UDynΣ_2

Theorem
All regular languages are in UDynΣ_2.

Proof sketch: We show that we can compute the evaluation in any monoid M.

The power of UDynΣ_2

Theorem
All regular languages are in UDynΣ_2.

Proof sketch: We show that we can compute the evaluation in any monoid M.

A table $R_{J,x}$ for all $\mathcal{J}$-class J and $x \in M$.
$\rightarrow$ contains i iff the greatest infix in J starting at i evaluates to x.
$\rightarrow$ Similarly: tables $L_{J,x}$

The power of UDynΣ_2

Theorem
All regular languages are in UDynΣ_2.

Proof sketch: We show that we can compute the evaluation in any monoid M.

A table $R_{J,x}$ for all $\mathcal{J}$-class J and $x \in M$.
$\rightarrow$ contains i iff the greatest infix in J starting at i evaluates to x.
$\rightarrow$ Similarly: tables $L_{J,x}$

We can check if $w[i,j]$ evaluates to x in Σ_2:
$\exists i = l_1 < \cdots < l_m = j$,
- $\rightarrow \forall l_k \leq j < l_{k+1}$, there is no jump in $\mathcal{J}$-class at j (thanks to L)
- $\rightarrow$ there is a jump in $\mathcal{J}$-class at each l_k (thanks to L)
- $\rightarrow$ the overall evaluation is x (thanks to R, and more work!)

The power of UDynΣ_2

Theorem
All regular languages are in UDynΣ_2.

Proof sketch: We show that we can compute the evaluation in any monoid M.

A table $R_{J,x}$ for all $\mathcal{J}$-class J and $x \in M$.
$\rightarrow$ contains i iff the greatest infix in J starting at i evaluates to x.
$\rightarrow$ Similarly: tables $L_{J,x}$

We can check if $w[i,j]$ evaluates to x in Σ_2:
$\exists i = l_1 < \cdots < l_m = j$,
$\rightarrow \forall l_k \leq j < l_{k+1}$, there is no jump in $\mathcal{J}$-class at j (thanks to L)
$\rightarrow$ there is a jump in $\mathcal{J}$-class at each l_k (thanks to L)
$\rightarrow$ the overall evaluation is x (thanks to R, and more work!)

Thus we can answer membership in Σ_2

The power of UDynΣ_2

Theorem
All regular languages are in UDynΣ_2.

Proof sketch: We show that we can compute the evaluation in any monoid M.

A table $R_{J,x}$ for all $\mathcal{J}$-class J and $x \in M$.
$\rightarrow$ contains i iff the greatest infix in J starting at i evaluates to x.
$\rightarrow$ Similarly: tables $L_{J,x}$

We can check if $w[i,j]$ evaluates to x in Σ_2:
$\exists i = l_1 < \cdots < l_m = j$,
- $\rightarrow \forall l_k \leq j < l_{k+1}$, there is no jump in $\mathcal{J}$-class at j (thanks to L)
- $\rightarrow$ there is a jump in $\mathcal{J}$-class at each l_k (thanks to L)
- $\rightarrow$ the overall evaluation is x (thanks to R, and more work!)

Thus we can answer membership in Σ_2
Updates of $R_{J,x}$ at i: there is an index j such that $w[i,j]$ evaluates to x and $w[i,j+1]$ is $> J$.

The regular languages of subclasses of UDynProp

Varieties

We used: maintain monoids $\Rightarrow$ maintain all recognized languages

Varieties

We used: maintain monoids $\Rightarrow$ maintain all recognized languages
$\rightarrow$ We want the converse.

Varieties

We used: maintain monoids $\Rightarrow$ maintain all recognized languages
$\rightarrow$ We want the converse.

Definition (Variety)

A set of languages is a variety if it is closed under:

- Boolean operations ($\cup$, $\cap$ and L^c),
- quotients ($a^{-1}L$ and La^{-1}),
- inverse morphisms ($\mu^{-1}(L)$).

