Bio-Inspired Operations on Formal Languages

Da-Jung Cho

Department of Computer Science
Yonsei University

Preliminary Defense of Ph.D Thesis

Outline

Outline

In Molecular Biology

Deoxyribonucleic acid (DNA) and ribonucleic acid (RNA)

- are sequences over $\{A, G, C, T\ (U)\}$,
- has hydrogen bonds with the strongest complementary pairs A-T(U) and G-C.

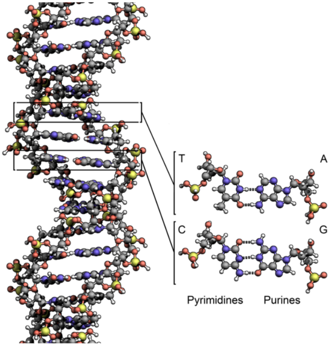


Figure: An example of double-stranded DNA.

DNA Rearrangements

DNA undergoes abnormal rearrangement such as insertion, deletion, inversion, and duplication.

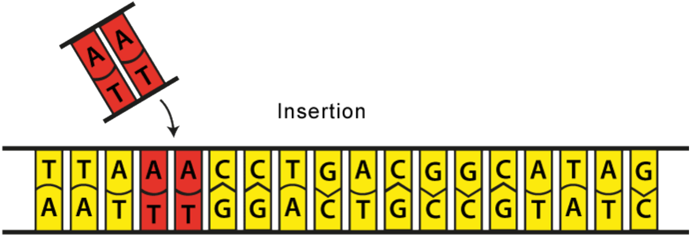


Figure: An example of insertion on a DNA sequence.

DNA Rearrangements

DNA undergoes abnormal rearrangement such as insertion, deletion, inversion, and duplication.

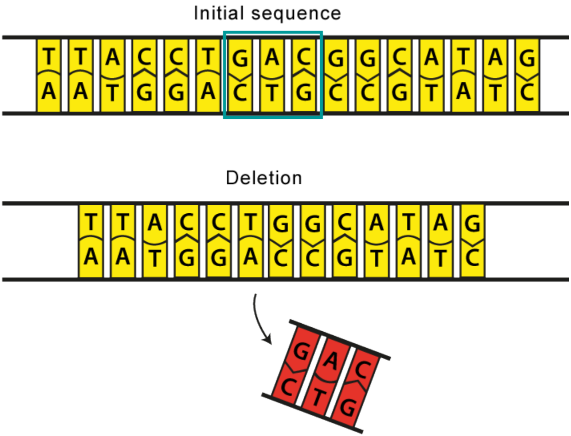


Figure: An example of deletion on a DNA sequence.

DNA Rearrangements

DNA undergoes abnormal rearrangement such as insertion, deletion, inversion, and duplication.

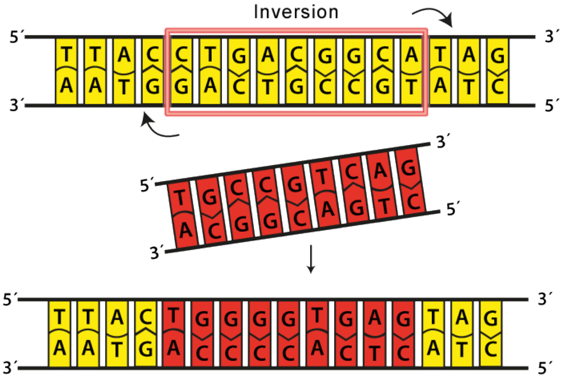
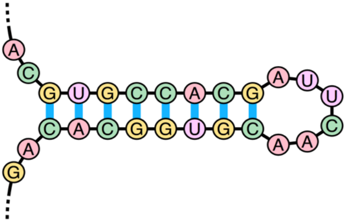


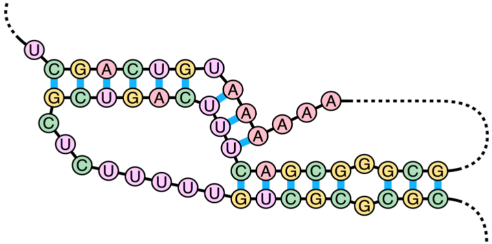
Figure: An example of inversion on a DNA sequence.

Secondary Structure

- An RNA is generated from *DNA transcription*
- The complementary paring *A*-*U* and *G*-*C* leads an RNA to form secondary structures



(a) Hairpin structure



(b) Pseudoknot structure

Figure: An example of RNA secondary structures

Relation between transformation and disease

Abnormal transformations on DNA (or RNA) are closely related to

- several diseases (can be inherited)
- species diversity

Figure: The Cri du chat (Cat-cry) syndrome caused by deletion mutation.

Relation between transformation and disease

Abnormal transformations on DNA (or RNA) are closely related to

- several diseases (can be inherited)
- species diversity

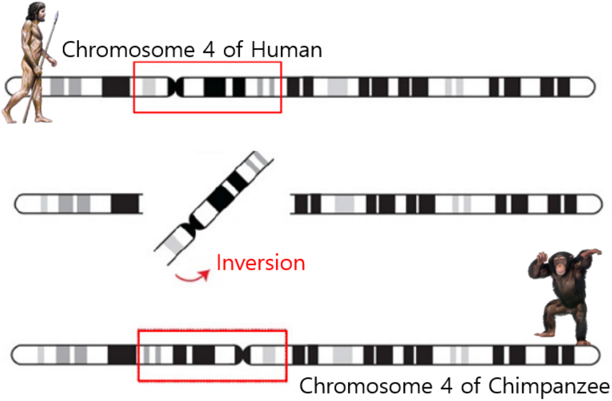
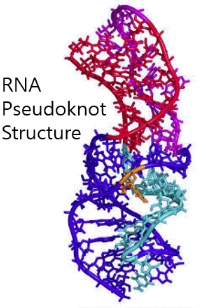


Figure: An inversion occurs on chromosome 4 between human and chimpanzee.

Relation between transformation and disease

Abnormal transformations on DNA (or RNA) are closely related to

- several diseases (can be inherited)
- species diversity



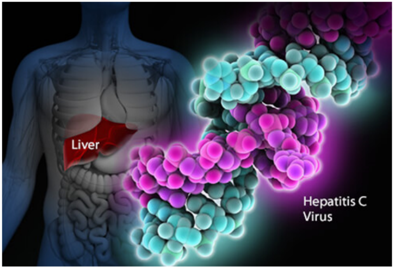


Figure: Pseudoknot structure is related to hepatitis C virus (HCV).

