

The height of piecewise-testable languages with applications in logical complexity

Prateek Karandikar & Philippe Schnoebelen
CMI, Chennai & LSV, ENS Cachan

Sep. 1st, 2016 — CSL 2016 — Marseilles

Motivations: The subword/subsequence relation

We consider **finite words** over a finite alphabet A .

Motivations: The subword/subsequence relation

We consider **finite words** over a finite alphabet A .

We use \sqsubseteq to denote the **(scattered) subword** relation on words.

Example:

$$abba \sqsubseteq abracadabra$$

Motivations: The subword/subsequence relation

We consider **finite words** over a finite alphabet A .

We use \sqsubseteq to denote the **(scattered) subword** relation on words.

Example:

$$abba \sqsubseteq abracadabra$$

Appears in combinatorics of words, language theory, bioinformatics, coding theory, ...

\implies beautiful algorithms for **longest common subsequence** between two words, for **counting embeddings** of a word in a text, etc.

Motivations: The subword/subsequence relation

We consider **finite words** over a finite alphabet A .

We use \sqsubseteq to denote the (**scattered**) **subword** relation on words.

Example:

$$abba \sqsubseteq abracadabra$$

Appears in combinatorics of words, language theory, bioinformatics, coding theory, ...

\implies beautiful algorithms for **longest common subsequence** between two words, for **counting embeddings** of a word in a text, etc.

But: **not** considered in logics of strings, constraint solvers, etc. ??
(unlike e.g. prefix ordering)

How do we reason about subwords?

First-Order Logic of Subwords

For a finite alphabet A , consider the first-order logic $\text{FO}(A^*, \sqsubseteq)$.

Formulae are built with:

- ▶ Variables, x, y, \dots
- ▶ Constant words $\epsilon, abc, bbcac, \dots$
- ▶ Atomic formulae $x \sqsubseteq y, x = aabc, \dots$
- ▶ Boolean operations \neg, \vee, \wedge
- ▶ Quantification $\forall x, \exists x$.

First-Order Logic of Subwords

For a finite alphabet A , consider the first-order logic $\text{FO}(A^*, \sqsubseteq)$.

Formulae are built with:

- ▶ Variables, x, y, \dots
- ▶ Constant words $\epsilon, abc, bbcac, \dots$
- ▶ Atomic formulae $x \sqsubseteq y, x = aabc, \dots$
- ▶ Boolean operations \neg, \vee, \wedge
- ▶ Quantification $\forall x, \exists x$.

$$\forall x, y (x \sqsubseteq y \vee y \sqsubseteq x)$$

True if and only if $|A| \leq 1$.

First-Order Logic of Subwords

For a finite alphabet A , consider the first-order logic $\text{FO}(A^*, \sqsubseteq)$.

Formulae are built with:

- ▶ Variables, x, y, \dots
- ▶ Constant words $\epsilon, abc, bbcac, \dots$
- ▶ Atomic formulae $x \sqsubseteq y, x = aabc, \dots$
- ▶ Boolean operations \neg, \vee, \wedge
- ▶ Quantification $\forall x, \exists x$.

$$\forall x, y (x \sqsubseteq y \vee y \sqsubseteq x)$$

True if and only if $|A| \leq 1$.

Example with constants:

$$\forall x [(ab \sqsubseteq x \wedge ba \sqsubseteq x) \implies (aa \sqsubseteq x \vee bb \sqsubseteq x)]$$

Decidability

Truth problem: Given a finite A , and a sentence ϕ in $\text{FO}(A^*, \sqsubseteq)$, is ϕ true?

Decidability

Truth problem: Given a finite A , and a sentence ϕ in $\text{FO}(A^*, \sqsubseteq)$, is ϕ true?

H. Comon & Treinen (1994) showed that the logic is undecidable (without even considering it!)

D. Kuske (2006) showed that it is undecidable even for the Σ_3 **fragment** (\equiv quantifiers occur only as $\exists^* \forall^* \exists^*$) and even for formulae without constants.

Kuske also showed that the Σ_1 fragment (without constants) is decidable.

Decidability

Truth problem: Given a finite A , and a sentence ϕ in $\text{FO}(A^*, \sqsubseteq)$, is ϕ true?

