CPM 2022

June 27, 2022

Session I

RePair Grammars are the Smallest Grammars for Fibonacci Words

Takuya Mieno (University of Electro-Communications)
Shunsuke Inenaga (Kyushu University)
Takashi Horiyama (Hokkaido University)

AFSA

Straight Line Program (SLP)

A context-free grammar in the Chomsky normal form that produces only a single string w is called a **straight-line program (SLP)** for w.

E.g.

The grammar $G = (V, \Sigma, \delta, S)$ below is an SLP which produces a single string $w =$ `ababaabaaba`.

$V = \{X_7, X_6, X_5, X_4, X_3, X_2, X_1, A, B\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}\}$, $S = X_7$,

$\delta = \{$

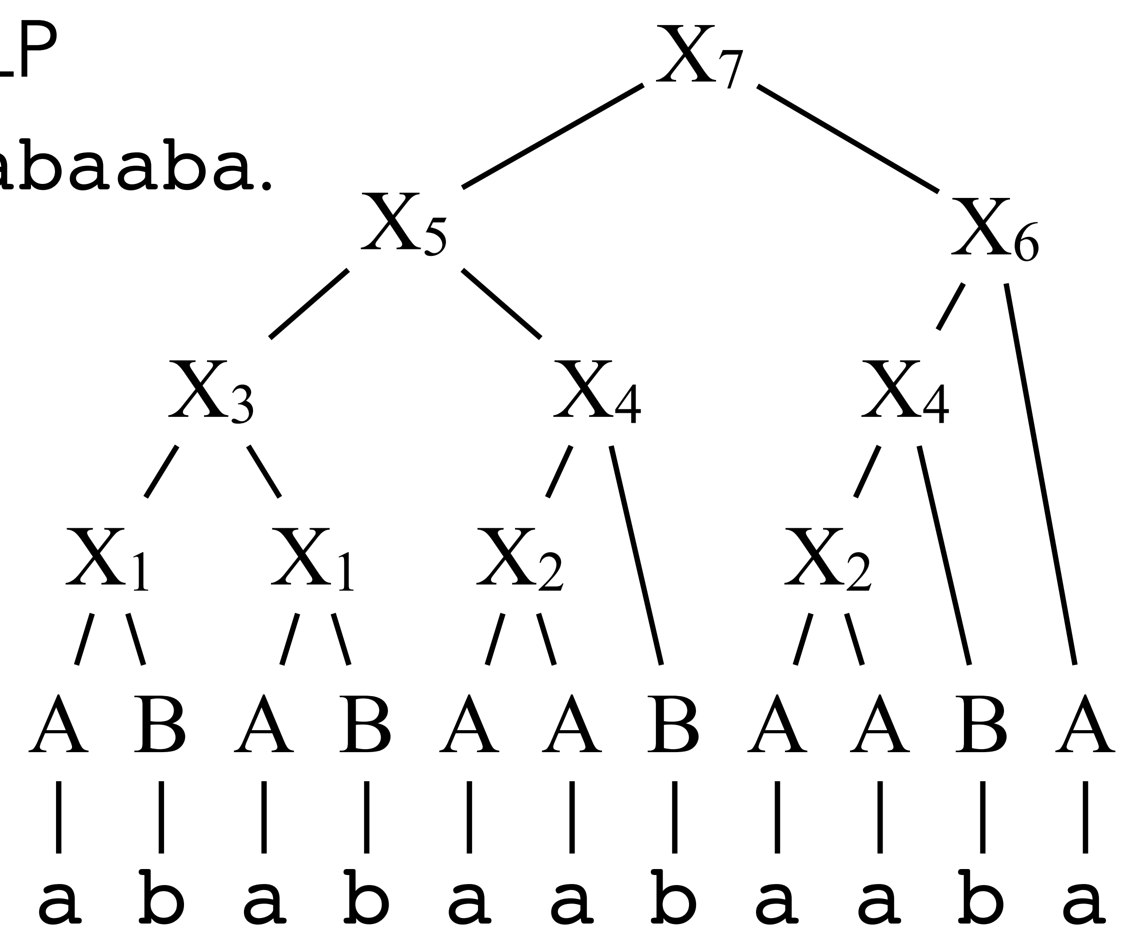
$X_7 \to X_5X_6$, $X_6 \to X_4\,A$, $X_5 \to X_3X_4$,

$X_4 \to X_2B$, $X_3 \to X_1X_1$, $X_2 \to AA$,

$X_1 \to AB$, $A \to \mathtt{a}$, $B = \mathtt{b}$

$\}$

$|G| = 9$



The ***size*** $|G|$ of an SLP G is **# of productions** in G.