Relation between transformation and disease

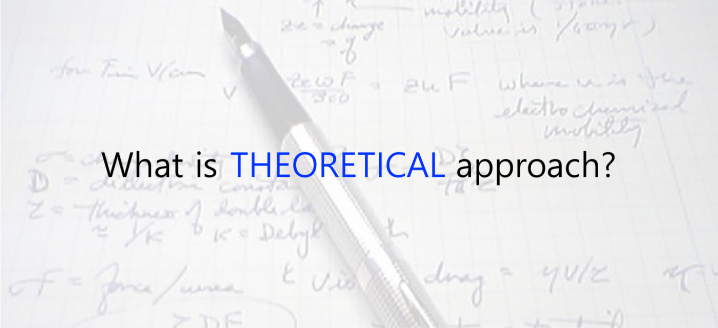
For analyzing and investigating DNA transformations

- predict abnormal transformations (insertion, deletion, inversion, duplication...)
- deliberately introduce a transformation to normal sequence

Relation between transformation and disease

For analyzing and investigating DNA transformations

- predict abnormal transformations (insertion, deletion, inversion, duplication...)
- deliberately introduce a transformation to normal sequence



Outline

In Formal Language Theory

Researchers in formal language theory characterize the biological phenomena into OPERATIONS on strings.

- Searls, "The computational linguistics of biological sequences", *Artificial Intelligence and Molecular Biology*, 1993
- Kari and Thierrin, "Contextual insertions/deletions and computability", *Information and Computation*, 1996
- Dassow et al., "Context-free evolutionary grammars and the structural language of nucleic acids", *Biosystems*, 1997
- Dassow et al., "Operations and language generating devices suggested by the genome evolution", *Theoretical Computer Science*, 2002
- Leupold et al., "Formal languages arising from gene repeated duplication", *Theoretical Computer Science*, 2004
- Enaganti et al., "A formal language model of DNA polymerase enzymatic activity", *Fundamenta Informaticae*, 2015

In Formal Language Theory

Researchers in formal language theory characterize the biological phenomena into OPERATIONS on strings.

Why is theoretical research needed?

- Genetic testing can take up to several months to receive the results
- The cost of genetic testing can be over \$2,000[1]
- Errors in genetic testing occur regularly[2]

[1] from U.S. national library of medicine, https://ghr.nlm.nih.gov

[2] Error rates in forensic DNA analysis: definition, numbers, impact and communication, 2014

In Formal Language Theory

Researchers in formal language theory characterize the biological phenomena into OPERATIONS on strings.

Why this research is needed? Theory can

- ~~Genetic testing can take up to several months to receive the results~~ Reduce testing time by an efficient algorithm
- ~~The cost of genetic testing can be over $2,000~~, Reduce testing cost
- ~~Errors in genetic testing occur regularly~~. Reduce errors by allowing us to test more frequently and efficiently

In Formal Language Theory

Researchers in formal language theory characterize the biological phenomena into OPERATIONS on strings.

Why this research is needed? Theory can

- ~~Genetic testing can take up to several months to receive the results~~ Reduce testing time by an efficient algorithm
- ~~The cost of genetic testing can be over $2,000~~, Reduce testing cost
- ~~Errors in genetic testing occur regularly~~. Reduce errors by allowing us to test more frequently and efficiently

How does formal language theory work?

Outline

1. Motivation
   - Background from Molecular Biology
   - Related Works on Bio-Inspired Operations
   - Problems from a Formal Language Viewpoint
2. Main Results
   - Definition of Bio-Inspired Operations
   - Closure Properties of Bio-Inspired Operations
   - Membership Problem for Bio-Inspired Operations
   - Freeness of Bio-Inspired Operation
3. Conclusions
   - Summary
   - Future Works

In Formal Language Theory

Formal language theory can be applied to these problems in practice by

- Modeling biological phenomena as an operation $\clubsuit$
- Solving several theoretical problems
  - Closure: Decide whether or not languages in the Chomsky hierarchy are closed under the operation $\clubsuit$

    $$\text{Given a language } L, \text{ is } \clubsuit(L) \text{ REGULAR?}$$

  - Membership problem: Decide whether or not a given string x belongs to the language $\clubsuit(L)$
  - Freeness: Decide whether or not a given language L contains any string $x \in \clubsuit(L)$

In Formal Language Theory

Formal language theory can be applied to these problems in practice by

- Modeling biological phenomena as an operation $\clubsuit$
- Solving several theoretical problems
  - Closure: Decide whether or not languages in the Chomsky hierarchy are closed under the operation $\clubsuit$
  - Membership problem: Decide whether or not a given string x belongs to the language $\clubsuit(L)$

    $$\text{Given } x \text{ and } L, \text{ is } x \in \clubsuit(L)?$$

  - Freeness: Decide whether or not a given language L contains any string $x \in \clubsuit(L)$

In Formal Language Theory

Formal language theory can be applied to these problems in practice by

- Modeling biological phenomena as an operation $\clubsuit$
- Solving several theoretical problems
  - Closure: Decide whether or not languages in the Chomsky hierarchy are closed under the operation $\clubsuit$
  - Membership problem: Decide whether or not a given string x belongs to the language $\clubsuit(L)$
  - Freeness: Decide whether or not a given language L contains any string $x \in \clubsuit(L)$

$$\text{Given } L, L \text{ is } \clubsuit\text{-free if } L \cap \clubsuit(L) = \emptyset$$

Our Goal

The objective of this thesis is to give a theoretical foundation for DNA computing by

- Characterizing realistic biological phenomena
  - The pseudo-inversion operation $\mathbb{PI}$
  - The pseudo-duplication operation $\mathbb{PD}_k$
  - The pseudoknot-generating operation $\mathbb{PK}_{\mathbb{R}}$
  - The site-directed insertion/deletion operations $\mathbb{SDI}, \mathbb{SDD}$
- Solving problems that might be applied to DNA computing in practice

Outline

Bio-Inspired Operation: Pseudo-Inversion

Definition

For a string $w = uxv \in \Sigma^*$, we define pseudo-inversion of w to be

$$\mathbb{PI}(w) = \{v^R x u^R \mid u, x, v \in \Sigma^*, uv \neq \lambda\}.$$

- for a string $w = w_1 w_2 \cdots w_n$, $w^R = w_n w_{n-1} \cdots w_1$
- λ denotes the empty string, and $\mathbb{PI}(\lambda) = \emptyset$
- extend $\mathbb{PI}$ to languages

$$\mathbb{PI}(L) = \bigcup_{w \in L} \mathbb{PI}(w)$$

- define iterated pseudo-inversion $\mathbb{PI}^*(L)$ as

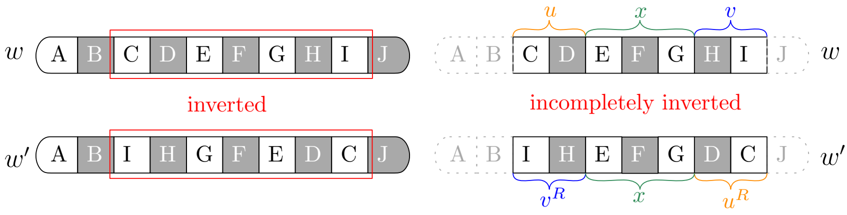
$$\mathbb{PI}^*(L) = \bigcup_{w \in L} \mathbb{PI}^*(w)$$