Varieties

We used: maintain monoids $\Rightarrow$ maintain all recognized languages
$\rightarrow$ We want the converse.

Definition (Variety)

A set of languages is a variety if it is closed under:

- Boolean operations ($\cup$, $\cap$ and L^c),
- quotients ($a^{-1}L$ and La^{-1}),
- inverse morphisms ($\mu^{-1}(L)$).

UDynProp is a variety.

Varieties

We used: maintain monoids $\Rightarrow$ maintain all recognized languages
$\rightarrow$ We want the converse.

Definition (Variety)

A set of languages is a variety if it is closed under:

- Boolean operations ($\cup$, $\cap$ and L^c),
- quotients ($a^{-1}L$ and La^{-1}),
- inverse morphisms ($\mu^{-1}(L)$).

UDynProp is a variety.

Theorem

Membership of a language in UDynFO only depends on its syntactic monoid.

Varieties

We used: maintain monoids $\Rightarrow$ maintain all recognized languages
$\rightarrow$ We want the converse.

Definition (Variety)

A set of languages is a variety if it is closed under:

- Boolean operations ($\cup$, $\cap$ and L^c),
- quotients ($a^{-1}L$ and La^{-1}),
- inverse morphisms ($\mu^{-1}(L)$).

UDynProp is a variety.

Theorem

Membership of a language in UDynFO only depends on its syntactic monoid.

Usefull to define a class of languages by their syntactic monoids

Varieties

We used: maintain monoids $\Rightarrow$ maintain all recognized languages
$\rightarrow$ We want the converse.

Definition (Variety)

A set of languages is a variety if it is closed under:

- Boolean operations ($\cup$, $\cap$ and L^c),
- quotients ($a^{-1}L$ and La^{-1}),
- inverse morphisms ($\mu^{-1}(L)$).

UDynProp is a variety.

Theorem

Membership of a language in UDynFO only depends on its syntactic monoid.

Usefull to define a class of languages by their syntactic monoids
$\rightarrow$ **G** is the class of languages whose syntactic monoid is a group

The power of UDynProp

Theorem
UDynProp = **G**

The power of UDynProp

Theorem
UDynProp $=$ **G**

Upper bound: In a group, the evaluation of $w[i,j]$ only depends on $w[1,i]$ and $w[1,j]$.

The power of UDynProp

Theorem
UDynProp = **G**

Upper bound: In a group, the evaluation of $w[i, j]$ only depends on $w[1, i]$ and $w[1, j]$.
$\rightarrow$ Only maintain evaluation prefixes is enough to retrieve all infixes

The power of UDynProp

Theorem
UDynProp = **G**

Upper bound: In a group, the evaluation of $w[i,j]$ only depends on $w[1,i]$ and $w[1,j]$.
$\rightarrow$ Only maintain evaluation prefixes is enough to retrieve all infixes
$\rightarrow$ the naive algorithm can be improved with unary tables.

The power of UDynProp

Theorem
UDynProp = **G**

Upper bound: In a group, the evaluation of $w[i, j]$ only depends on $w[1, i]$ and $w[1, j]$.
$\rightarrow$ Only maintain evaluation prefixes is enough to retrieve all infixes
$\rightarrow$ the naive algorithm can be improved with unary tables.

Lower bound: How are the monoids that are not groups?

The power of UDynProp

Theorem
UDynProp = **G**

Upper bound: In a group, the evaluation of $w[i, j]$ only depends on $w[1, i]$ and $w[1, j]$.
$\rightarrow$ Only maintain evaluation prefixes is enough to retrieve all infixes
$\rightarrow$ the naive algorithm can be improved with unary tables.

Lower bound: How are the monoids that are not groups?
$\rightarrow$ M is a group $\Leftrightarrow$ the identity is the only x such that $x^2 = x$

The power of UDynProp

Theorem
UDynProp = **G**

Upper bound: In a group, the evaluation of $w[i,j]$ only depends on $w[1,i]$ and $w[1,j]$.
$\rightarrow$ Only maintain evaluation prefixes is enough to retrieve all infixes
$\rightarrow$ the naive algorithm can be improved with unary tables.