H. Comon & Treinen (1994) showed that the logic is undecidable (without even considering it!)

D. Kuske (2006) showed that it is undecidable even for the Σ_3 fragment (\equiv quantifiers occur only as $\exists^* \forall^* \exists^*$) and even for formulae without constants.

Kuske also showed that the Σ_1 fragment (without constants) is decidable.

We showed (FST&TCS 2015) that the Σ_2 fragment is undecidable, even for formulae without constants and for a two-letter alphabet

What about bounded variable fragments?

$\text{FO}^2(A^*; \sqsubseteq)$: only two variables are allowed, no other restrictions
(on quantifiers or ...)

What about bounded variable fragments?

$FO^2(A^*; \sqsubseteq)$: only two variables are allowed, no other restrictions (on quantifiers or ...)

The variables can be reused and one can express nontrivial properties:

- ▶ y has length at most 1:

$$\forall x(x \sqsubseteq y \implies y \sqsubseteq x \vee \forall y.x \sqsubseteq y)$$

- ▶ All letters appear in x :

$$\forall y((\forall x(x \sqsubseteq y \implies y \sqsubseteq x \vee \forall y.x \sqsubseteq y)) \implies y \sqsubseteq x)$$

Decidability for FO^2

Theorem (Kuske, with prompting from K. & S.)

Let $\phi(x)$ be an $\text{FO}^2(A^*; \sqsubseteq)$ formula with at most one free variable, x . Then $(L_\phi \stackrel{\text{def}}{=} \{w \in A^* : A^* \models \phi(w)\})$ is effectively regular

Decidability for FO^2

Theorem (Kuske, with prompting from K. & S.)

Let $\phi(x)$ be an $\text{FO}^2(A^*; \sqsubseteq)$ formula with at most one free variable, x . Then $(L_\phi \stackrel{\text{def}}{=} \{w \in A^* : A^* \models \phi(w)\})$ is effectively regular

We allow a more expressive logic by allowing predicates $x \in L$ and $y \in L$ to appear, for any regular L

E.g. formulae like “ $x \in a^*b^* \iff c \sqsubseteq x \wedge ba \sqsubseteq x$ ”

Decidability for FO^2

Theorem (Kuske, with prompting from K. & S.)

Let $\phi(x)$ be an $\text{FO}^2(A^*; \sqsubseteq)$ formula with at most one free variable, x . Then $(L_\phi \stackrel{\text{def}}{=} \{w \in A^* : A^* \models \phi(w)\})$ is effectively regular

We allow a more expressive logic by allowing predicates $x \in L$ and $y \in L$ to appear, for any regular L

E.g. formulae like “ $x \in a^*b^* \iff c \sqsubseteq x \wedge ba \sqsubseteq x$ ”

Proof of Thm: by induction on the structure of the formula.

Base case: ϕ is $x \in L$, or $x = x$, or $x \sqsubseteq x$, etc: trivial.

Decidability for FO^2

Theorem (Kuske, with prompting from K. & S.)

Let $\phi(x)$ be an $\text{FO}^2(A^*; \sqsubseteq)$ formula with at most one free variable, x . Then $(L_\phi \stackrel{\text{def}}{=} \{w \in A^* : A^* \models \phi(w)\})$ is effectively regular

We allow a more expressive logic by allowing predicates $x \in L$ and $y \in L$ to appear, for any regular L

E.g. formulae like “ $x \in a^*b^* \iff c \sqsubseteq x \wedge ba \sqsubseteq x$ ”

Proof of Thm: by induction on the structure of the formula.

Base case: ϕ is $x \in L$, or $x = x$, or $x \sqsubseteq x$, etc: trivial.

Inductive case: If ϕ is $\neg\phi_1$, or $\phi_1 \vee \phi_2$: regular languages are closed under boolean operations.

Decidability for FO^2

Theorem (Kuske, with prompting from K. & S.)

Let $\phi(x)$ be an $FO^2(A^*; \sqsubseteq)$ formula with at most one free variable, x . Then $(L_\phi \stackrel{\text{def}}{=} \{w \in A^* : A^* \models \phi(w)\})$ is effectively regular

We allow a more expressive logic by allowing predicates $x \in L$ and $y \in L$ to appear, for any regular L

E.g. formulae like “ $x \in a^*b^* \iff c \sqsubseteq x \wedge ba \sqsubseteq x$ ”

Proof of Thm: by induction on the structure of the formula.