Bio-Inspired Operation: Pseudo-Inversion

Definition

For a string $w = uxv \in \Sigma^*$, we define pseudo-inversion of w to be

$$\mathbb{PI}(w) = \{v^R x u^R \mid u, x, v \in \Sigma^*, uv \neq \lambda\}.$$



(a) Inversion (b) Pseudo-inversion

Bio-Inspired Operation: Pseudo-Duplication

Definition

For a string $w = uxv \in \Sigma^*$, we define k-pseudo-duplication $\mathbb{PD}_k$ of w to be

$$\mathbb{PD}_k(w) = \{uxx'v \mid u, x, v \in \Sigma^* \text{ and } d(x, x') \leq k\}.$$

- $d(x, x')$ denotes the smallest number of operations that transform x to x', the *edit-distance* between x and x'

$$d(city, kitty) = 2$$

- $\mathbb{PD}_k(L) = \bigcup_{w \in L} \mathbb{PD}_k(w)$ and $\mathbb{PD}_k^*(L) = \bigcup_{w \in L} \mathbb{PD}_k^*(w)$

Bio-Inspired Operation: Pseudo-Duplication

Definition

For a string $w = uxv \in \Sigma^*$, we define *k-pseudo-duplication* $\mathbb{PD}_k$ of w to be

$$\mathbb{PD}_k(w) = \{uxx'v \mid u, x, v \in \Sigma^* \text{ and } d(x, x') \leq k\}.$$

- $d(x, x')$ denotes the smallest number of operations that transform x to x', the *edit-distance* between x and x'

$$cit\_y \to kitty$$

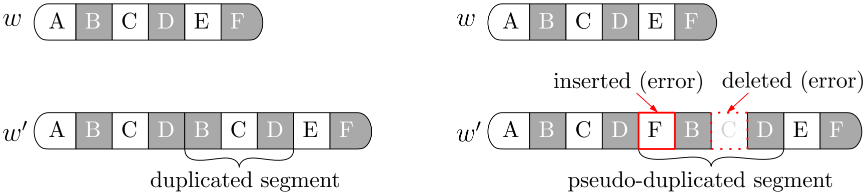
- $\mathbb{PD}_k(L) = \bigcup_{w \in L} \mathbb{PD}_k(w)$ and $\mathbb{PD}_k^*(L) = \bigcup_{w \in L} \mathbb{PD}_k^*(w)$

Bio-Inspired Operation: Pseudo-Duplication

Definition

For a string $w = uxv \in \Sigma^*$, we define *k-pseudo-duplicaiton* $\mathbb{PD}_k$ of w to be

$$\mathbb{PD}_k(w) = \{uxx'v \mid u, x, v \in \Sigma^* \text{ and } d(x, x') \leq k\}.$$



(a) Ordinary duplication

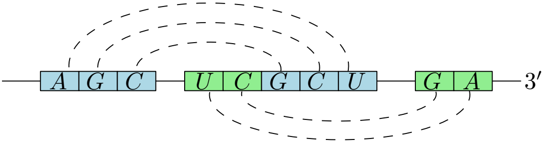
(b) Pseudo-duplication

Bio-Inspired Operation: Pseudoknot-Generating

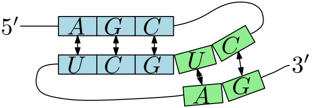
Definition

For a string $w = w_1 w_2 w_3$, we define the pseudoknot-generating operation of w to be

$$\mathbb{PK}_{\mathbb{R}}(w) = \{w_1 w_2 w_3 w_1^R w_4 w_3^R \mid w_1, w_2, w_3, w_4 \in \Sigma^+\}$$



(a) Sequence for pseudoknot



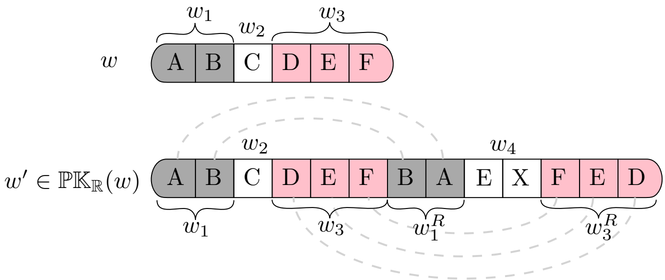
(b) Pseudoknot structure

Bio-Inspired Operation: Pseudoknot-Generating

Definition

For a string $w = w_1 w_2 w_3$, we define the pseudoknot-generating operation of w to be

$$\mathbb{PK}_{\mathbb{R}}(w) = \{w_1 w_2 w_3 w_1^R w_4 w_3^R \mid w_1, w_2, w_3, w_4 \in \Sigma^+\}$$



Bio-Inspired Operation: Pseudoknot-Generating

Definition

For a string $w = w_1 w_2 w_3$, we define the pseudoknot-generating operation of w to be

$$\mathbb{PK}_{\mathbb{R}}(w) = \{w_1 w_2 w_3 w_1^R w_4 w_3^R \mid w_1, w_2, w_3, w_4 \in \Sigma^+\}$$

- extend the pseudoknot-generating operation to languages

$$\mathbb{PK}_{\mathbb{R}}(L) = \bigcup_{w \in L} \mathbb{PK}_{\mathbb{R}}(w)$$

- define iterated $\mathbb{PK}_{\mathbb{R}}$ of w to be

$$\mathbb{PK}_{\mathbb{R}}^*(L) = \bigcup_{w \in L} \mathbb{PK}_{\mathbb{R}}^*(w)$$

Bio-Inspired Operation: Site-Directed Insertion

Definition

Given two strings $x = x_1 uv x_2$ and $y = uwv$, the site-directed insertion of y into x is defined to be

$$x \overset{sdi}{\leftarrow} y = \{x_1 uwvx_2 \mid u \neq \lambda \text{ and } v \neq \lambda\}.$$

- for a string $y = uwv$, we say (u, v) is an outfix of y
- an outfix (u, v) of y is an insertion guide of x if $x \overset{sdi}{\leftarrow} y \neq \emptyset$.
- extend site-directed insertion to languages

$$L_1 \overset{sdd}{\leftarrow} L_2 = \bigcup_{w_i \in L_i, i=1,2} w_1 \overset{sdi}{\leftarrow} w_2.$$