Lower bound: How are the monoids that are not groups?
$\rightarrow$ M is a group $\Leftrightarrow$ the identity is the only x such that $x^2 = x$
$\rightarrow$ Take $M \notin \mathbf{G}$ and $x \neq 1$ such that $x^2 = x$

The power of UDynProp

Theorem
UDynProp = **G**

Upper bound: In a group, the evaluation of $w[i,j]$ only depends on $w[1,i]$ and $w[1,j]$.
$\rightarrow$ Only maintain evaluation prefixes is enough to retrieve all infixes
$\rightarrow$ the naive algorithm can be improved with unary tables.

Lower bound: How are the monoids that are not groups?
$\rightarrow$ M is a group $\Leftrightarrow$ the identity is the only x such that $x^2 = x$
$\rightarrow$ Take $M \notin \mathbf{G}$ and $x \neq 1$ such that $x^2 = x$
$\rightarrow$ Consider $\mu : \{a,b\}^* \rightarrow M$ such that $\mu(b) = 1$ and $\mu(a) = x$

The power of UDynProp

Theorem
UDynProp = **G**

Upper bound: In a group, the evaluation of $w[i,j]$ only depends on $w[1,i]$ and $w[1,j]$.
$\rightarrow$ Only maintain evaluation prefixes is enough to retrieve all infixes
$\rightarrow$ the naive algorithm can be improved with unary tables.

Lower bound: How are the monoids that are not groups?
$\rightarrow$ M is a group $\Leftrightarrow$ the identity is the only x such that $x^2 = x$
$\rightarrow$ Take $M \notin \mathbf{G}$ and $x \neq 1$ such that $x^2 = x$
$\rightarrow$ Consider $\mu : \{a, b\}^* \rightarrow M$ such that $\mu(b) = 1$ and $\mu(a) = x$
$\rightarrow$ It recognizes $(\mathtt{a} + \mathtt{b})^* \mathtt{a} (\mathtt{a} + \mathtt{b})^*$

The power of UDynProp

Theorem
UDynProp = **G**

Upper bound: In a group, the evaluation of $w[i,j]$ only depends on $w[1,i]$ and $w[1,j]$.
→ Only maintain evaluation prefixes is enough to retrieve all infixes
→ the naive algorithm can be improved with unary tables.

Lower bound: How are the monoids that are not groups?
→ M is a group ⇔ the identity is the only x such that $x^2 = x$
→ Take $M \notin \mathbf{G}$ and $x \neq 1$ such that $x^2 = x$
→ Consider $\mu : \{a, b\}^* \to M$ such that $\mu(b) = 1$ and $\mu(a) = x$
→ It recognizes $(\mathtt{a} + \mathtt{b})^*\mathtt{a}(\mathtt{a} + \mathtt{b})^*$
→ This language in not in UDynProp [Schwentick, Zeume 2015]

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
UDyn$\Sigma_1^+ \cap$ Reg $= \mathbf{J}^+ * \mathbf{G}$

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
UDyn$\Sigma_1^+ \cap \mathrm{Reg} = \mathbf{J}^+ * \mathbf{G}$

$\rightarrow$ $\mathbf{J}^+$: class of languages definable in Σ_1

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
UDyn$\Sigma_1^+ \cap \mathrm{Reg} = \mathbf{J}^+ * \mathbf{G}$

$\rightarrow$ $\mathbf{J}^+$: class of languages definable in Σ_1
$\rightarrow$ not closed under complement: it is a positive variety of ordered monoid

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
UDyn$\Sigma_1^+ \cap \mathrm{Reg} = \mathbf{J}^+ * \mathbf{G}$