Base case: ϕ is $x \in L$, or $x = x$, or $x \sqsubseteq x$, etc: trivial.

Inductive case: If ϕ is $\neg\phi_1$, or $\phi_1 \vee \phi_2$: regular languages are closed under boolean operations.

If $\phi(x)$ is $\exists y\psi(x, y)$: This is the interesting case!

Decidability proof for FO^2 (2)

We have $\exists y \psi(x, y)$, and we want to translate it to $x \in L$, for some regular L . We simplify ψ in steps.

Decidability proof for FO^2 (2)

We have $\exists y \psi(x, y)$, and we want to translate it to $x \in L$, for some regular L . We simplify ψ in steps.

1. Replace every strict subformula with only one free variable by “ $x \in L_1$ ” or “ $y \in L_2$ ” (by induction hypothesis).

Decidability proof for FO^2 (2)

We have $\exists y \psi(x, y)$, and we want to translate it to $x \in L$, for some regular L . We simplify ψ in steps.

1. Replace every strict subformula with only one free variable by “ $x \in L_1$ ” or “ $y \in L_2$ ” (by induction hypothesis).
2. ψ is quantifier-free now. Push negations inside formulae and eliminate them using $=$, \sqsubset , \supset , \perp as only relations.

NB: For every $x, y \in A^*$, exactly one of the following holds:

- Equality: $x = y$
- Strict subword: $x \sqsubset y$, defined as $x \sqsubseteq y$ and $x \neq y$
- Strict superword: $x \supset y$, defined as $x \supseteq y$ and $x \neq y$
- Incomparability: $x \perp y$, defined as $x \not\sqsubseteq y$ and $y \not\sqsubseteq x$

Decidability proof for FO^2 (2)

We have $\exists y \psi(x, y)$, and we want to translate it to $x \in L$, for some regular L . We simplify ψ in steps.

1. Replace every strict subformula with only one free variable by “ $x \in L_1$ ” or “ $y \in L_2$ ” (by induction hypothesis).
2. ψ is quantifier-free now. Push negations inside formulae and eliminate them using $=, \sqsubset, \sqsupset, \perp$ as only relations.
3. Put ψ in DNF and replace $\exists y(\psi_1 \vee \psi_2)$ with $\exists y\psi_1 \vee \exists y\psi_2$.

Decidability proof for FO² (2)

We have $\exists y \psi(x, y)$, and we want to translate it to $x \in L$, for some regular L . We simplify ψ in steps.

1. Replace every strict subformula with only one free variable by “ $x \in L_1$ ” or “ $y \in L_2$ ” (by induction hypothesis).
2. ψ is quantifier-free now. Push negations inside formulae and eliminate them using $=, \sqsubset, \supset, \perp$ as only relations.
3. Put ψ in DNF and replace $\exists y(\psi_1 \vee \psi_2)$ with $\exists y\psi_1 \vee \exists y\psi_2$.
4. Now ψ is a conjunction of formulae of the form

$$\begin{aligned} \psi = & \quad x \in L_1 \wedge x \in L_2 \wedge \dots \\ & \wedge y \in L'_1 \wedge y \in L'_2 \wedge \dots \\ & \wedge xR_1y \wedge xR_2y \wedge \dots \end{aligned}$$

where each R_i is $\supset, \sqsubset, =$, or \perp .

Decidability proof for FO² (2)

$$\exists y \psi(x, y) = \exists y \left(\begin{array}{l} x \in L_1 \wedge x \in L_2 \wedge \dots \\ \wedge y \in L'_1 \wedge y \in L'_2 \wedge \dots \\ \wedge xR_1y \wedge xR_2y \wedge \dots \end{array} \right)$$

Decidability proof for FO² (2)

$$\exists y \psi(x, y) = \exists y \left(\begin{array}{l} x \in L_1 \wedge x \in L_2 \wedge \dots \\ \wedge y \in L'_1 \wedge y \in L'_2 \wedge \dots \\ \wedge xR_1y \wedge xR_2y \wedge \dots \end{array} \right)$$

We now rewrite this under the form

$$x \in L \wedge \exists y (y \in L' \wedge xRy)$$

with R among $\{\supset, \sqsubset, =, \perp\}$.