- site-directed insertion of L is inductively defined as $\mathbb{SDI}^0(L) = L$,

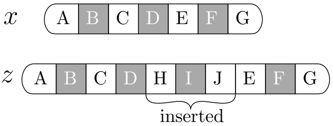
$$\text{and } \mathbb{SDI}^{i+1}(L) = \mathbb{SDI}^i(L) \overset{sdi}{\leftarrow} \mathbb{SDI}^i(L).$$

Bio-Inspired Operation: Site-Directed Insertion

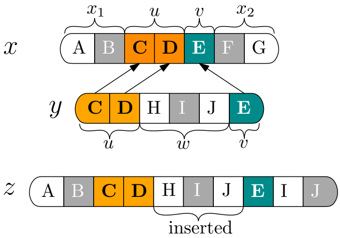
Definition

Given two strings $x = x_1 uv x_2$ and $y = uwv$, the site-directed insertion of y into x is defined to be

$$x \overset{sdi}{\leftarrow} y = \{x_1 uwv x_2 \mid u \neq \lambda \text{ and } v \neq \lambda\}.$$



(a) Ordinary insertion on x



(b) Site-directed insertion of x and y

Bio-Inspired Operation: Site-Directed Deletion

Definition

Given two strings $x = x_1 uwv x_2$ and $y = uv$, the site-directed deletion from x by y is defined to be

$$x \overset{sdd}{\leftarrow} y = \{x_1 uv x_2 \mid u \neq \lambda \text{ and } v \neq \lambda\}.$$

- $y = uv$ is a deletion guide of x if $x \overset{sdd}{\leftarrow} y \neq \emptyset$.
- extend site-directed deletion to languages

$$L_1 \overset{sdd}{\leftarrow} L_2 = \bigcup_{w_i \in L_i, i=1,2} w_1 \overset{sdd}{\leftarrow} w_2.$$

- site-directed deletion of L is inductively defined as $\mathbb{SDD}^0(L) = L$,

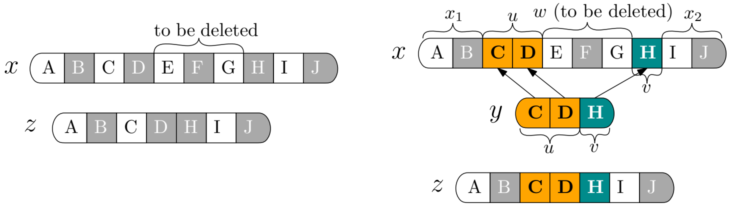
$$\text{and } \mathbb{SDD}^{i+1}(L) = \mathbb{SDD}^i(L) \overset{sdd}{\leftarrow} \mathbb{SDD}^i(L).$$

Bio-Inspired Operation: Site-Directed Deletion

Definition

Given two strings $x = x_1 uwv x_2$ and $y = uv$, the site-directed deletion from x by y is defined to be

$$x \overset{sdd}{\leftarrow} y = \{x_1 uvx_2 \mid u \neq \lambda \text{ and } v \neq \lambda\}.$$



(a) Ordinary deletion on x (b) Site-directed deletion of x and y

Summary

- Pseudo-inversion $\mathbb{PI}$ of $w = uxv$:

$$\mathbb{PI}(w) = \{v^R x u^R \mid u, x, v \in \Sigma^* \text{ and } uv \neq \lambda\}$$

- Pseudo-duplication $\mathbb{PD}_k$ of $w = uxv$:

$$\mathbb{PD}_k(w) = \{uxx'v \mid u, x, v \in \Sigma^* \text{ and } d(x, x') \leq k\}$$

- Pseudoknot-generating $\mathbb{PK}_\mathbb{R}$ of $w = w_1 w_2 w_3$:

$$\mathbb{PK}_\mathbb{R}(w) = \{w_1 w_2 w_3 w_1^R w_4 w_3^R \mid w_1, w_2, w_3, w_4 \in \Sigma^+\}$$

- Site-directed insertion of y into x:

$$x \overset{sdi}{\leftarrow} y = \{x_1 uwv x_2 \mid x = x_1 uv x_2, y = uwv, u \neq \lambda \text{ and } v \neq \lambda\}$$

- Site-directed deletion from x by y:

$$x \overset{sdd}{\leftarrow} y = \{x_1 uv x_2 \mid x = x_1 uwv x_2, y = uv, u \neq \lambda \text{ and } v \neq \lambda\}$$

Outline

Closure Properties of Pseudo-Inversion

Definition

$\mathbb{PI}(w) = \{v^R x u^R \mid w = uxv, u, x, v \in \Sigma^*$ and $uv \neq \lambda\}$

Theorem

Regular languages are closed under the pseudo-inversion operation.

Closure Properties of Pseudo-Inversion

Theorem

Regular languages are closed under the pseudo-inversion operation.

Given an NFA $A = (Q, \Sigma, \delta, q_0, F)$ recognizing a language L, we construct a λ-NFA $B = (P, \Sigma, \gamma, p_0, F_B)$ recognizing $\mathbb{PI}(L)$.

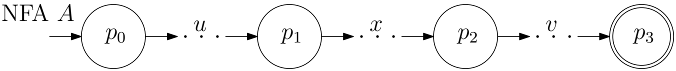
- Q is a finite set of states
- Σ is the alphabet
- $\delta : Q \times (\Sigma \cup \lambda) \to 2^Q$
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states

Closure Properties of Pseudo-Inversion

Theorem

Regular languages are closed under the pseudo-inversion operation.

Given an NFA $A = (Q, \Sigma, \delta, q_0, F)$ recognizing a language L, we construct a λ-NFA $B = (P, \Sigma, \gamma, p_0, F_B)$ recognizing $\mathbb{PI}(L)$.



Closure Properties of Pseudo-Inversion

Theorem

Regular languages are closed under the pseudo-inversion operation.