$\rightarrow$ $\mathbf{J}^+$: class of languages definable in Σ_1
$\rightarrow$ not closed under complement: it is a positive variety of ordered monoid
$\rightarrow$ $*$: sequential composition of automata

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
UDyn$\Sigma_1^+ \cap \mathrm{Reg} = \mathbf{J}^+ * \mathbf{G}$

$\rightarrow$ $\mathbf{J}^+$: class of languages definable in Σ_1
$\rightarrow$ not closed under complement: it is a positive variety of ordered monoid
$\rightarrow$ $*$: sequential composition of automata
$\rightarrow$ algebraic counterpart: wreath product

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
UDyn$\Sigma_1^+ \cap \mathrm{Reg} = \mathbf{J}^+ * \mathbf{G}$

$\rightarrow$ $\mathbf{J}^+$: class of languages definable in Σ_1
$\rightarrow$ not closed under complement: it is a positive variety of ordered monoid
$\rightarrow$ $*$: sequential composition of automata
$\rightarrow$ algebraic counterpart: wreath product

Upper bound: we can maintain with Prop the evaluation in a group of all prefixes

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
UDyn$\Sigma_1^+ \cap \mathrm{Reg} = \mathbf{J}^+ * \mathbf{G}$

$\rightarrow$ $\mathbf{J}^+$: class of languages definable in Σ_1
$\rightarrow$ not closed under complement: it is a positive variety of ordered monoid
$\rightarrow$ $*$: sequential composition of automata
$\rightarrow$ algebraic counterpart: wreath product

Upper bound: we can maintain with Prop the evaluation in a group of all prefixes
$\rightarrow$ we can maintain the word labeled by a group

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
UDyn$\Sigma_1^+ \cap \mathrm{Reg} = \mathbf{J}^+ * \mathbf{G}$

$\rightarrow$ $\mathbf{J}^+$: class of languages definable in Σ_1
$\rightarrow$ not closed under complement: it is a positive variety of ordered monoid
$\rightarrow$ $*$: sequential composition of automata
$\rightarrow$ algebraic counterpart: wreath product

Upper bound: we can maintain with Prop the evaluation in a group of all prefixes
$\rightarrow$ we can maintain the word labeled by a group
$\rightarrow$ a Σ_1^+ formula can take care of the $\mathbf{J}^+$ part

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
UDyn$\Sigma_1^+ \cap \mathrm{Reg} = \mathbf{J}^+ * \mathbf{G}$

$\rightarrow$ $\mathbf{J}^+$: class of languages definable in Σ_1
$\rightarrow$ not closed under complement: it is a positive variety of ordered monoid
$\rightarrow$ $*$: sequential composition of automata
$\rightarrow$ algebraic counterpart: wreath product

Upper bound: we can maintain with Prop the evaluation in a group of all prefixes
$\rightarrow$ we can maintain the word labeled by a group
$\rightarrow$ a Σ_1^+ formula can take care of the $\mathbf{J}^+$ part

Lower bound: lot of work on wreath product by $\mathbf{G}$ [Almeida, Escada 2002] [Pin, Weil 2002]

The power of $\mathrm{UDyn}\Sigma_1^+$

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
$\mathrm{UDyn}\Sigma_1^+ \cap \mathrm{Reg} = \mathbf{J}^+ * \mathbf{G}$

$\rightarrow$ $\mathbf{J}^+$: class of languages definable in Σ_1
$\rightarrow$ not closed under complement: it is a positive variety of ordered monoid
$\rightarrow$ $*$: sequential composition of automata
$\rightarrow$ algebraic counterpart: wreath product

Upper bound: we can maintain with Prop the evaluation in a group of all prefixes
$\rightarrow$ we can maintain the word labeled by a group
$\rightarrow$ a Σ_1^+ formula can take care of the $\mathbf{J}^+$ part

Lower bound: lot of work on wreath product by $\mathbf{G}$ [Almeida, Escada 2002] [Pin, Weil 2002]
$\rightarrow$ If any ordered monoid not in $\mathbf{J}^+ * \mathbf{G}$ recognizes b^*

The power of UDynΣ_1^+

Σ_1^+: formulas of the form $\exists x_1, \cdots, x_k,\ \varphi$ where φ has no negations.
$\rightarrow$ All formulas used so far had no negations.