Thus $\exists y \psi(x, y)$ is equivalent to $x \in L \cap R^{-1}(L')$

Decidability proof for FO² (2)

$$\exists y \psi(x, y) = \exists y \left(\begin{array}{l} x \in L_1 \wedge x \in L_2 \wedge \dots \\ \wedge y \in L'_1 \wedge y \in L'_2 \wedge \dots \\ \wedge xR_1y \wedge xR_2y \wedge \dots \end{array} \right)$$

We now rewrite this under the form

$$x \in L \wedge \exists y (y \in L' \wedge xRy)$$

with R among $\{\supseteq, \sqsubseteq, =, \perp\}$.

Thus $\exists y \psi(x, y)$ is equivalent to $x \in L \cap R^{-1}(L')$

This is **regular** since each R is a rational relation (= a transducer)

NB: not trivial in the case of \perp

Complexity for FO^2 ?

- ▶ each quantifier elimination involves Boolean operations on regular languages and computing some $R^{-1}(L)$

Complexity for FO²?

- ▶ each quantifier elimination involves Boolean operations on regular languages and computing some $R^{-1}(L)$
- ▶ computing $\downarrow L = \sqsubseteq^{-1}(L)$ and $\uparrow L = \sqsupseteq^{-1}(L)$ on finite automata is simple but may involve an exponential blowup in size (Niewerth & K & S, TCS 2016)

Complexity for FO^2 ?

- ▶ each quantifier elimination involves Boolean operations on regular languages and computing some $R^{-1}(L)$
- ▶ computing $\downarrow L = \sqsubseteq^{-1}(L)$ and $\uparrow L = \sqsupseteq^{-1}(L)$ on finite automata is simple but may involve an exponential blowup in size (Niewerth & K & S, TCS 2016)
- ▶ leading to a nonelementary complexity upper bound for the quantifier elimination algorithm
- ▶ however we only managed to prove a PSPACE lower bound for the problem

Complexity for FO^2 ?

- ▶ each quantifier elimination involves Boolean operations on regular languages and computing some $R^{-1}(L)$
- ▶ computing $\downarrow L = \sqsubseteq^{-1}(L)$ and $\uparrow L = \sqsupseteq^{-1}(L)$ on finite automata is simple but may involve an exponential blowup in size (Niewerth & K & S, TCS 2016)
- ▶ leading to a nonelementary complexity upper bound for the quantifier elimination algorithm
- ▶ however we only managed to prove a PSPACE lower bound for the problem
- ▶ Eventually we managed to prove a better upper bound using a new measure of complexity: the height of piecewise-testable languages (this paper)

Piecewise testable languages

Are simple regular languages introduced by I. Simon in 1972

Def: a language is **piecewise testable** if it can be defined by **required and excluded subwords** (in any finite boolean combination).

Piecewise testable languages

Are simple regular languages introduced by I. Simon in 1972

Def: a language is **piecewise testable** if it can be defined by **required and excluded subwords** (in any finite boolean combination).

For example, with alphabet $A = \{a, b, c\}$:

$$a^*b^* = \{x \in A^* \mid c \notin x \wedge ba \notin x\}$$

Piecewise testable languages

Are simple regular languages introduced by I. Simon in 1972

Def: a language is **piecewise testable** if it can be defined by **required and excluded subwords** (in any finite boolean combination).

For example, with alphabet $A = \{a, b, c\}$:

$$a^*b^* = \{x \in A^* \mid c \not\sqsubseteq x \wedge ba \not\sqsubseteq x\}$$

$$\begin{aligned} & a \sqsubseteq x \wedge bc \sqsubseteq x \wedge ca \not\sqsubseteq x \\ \iff & x \in (A^*aA^* \cap A^*bA^*cA^*) \setminus A^*cA^*aA^* \end{aligned}$$

NB: was defined by analogy with **locally testable** languages (using required and excluded *factors*)

Applications in linguistics, machine learning, ..