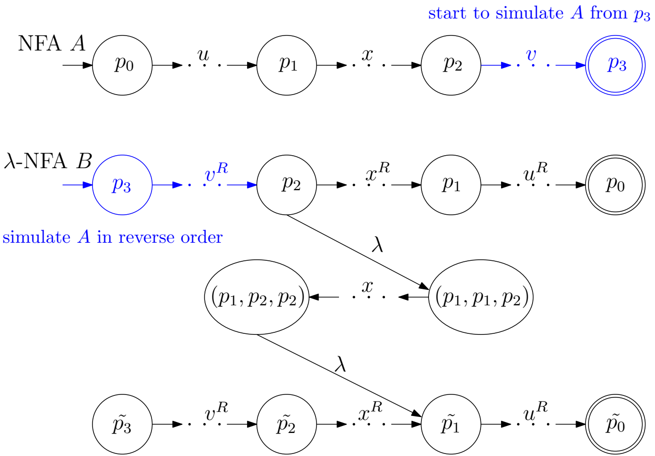
λ-NFA $B = (P, \Sigma, \gamma, p_0, F_B)$ recognizing $\mathbb{PI}(L)$

- $P = Q \cup \tilde{Q} \cup (Q \times Q \times Q)$
- $F_B = \{q_0, \tilde{q}_0\}$
- γ is defined as follows:
  1. for all $p, q \in Q, a \in \Sigma$ and $p \in \delta(q, a)$, $q \in \gamma(p, a)$ and $\tilde{q} \in \gamma(\tilde{p}, a)$
  2. for all $p, q \in Q$, $(q, q, p) \in \gamma(p, \lambda)$, where $p \neq q_f$ or $q \notin F_B$
  3. for all $q, p, r_1, r_2 \in Q, a \in \Sigma$ and $r_2 \in \delta(r_1, a)$, $(q, r_2, p) \in \gamma((q, r_1, p), a)$
  4. for all $p, q \in Q$, $\tilde{q} \in \gamma((q, p, p), \lambda)$

Closure Properties of Pseudo-Inversion

Theorem

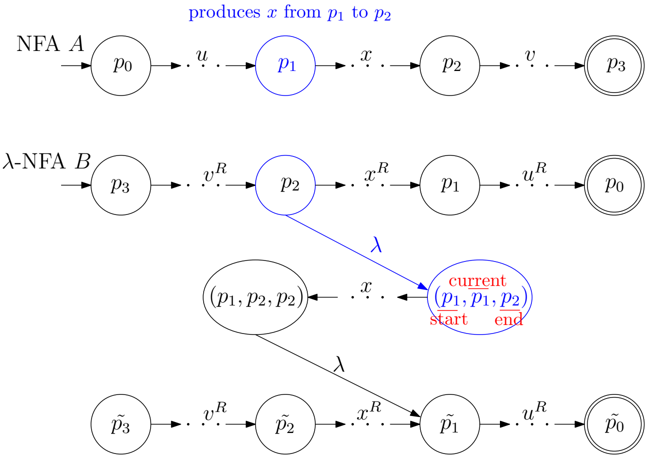
Regular languages are closed under the pseudo-inversion operation.



Closure Properties of Pseudo-Inversion

Theorem

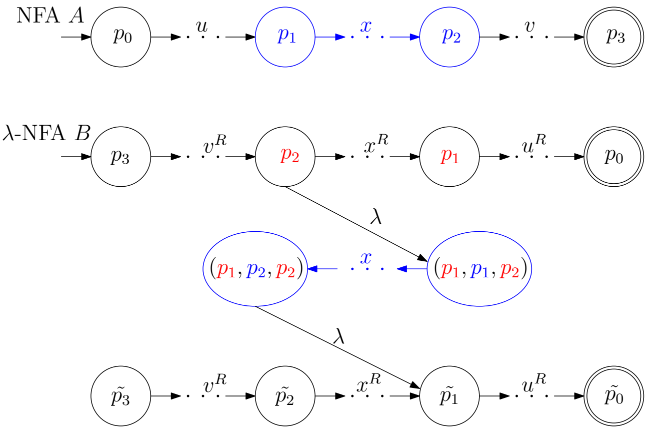
Regular languages are closed under the pseudo-inversion operation.



Closure Properties of Pseudo-Inversion

Theorem

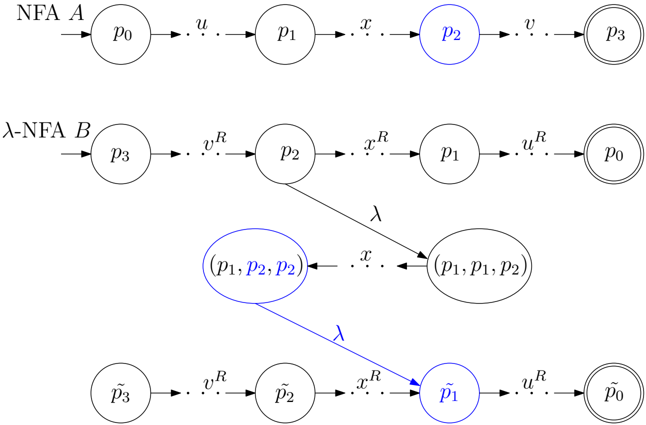
Regular languages are closed under the pseudo-inversion operation.



Closure Properties of Pseudo-Inversion

Theorem

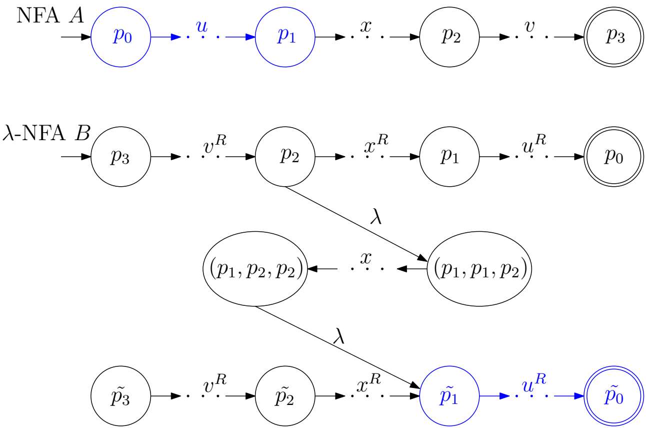
Regular languages are closed under the pseudo-inversion operation.



Closure Properties of Pseudo-Inversion

Theorem

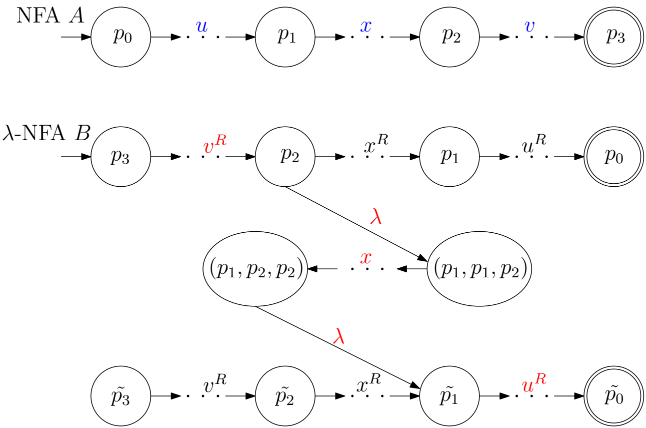
Regular languages are closed under the pseudo-inversion operation.