Theorem
UDyn$\Sigma_1^+ \cap \mathrm{Reg} = \mathbf{J}^+ * \mathbf{G}$

$\rightarrow$ $\mathbf{J}^+$: class of languages definable in Σ_1
$\rightarrow$ not closed under complement: it is a positive variety of ordered monoid
$\rightarrow$ $*$: sequential composition of automata
$\rightarrow$ algebraic counterpart: wreath product

Upper bound: we can maintain with Prop the evaluation in a group of all prefixes
$\rightarrow$ we can maintain the word labeled by a group
$\rightarrow$ a Σ_1^+ formula can take care of the $\mathbf{J}^+$ part

Lower bound: lot of work on wreath product by $\mathbf{G}$ [Almeida, Escada 2002] [Pin, Weil 2002]
$\rightarrow$ If any ordered monoid not in $\mathbf{J}^+ * \mathbf{G}$ recognizes b^*
$\rightarrow$ This language can be shown to not be in UDynΣ_1^+

Conclusion

The only problem left is to identify the regular languages of $\mathrm{UDyn}\Sigma_1$.

Conclusion

The only problem left is to identify the regular languages of $\mathsf{UDyn}\Sigma_1$.

$\rightarrow$ Strictly more powerful than $\mathsf{UDyn}\Sigma_1^+$: witnessed by $\mathtt{b}^*$ and $(\mathtt{a}+\mathtt{b})^*\mathtt{aa}(\mathtt{a}+\mathtt{b})^*$.

Conclusion

The only problem left is to identify the regular languages of $\mathrm{UDyn}\Sigma_1$.

$\rightarrow$ Strictly more powerful than $\mathrm{UDyn}\Sigma_1^+$: witnessed by $\mathtt{b}^*$ and $(\mathtt{a}+\mathtt{b})^*\mathtt{aa}(\mathtt{a}+\mathtt{b})^*$.

$\rightarrow$ Conjecture: $\mathbf{J} * \mathbf{G} \subsetneq \mathrm{UDyn}\Sigma_1 \subseteq \mathbf{\Sigma_2} * \mathbf{G}$

Conclusion

The only problem left is to identify the regular languages of $\mathrm{UDyn}\Sigma_1$.

$\rightarrow$ Strictly more powerful than $\mathrm{UDyn}\Sigma_1^+$: witnessed by $\mathtt{b}^*$ and $(\mathtt{a}+\mathtt{b})^*\mathtt{aa}(\mathtt{a}+\mathtt{b})^*$.

$\rightarrow$ Conjecture: $\mathbf{J} * \mathbf{G} \subsetneq \mathrm{UDyn}\Sigma_1 \subseteq \mathbf{\Sigma_2} * \mathbf{G}$

$\rightarrow$ We lack lower bounds!

Conclusion

The only problem left is to identify the regular languages of $\mathrm{UDyn}\Sigma_1$.

$\rightarrow$ Strictly more powerful than $\mathrm{UDyn}\Sigma_1^+$: witnessed by b^* and $(a+b)^*aa(a+b)^*$.

$\rightarrow$ Conjecture: $\mathbf{J} * \mathbf{G} \subsetneq \mathrm{UDyn}\Sigma_1 \subseteq \mathbf{\Sigma_2} * \mathbf{G}$

$\rightarrow$ We lack lower bounds!

Recap:

Dynamic class	UDynProp	$\mathrm{UDyn}\Sigma_1^+$	$\mathrm{UDyn}\Sigma_2$
Regular languages	$\mathbf{G}$	$\mathbf{J}^+ * \mathbf{G}$	Reg