Piecewise testable languages

$$\text{Fin} \subseteq \text{PT} \subseteq \mathcal{B}_1 \subseteq \text{SF} \subseteq \text{Reg}$$

Piecewise testable languages

$\text{Fin} \subseteq \text{PT} \subseteq \mathcal{B}_1 \subseteq \text{SF} \subseteq \text{Reg}$

L is PT

- $\iff L$ is a boolean combination of filters and ideals of $\langle A^*; \sqsubseteq \rangle$
- $\iff L$ is $B(\Sigma_1)$ -definable (NB: here in FO logic *over words*)
- $\iff L$ has a (finite and) \mathcal{J} -trivial syntactic monoid
- \iff the canonical automaton for L is acyclic and locally confluent

Piecewise testable languages

$\text{Fin} \subseteq \text{PT} \subseteq \mathcal{B}_1 \subseteq \text{SF} \subseteq \text{Reg}$

L is PT

- $\iff L$ is a boolean combination of filters and ideals of $\langle A^*; \sqsubseteq \rangle$
- $\iff L$ is $B(\Sigma_1)$ -definable (NB: here in FO logic *over words*)
- $\iff L$ has a (finite and) \mathcal{J} -trivial syntactic monoid
- \iff the canonical automaton for L is acyclic and locally confluent
- $\iff L$ is a union of \sim_n equivalence classes (for some n)

Recall: $x \sim_n y \stackrel{\text{def}}{\iff}$ x and y have the same subwords of length $\leq n$

Prop.

1. \sim_n is an equivalence with finite index.
2. It is a congruence : $x \sim_n y \implies uxv \sim_n uyv$

The hierarchy of PT languages

Def. A PT language L is n -PT if it is a union of \sim_n classes

$\stackrel{\text{def}}{\Leftrightarrow} L$ is definable by excluding/requiring subwords of length $\leq n$

Examples: A^* is 0-PT, $(a + c)^*$ is 1-PT, ab is ?-PT

The hierarchy of PT languages

Def. A PT language L is n -PT if it is a union of \sim_n classes

$\stackrel{\text{def}}{\Leftrightarrow} L$ is definable by excluding/requiring subwords of length $\leq n$

Examples: A^* is 0-PT, $(a + c)^*$ is 1-PT, ab is ?-PT

The smallest n such that L is n -PT is called the **PT-height** (also PT-index) of L . Notation : $h(L)$

The hierarchy $0\text{-PT} \subseteq 1\text{-PT} \subseteq 2\text{-PT} \cdots \subseteq n\text{-PT} \cdots$ is strict

PT-height : a new measure of **descriptive complexity**

PT-height as a measure of descriptive complexity

PT-height is **coarser** and **higher-level** than state-complexity (i.e., size of automata):

- ▶ for $|A| = k$, there are $\approx k^n$ words of length $\leq n$, hence $\approx 2^{2^{k^n}}$ combinations excluding/requiring subwords of length $\leq n$.

PT-height as a measure of descriptive complexity

PT-height is **coarser** and **higher-level** than state-complexity (i.e., size of automata):

- ▶ for $|A| = k$, there are $\approx k^n$ words of length $\leq n$, hence $\approx 2^{2^{k^n}}$ combinations excluding/requiring subwords of length $\leq n$.
- ▶ $h(L) \leq \text{diam}(\text{DFA for } L)$ [Klíma & Polák 2013]

Problem: How to compute $h(L)$? How to bound it?

Computing PT-heights: Basics

$$h(A^* \setminus L) = h(L)$$

$$h(L \cup L') \leq \max(h(L), h(L'))$$

Computing PT-heights: Basics

$$h(A^* \setminus L) = h(L)$$

$$h(L \cup L') \leq \max(h(L), h(L'))$$

$$h(\text{mirror}(L)) = h(L)$$

Computing PT-heights: Basics

$$h(A^* \setminus L) = h(L)$$

$$h(L \cup L') \leq \max(h(L), h(L'))$$

$$h(\text{mirror}(L)) = h(L)$$

Question 1: What about other operations ?

Few yield/preserve PT languages: $\uparrow L$, $\downarrow L$, $\text{prefixes}(L)$, $\text{min}(L)$, ??

Question 2: What about algorithms?

$h(L)$ is clearly computable by brute-force enumeration. But what about efficient algorithms?

Case study: single words

Let $u \in A^*$. What is $h(\{u\})$, simply written $h(u)$?