Closure Properties of Pseudo-Inversion

Theorem

Regular languages are closed under the pseudo-inversion operation.



Closure Properties of Pseudo-Duplication

Definition

$\mathbb{PD}_k(w) = \{uxx'v \mid w = uxv, u, x, v \in \Sigma^* \text{ and } d(x, x') \leq k\}$

Theorem

Regular languages are not closed under the k-pseudo-duplication operation.

Closure Properties of Pseudo-Duplication

Theorem

Regular languages are not closed under the k-pseudo-duplication operation.

Let $L = L(a^*)$ over $\Sigma = \{a, b\}$. We show that

$$L(a^*b^*) \cap \mathbb{PD}_k(L) = \{a^ib^j \mid i \geq j\} \text{ is not regular.}$$

- $\{a^ib^j \mid i \geq j\}$ is not regular by the *pumping lemma* for regular languages
- Since $\underbrace{L(a^*b^*)}_{regular} \cap \mathbb{PD}_k(L) = \underbrace{\{a^ib^j \mid i \geq j\}}_{not\ regular}$,

$$\mathbb{PD}_k(L) \text{ is not regular!}$$

Summary of Closure Properties

- For $\mathbb{PI}$, Closed (regular), Not closed (context-free)
- For $\mathbb{PD}_k$, Not closed (regular, context-free), Closed (context-sensitive)
- For $\mathbb{PK}_{\mathbb{R}}$, Not closed (regular, context-free)
- For $L_1 \overset{sdi}{\leftarrow} L_2$, Closed (regular), Not closed (context-free)
- For $L_1 \overset{sdd}{\leftarrow} L_2$, Closed (regular), Not closed (context-free)

Outline

Membership Problem for Site-Directed Deletion

Definition

Site-directed deletion from x by y:

$$x \overset{sdd}{\leftarrow} y = \{x_1 uv x_2 \mid x = x_1 {\color{blue}u}{\color{red}w}{\color{blue}v} x_2, y = {\color{blue}uv}, u \neq \lambda \text{ and } v \neq \lambda\}$$

Problem

Given three strings x, y, and z, where $|x| \geq |z| \geq |y| \geq 2$, can we determine whether or not

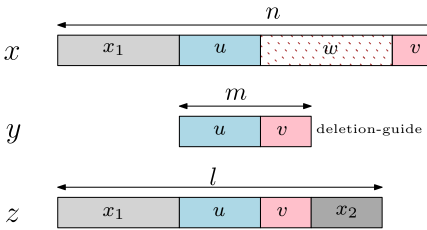
$$z \in x \overset{sdd}{\leftarrow} y?$$

Membership Problem of Site-Directed Deletion

Theorem

Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.

Suppose that there exist x, y, and z such that $z \in x \overset{sdd}{\leftarrow} y$.

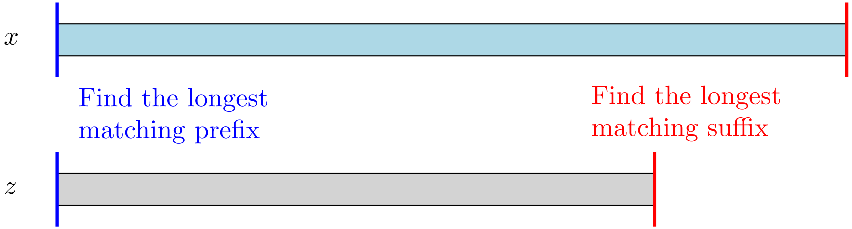


Membership Problem of Site-Directed Deletion

Theorem

Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.

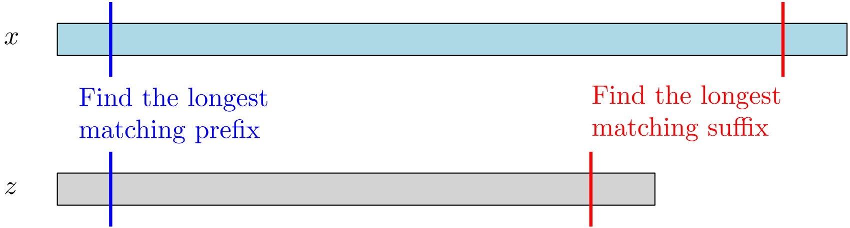
Scan both ends of x and z until a mismatch occurs.



Membership Problem of Site-Directed Deletion

Theorem

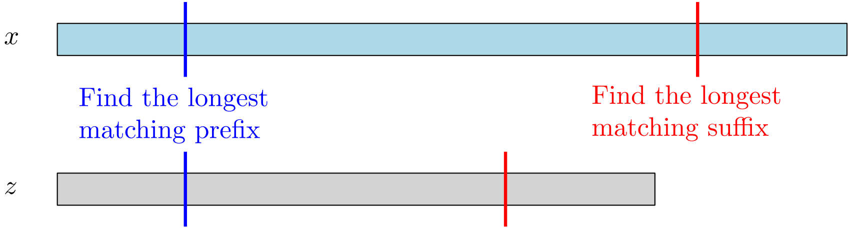
Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.



Membership Problem of Site-Directed Deletion

Theorem

Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.

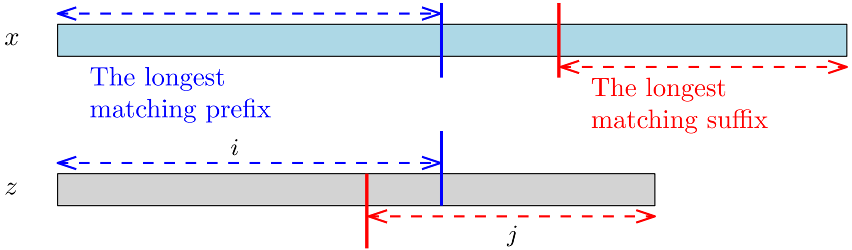


Membership Problem of Site-Directed Deletion

Theorem

Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.

$x[1:i] = z[1:i]$ and $x[n-j+1:n] = z[l-j+1:l]$



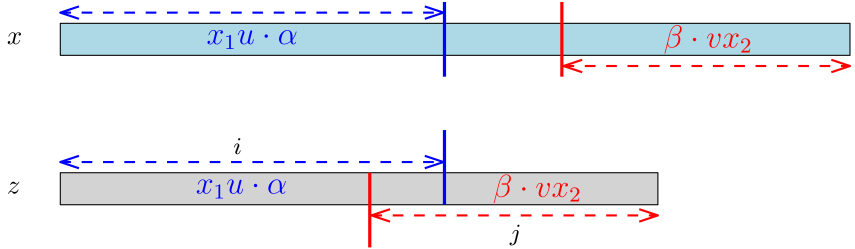
Membership Problem of Site-Directed Deletion

Theorem

Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.