Example: What is $h(aabb)$?

Case study: single words

Let $u \in A^*$. What is $h(\{u\})$, simply written $h(u)$?

Example: What is $h(aabb)$? Answer: 3

Case study: single words

Let $u \in A^*$. What is $h(\{u\})$, simply written $h(u)$?

Example: What is $h(aabb)$? Answer: 3

Thm. $h(u)$ can be computed in polynomial time

Proof Idea. 1. For given u, v , one can compute the smallest n such that $u \not\sim_n v$ (Hint: the DFA for $\downarrow u$ has $|u| + 2$ states and we look for the shortest word in $\downarrow u \setminus \downarrow v \cup \downarrow v \setminus \downarrow u$)

2. If $[u]_{\sim_n}$ is not a singleton then there is a word v in $u \sqcup A$ such that $u \sim_n v$. \square

NB: we also have a linear time algorithm

Coro. $h(L)$ for finite $L \subseteq A^*$ computable in polynomial time since $h(L) = \max_{u \in L} h(u)$ holds when L finite

Case study: filters

Prop. $h(\uparrow u) = |u|$

Proof. To prove $h(\uparrow u) \geq n = |u|$ we exhibit two words v, v' with $v \sim_{n-1} v'$ and $v \ni u \not\subseteq v'$

Case study: filters

Prop. $h(\uparrow u) = |u|$

Proof. To prove $h(\uparrow u) \geq n = |u|$ we exhibit two words v, v' with $v \sim_{n-1} v'$ and $v \ni u \not\subseteq v'$

NB. $h(\uparrow u \cup \uparrow v)$ can be smaller than $|u|$ and/or $|v|$

E.g. $\uparrow aba \cup \uparrow bab = \uparrow ab \cap \uparrow ba$ hence $h(\uparrow aba \cup \uparrow bab) \leq 2$

Case study: single words again

Let $A_k = \{a_1, \dots, a_k\}$. Pick some length parameter ℓ

Define $U_k \in A_k^*$ with

$$U_k = \begin{cases} \epsilon & \text{for } k = 0 \\ (U_{k-1}a_k)^\ell U_{k-1} & \end{cases}$$

E.g. with $\ell = 3$:

$$U_1 = a_1 a_1 a_1$$

$$U_2 = a_1 a_1 a_1 a_2 a_1 a_1 a_1 a_2 a_1 a_1 a_1 a_2 a_1 a_1 a_1$$

\vdots

Fact. $|U_k| = (\ell + 1)^k - 1$ and $h(U_k) = \ell k + 1$

Main tool for upper bounds (also lower)

Small Subword Thm. Let $k = |A|$. For any $n \in \mathbb{N}$ and $u \in A^*$ there exists $v \sqsubseteq u$ with $v \sim_n u$ and $|v| \leq f_k(n) \stackrel{\text{def}}{=} \left(\frac{n-1}{k} + 2\right)^k$

Main tool for upper bounds (also lower)

Small Subword Thm. Let $k = |A|$. For any $n \in \mathbb{N}$ and $u \in A^*$ there exists $v \sqsubseteq u$ with $v \sim_n u$ and $|v| \leq f_k(n) \stackrel{\text{def}}{=} \left(\frac{n-1}{k} + 2\right)^k$

Coro. $|u| + 1 \geq h(u) > 1 + k(|u|^{1/k} - 2)$ for any $u \in A^*$

Rem. U_k from previous slide had $h(U_k) = 1 + k((|U_k| + 1)^{1/k} - 1)$

Using the Small Subword Theorem

Thm 1. If L is n -PT then $\uparrow L$ (resp., $\min(L)$ and $\uparrow_{<} L$) is n' -PT for $n' \leq f_k(n)$ (resp. $n' \leq f_k(n) + 1$).

Using the Small Subword Theorem

Thm 1. If L is n -PT then $\uparrow L$ (resp., $\min(L)$ and $\uparrow_{<} L$) is n' -PT for $n' \leq f_k(n)$ (resp. $n' \leq f_k(n) + 1$).

Thm 2. If L is n -PT then $\downarrow L$ (resp., $\downarrow_{<} L$) is n' -PT for $n' \leq (k + 1)(f_k(n) + 1)$.