If $z[1 : i]$ and $z[l{-}j{+}1]$ do not overlap ($\alpha, \beta = \lambda$),

$$z[1 : i] \cdot z[l{-}j{+}1 : l] = x_1 uvx_2.$$

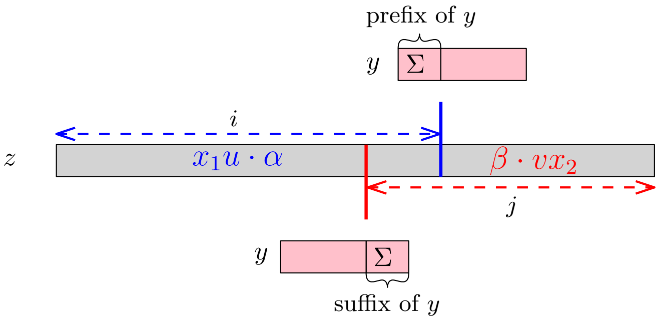


Membership Problem of Site-Directed Deletion

Theorem

Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.

We check whether or not $y = uv$ is a substring of z.

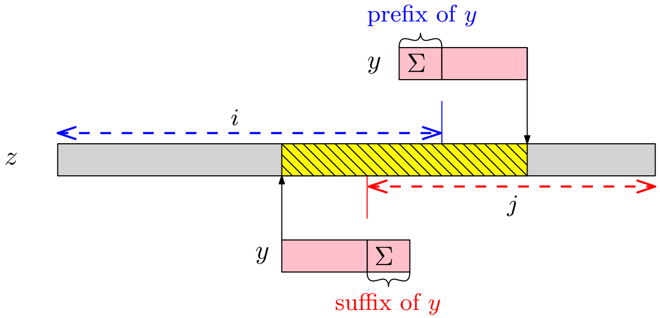


Membership Problem of Site-Directed Deletion

Theorem

Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.

A prefix of y should be a suffix of the longest matching prefix of z,
A suffix of y should be a prefix of the longest matching suffix of z.

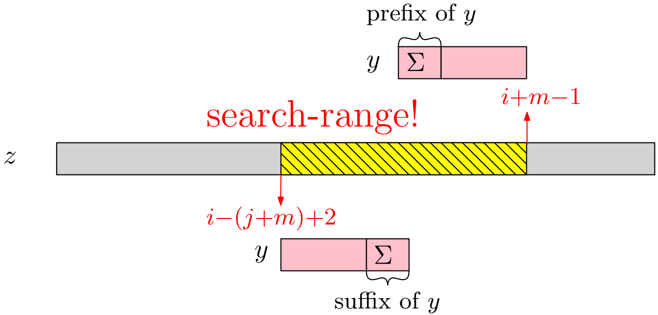


Membership Problem of Site-Directed Deletion

Theorem

Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.

We check for an occurrence of y within $z[l-(j+m)+2 : i+m-1]$.

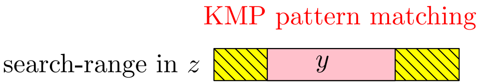


Membership Problem of Site-Directed Deletion

Theorem

Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.

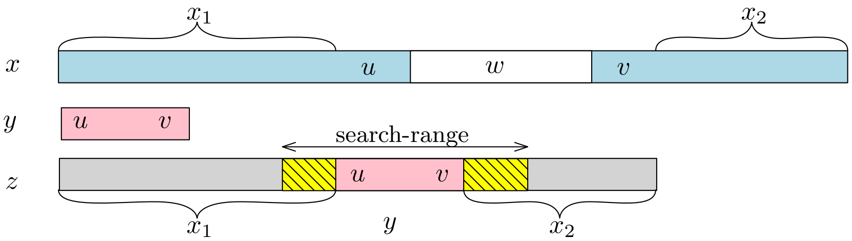
KMP algorithm returns 1 if y occurs in the search-range.



Membership Problem of Site-Directed Deletion

Theorem

Given three strings x, y, and z, we can determine whether or not $z \in x \overset{sdd}{\leftarrow} y$ in $O(n)$ time, where $n = |x|$ and $|x| \geq |z| \geq |y| \geq 2$.



Summary

- Given two strings u and v of length n, we can determine whether or not $v \in \mathbb{PI}(u)$ (and, $v \in \mathbb{PI}^*(u)$) in $O(n)$ time

- Given a string w and an FA A,

$$w \in \mathbb{PK}_{\mathbb{R}}(L(A)) \text{ iff } I_{pk}(w) \cap I_p(w, A) \neq \emptyset$$

- Given two strings x and y, we can determine whether or not $x \overset{sdi}{\leftarrow} y \neq \emptyset$ in $O(n+m)$ time, where $|x| = n$ and $|y| = m$

- Given two strings x and y, we can determine whether or not $x \overset{sdd}{\leftarrow} y \neq \emptyset$ in $O(n)$ time, where $|x| = n$, $|y| = m$ and $m \leq n$

Outline

Freeness of Pseudoknot-Generating

Definition

$\mathbb{PK}_{\mathbb{R}}(w) = \{w_1 w_2 w_3 w_1^R w_4 w_3^R \mid w = w_1 w_2 w_3 \text{ and } w_1, w_2, w_3, w_4 \in \Sigma^+\}$

Theorem

For a given context-free language L, it is undecidable to determine whether or not L is $\mathbb{PK}_{\mathbb{R}}$-free.

A language L is $\mathbb{PK}_{\mathbb{R}}$-free if $L \cap \mathbb{PK}_{\mathbb{R}}(L) = \emptyset$.

Example

The language $L = \{computer, computeroccret\}$ is not $\mathbb{PK}_{\mathbb{R}}$-free since

$$computeroccret \in \mathbb{PK}_{\mathbb{R}}(computer).$$

Freeness of Pseudoknot-Generating

Theorem

For a given context-free language L, it is undecidable to determine whether or not L is $\mathbb{PK}_{\mathbb{R}}$-free.

We use a reduction from the *Post Correspondence Problem* (PCP)

- Let $((u_1, u_2, \ldots, u_n), (v_1, v_2, \ldots, v_n))$ be an instance of PCP, where $u_i, v_i \in \Sigma^*$ and $1 \leq i \leq n$
- A solution of the PCP instance is $i_1, \ldots, i_k \in \{1, \ldots, n\}$ such that

$$u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k}$$

Freeness of Pseudoknot-Generating

Theorem

For a given context-free language L, it is undecidable to determine whether or not L is $\mathbb{PK}_{\mathbb{R}}$-free.

We use a reduction from the *Post Correspondence Problem* (PCP)

Example

Let $I_{PCP} = ((\underbrace{ab}_{u_1}, \underbrace{bbb}_{u_2}, \underbrace{a}_{u_3})(\underbrace{ab}_{v_1}, \underbrace{b}_{v_2}, \underbrace{bba}_{v_3}))$.