Using the Small Subword Theorem

Thm 1. If L is n -PT then $\uparrow L$ (resp., $\min(L)$ and $\uparrow_{<} L$) is n' -PT for $n' \leq f_k(n)$ (resp. $n' \leq f_k(n) + 1$).

Thm 2. If L is n -PT then $\downarrow L$ (resp., $\downarrow_{<} L$) is n' -PT for $n' \leq (k + 1)(f_k(n) + 1)$.

Thm 3. If L is PT then $\perp^{-1}(L) \stackrel{\text{def}}{=} \{u \mid \exists v \in L : u \perp v\}$ is PT.
NB: $\perp^{-1}(v) = A^* \setminus \uparrow v \setminus \downarrow v$ and $\perp^{-1}(L) = \bigcup_{v \in L} \perp^{-1}(v)$

Furthermore: If L is n -PT then $\perp^{-1}(L)$ is n' -PT for $n' \leq f_k(n) + 1$

Using the Small Subword Theorem

Thm 1. If L is n -PT then $\uparrow L$ (resp., $\min(L)$ and $\uparrow_{<} L$) is n' -PT for $n' \leq f_k(n)$ (resp. $n' \leq f_k(n) + 1$).

Thm 2. If L is n -PT then $\downarrow L$ (resp., $\downarrow_{<} L$) is n' -PT for $n' \leq (k + 1)(f_k(n) + 1)$.

Thm 3. If L is PT then $\perp^{-1}(L) \stackrel{\text{def}}{=} \{u \mid \exists v \in L : u \perp v\}$ is PT.
NB: $\perp^{-1}(v) = A^* \setminus \uparrow v \setminus \downarrow v$ and $\perp^{-1}(L) = \bigcup_{v \in L} \perp^{-1}(v)$

Furthermore: If L is n -PT then $\perp^{-1}(L)$ is n' -PT for $n' \leq f_k(n) + 1$

Bottom line. polynomial $O(n^k)$ upper bound on PT-height when considering $R^{-1}(L)$ for $R = \sqsubset, \sqsupset, \perp$ and a few more operations

$FO^2(A^*, \sqsubseteq)$ and piecewise testability

Thm. Let $\phi(x)$ be an $FO^2(A^*; \sqsubseteq)$ formula with at most one free variable, x . Then L_ϕ is effectively regular and is PT

Proof. Recall “ $\exists y \psi(x, y)$ is equivalent to $x \in L \cap R^{-1}(L')$ ”
Add that $R^{-1}(L')$ is PT when L' is.

$FO^2(A^*, \sqsubseteq)$ and piecewise testability

Thm. Let $\phi(x)$ be an $FO^2(A^*; \sqsubseteq)$ formula with at most one free variable, x . Then L_ϕ is effectively regular and is PT

Proof. Recall “ $\exists y \psi(x, y)$ is equivalent to $x \in L \cap R^{-1}(L')$ ”
Add that $R^{-1}(L')$ is PT when L' is.

Complexity. L_ϕ is n -PT for n in $2^{2^{O(|\phi|)}}$

Coro. Constructing a canonical DFA for L_ϕ can be done in 3-EXPTIME

Conclusions

PT-height a useful **measure of descriptive complexity**

Computing and reasoning about PT-heights is difficult

- We recently managed to prove $h(uv) < h(u) + h(v)$
- For a long time we conjectured $\forall w \in u \sqcup v : h(w) < h(u) + h(v)$
but this fails (took weeks of computer time to find counter example)

Conclusions

PT-height a useful **measure of descriptive complexity**

Computing and reasoning about PT-heights is difficult

- We recently managed to prove $h(uv) < h(u) + h(v)$
- For a long time we conjectured $\forall w \in u \sqcup v : h(w) < h(u) + h(v)$
but this fails (took weeks of computer time to find counter example)

Regarding $\text{FO}^2(\sqsubseteq)$:

- ▶ we still have a gap to close between PSPACE and 3-EXPTIME
- ▶ what about $\text{FO}^2(\sqsubseteq)$ with regular predicates “ $x \in L$ ” ?
- ▶ what about $\text{FO}^2(\sqsubseteq, \sqsubseteq_1)$? Counting extensions ?
- ▶ what about *logics of embedding relation* in other classes of structures?