The solution is $2, 3, 1$ since

$$u_2 u_3 u_1 = v_2 v_3 v_1 = bbbaab.$$

Freeness of Pseudoknot-Generating

Theorem

For a given context-free language L, it is undecidable to determine whether or not L is $\mathbb{PK}_{\mathbb{R}}$-free.

We use a reduction from the *Post Correspondence Problem* (PCP)

- Let $((u_1, u_2, \ldots, u_n), (v_1, v_2, \ldots, v_n))$ be an instance of PCP, where $u_i, v_i \in \Sigma^*$ and $1 \leq i \leq n$
- A solution of this instance is $i_1, \ldots, i_k \in \{1, \ldots, n\}$ such that

$$u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k}$$

- PCP is undecidable!

Freeness of Pseudoknot-Generating

Theorem

For a given context-free language L, it is undecidable to determine whether or not L is $\mathbb{PK}_{\mathbb{R}}$-free.

Let $L = L_1 \cup L_2$, where

$$L_1 = \{\$ i_k i_{k-1} \cdots i_1 \$' \% \# u_{i_1} u_{i_2} \cdots u_{i_k} \#' \$' j_1 j_2 \cdots j_l \$ \% \#' v_{j_l}^R v_{j_{l-1}}^R \cdots v_{j_1}^R \#\},$$

$$L_2 = \{\$ i_k i_{k-1} \cdots i_1 \$' \% \# u_{i_1} u_{i_2} \cdots u_{i_k} \#'\},$$

for $k, l \geq 1, 1 \leq i_1, \ldots, i_k, j_1, \ldots, j_l \in \{1, \ldots, k\}$.

Freeness of Pseudoknot-Generating

Theorem

For a given context-free language L, it is undecidable to determine whether or not L is $\mathbb{PK}_{\mathbb{R}}$-free.

If there is a solution $i_1 i_2 \cdots i_k = j_1 j_2 \cdots j_l$ such that

$$u_{i_1} u_{i_2} \cdots u_{i_k} = v_{j_1} v_{j_2} \cdots v_{j_l},$$

L is not $\mathbb{PK}_{\mathbb{R}}$-free.

$$x \in L_2 \quad \overbrace{\$ \; i_k i_{k-1} \cdots i_1 \; \$'}^{w_1} \; \overbrace{\%}^{w_2} \; \overbrace{\# \; u_{i_1} u_{i_2} \cdots u_{i_k} \; \#'}^{w_3}$$

$$y \in L_1 \quad \underbrace{\$ \; i_k i_{k-1} \cdots i_1 \; \$'}_{w_1} \; \underbrace{\%}_{w_2} \; \underbrace{\# \; u_{i_1} u_{i_2} \cdots u_{i_k} \; \#'}_{w_3} \; \underbrace{\$' \; j_1 j_2 \cdots j_l \; \$}_{w_1^R} \; \underbrace{\%}_{w_4} \; \underbrace{\#' \; v_{j_1}^R v_{l-1}^R \cdots v_{j_1}^R \; \#}_{w_3^R}$$

Freeness of Pseudoknot-Generating

Theorem

For a given context-free language L, it is undecidable to determine whether or not L is $\mathbb{PK}_{\mathbb{R}}$-free.

If there is a solution $i_1 i_2 \cdots i_k = j_1 j_2 \cdots j_l$ such that

$$u_{i_1} u_{i_2} \cdots u_{i_k} = v_{j_1} v_{j_2} \cdots v_{j_l},$$

L is not $\mathbb{PK}_{\mathbb{R}}$-free.

PCP is undecidable, thus, it is undecidable!

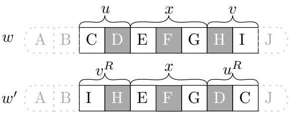
Summary of Freeness

- A given language L is $\underline{\mathbb{PI}\text{-free}}$ if $(\Sigma^* \cdot \mathbb{PI}(L) \cdot \Sigma^*) \cap L = \emptyset$.
  - Regular language L: Decidable in polynomial time
  - Context-free language L: Undecidable
- A given language L is $\underline{\mathbb{PK}_{\mathbb{R}}\text{-free}}$ if $L \cup \mathbb{PK}_{\mathbb{R}}(L) = \emptyset$.
  - Regular language L: Decidable in polynomial time
  - Context-free language L: Undecidable
- A given language L is $\underline{\mathbb{SDI}\text{-closed}}$ if $(L \overset{sdi}{\leftarrow} L) \subseteq L$.
  - Regular language L: Decidable in polynomial time
  - Context-free language L: Undecidable
- A given language L is $\underline{\mathbb{SDD}\text{-closed}}$ if $(L \overset{sdd}{\leftarrow} L) \subseteq L$.
  - Regular language L: Decidable in polynomial time
  - Context-free language L: Undecidable
- A given language L is $\underline{\mathbb{SDD}\text{-free}}$ if $x \overset{sdd}{\leftarrow} y = \emptyset$, where $x, y \in L$.
  - Regular language L: Decidable in polynomial time
  - Context-free language L: Undecidable

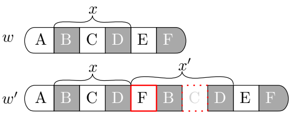
Outline

Summary of Thesis

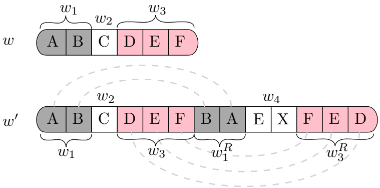
We have studied bio-inspired operations and their properties.



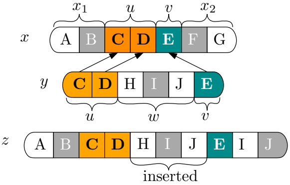
Pseudo-Inversion: Closure Properties and Decidability
in *Natural Computing*, 2016



Duplications and Pseudo-Duplications
in *International Journal of Unconventional Computing*, 2016



Pseudoknot-Generating Operation
in *Theoretical Computer Science*, 2017



Site-Directed Insertion
in *Theoretical Computer Science*, 2017

Outline

Future Works

Closure properties for iterated bio-inspired operations:

- For a bio-inspired operation ♣, iterated ♣ closure is not easy
- Finding a counter example to or a construction for iterated ♣

Bio-inspired operation on finite tree automata and tree grammars:

- Tree automata accept tree structures, while FA accept strings
- Tree automata were introduced in the 1900s to solve certain decision problems in logic
- Tree automata and tree grammars can be used to characterize structural properties of DNA (RNA)

Thank you for your attention!