Smaller grammar is better in terms of string compression.

Smallest Grammar Problem (SLP version)

Input : String w
Output : SLP for w with the **smallest** size.

This problem is known to be NP-hard [Charikar et al., '05].

Even if the alphabet size σ is a constant with $\sigma \geq 17$ [Casel et al., '21].

The hardness for $1 \leq \sigma \leq 16$ is open.

There are practical grammar compressors.

RePair, LZ78, BISECTION, SEQUENTIAL, etc.

Bounds for the *approximation ratios* of these compressors have been investigated [Charikar et al., '05], [Bannai et al., '21].

Our Result (informal statement)

We focus on the smallest grammars of ***Fibonacci words***:

$$F_1 = \mathtt{b},\ F_2 = \mathtt{a},\ F_i = F_{i-1}F_{i-2}\ (i \geq 3).$$

n	F_n	length
1	b	1
2	a	1
3	ab	2
4	aba	3
5	abaab	5
6	abaababa	8

Our main result

For any **Fibonacci word** F_n, the set of **smallest grammars** of F_n **equals** the set of grammars obtained by applying **RePair** algorithms to F_n.

To our knowledge, Fibonacci words are **the first non-trivial strings** whose smallest grammar sizes are computable in polynomial time (except trivial ones such as $\mathtt{a}^{2^k}$ and $(\mathtt{ab})^{2^k}$).

RePair [Larsson & Moffat, '99]

RePair is a greedy algorithm which computes an SLP for given string w.

1. Replace all terminals in w with non-terminals.
2. Recursively do the following while the length of string is ≥ 2:
   Choose **a most frequent bigram** and replace it with a non-terminal.

E.g.

$w = \mathtt{ababaaaab}$

↓

ABABAAAAB

↓

CCAAAAC

↓

CCDDC

↓

EDDC

↓

⋮

$$\delta = \{ A \to \mathtt{a},\ B \to \mathtt{b},\ C \to AB,\ D \to AA,\ E \to CC,\ \ldots \}$$

Replace terminals $\mathtt{a}$, $\mathtt{b}$ with non-terminals A, B.

# of occurrences of each bigram are
AA: 2, AB: **3**, BA: 2, BB: 0.
→ choose AB and replace it with non-terminal C.

# of occurrences of all bigrams are the same,
so we can choose a most frequent one ***arbitrarily***.
Here, choose CC and replace it with E.

RePair Grammars

RePair is a greedy algorithm which computes an SLP for given string w.
When there are multiple frequent bigram, we can choose any of them **arbitrarily**.

Which one is chosen depends on the implementation of RePair.

E.g.

Most frequent bigrams in $w = \mathtt{ababaabaaba}$ are **ab and ba.**

If we replace $\mathtt{ab}$ with X, the string changes to $\mathrm{XXaXaXa}$.

If we replace $\mathtt{ba}$ with X, the string changes to $\mathrm{aXXaXaX}$.

A grammar which is obtained by applying (any implementation of) RePair to w is called a **RePair grammar** of w.
We denote by RePair(w) the set of all RePair grammars of w.

Our Result (formal statement)

Let **Opt**(w) be the set of smallest grammars of w.

Theorem 1 [This work]

For every $n \geq 1$, Opt(F_n) = RePair(F_n) holds. Also, for every $n \geq 4$, $|$Opt$(\mathrm{F}_n)| = n - 2$ if n is even, and $|$Opt$(\mathrm{F}_n)| = n - 1$ if n is odd.

The proof consists of 3 steps. We will show that

Lemma 1 ◀ NEXT

the smallest size of grammars of F_n equals n,

Lemma 2

the size of **any** RePair grammar of F_n is the smallest, and

Lemma 3

the size of **any non-RePair** grammar of F_n is **not** the smallest.

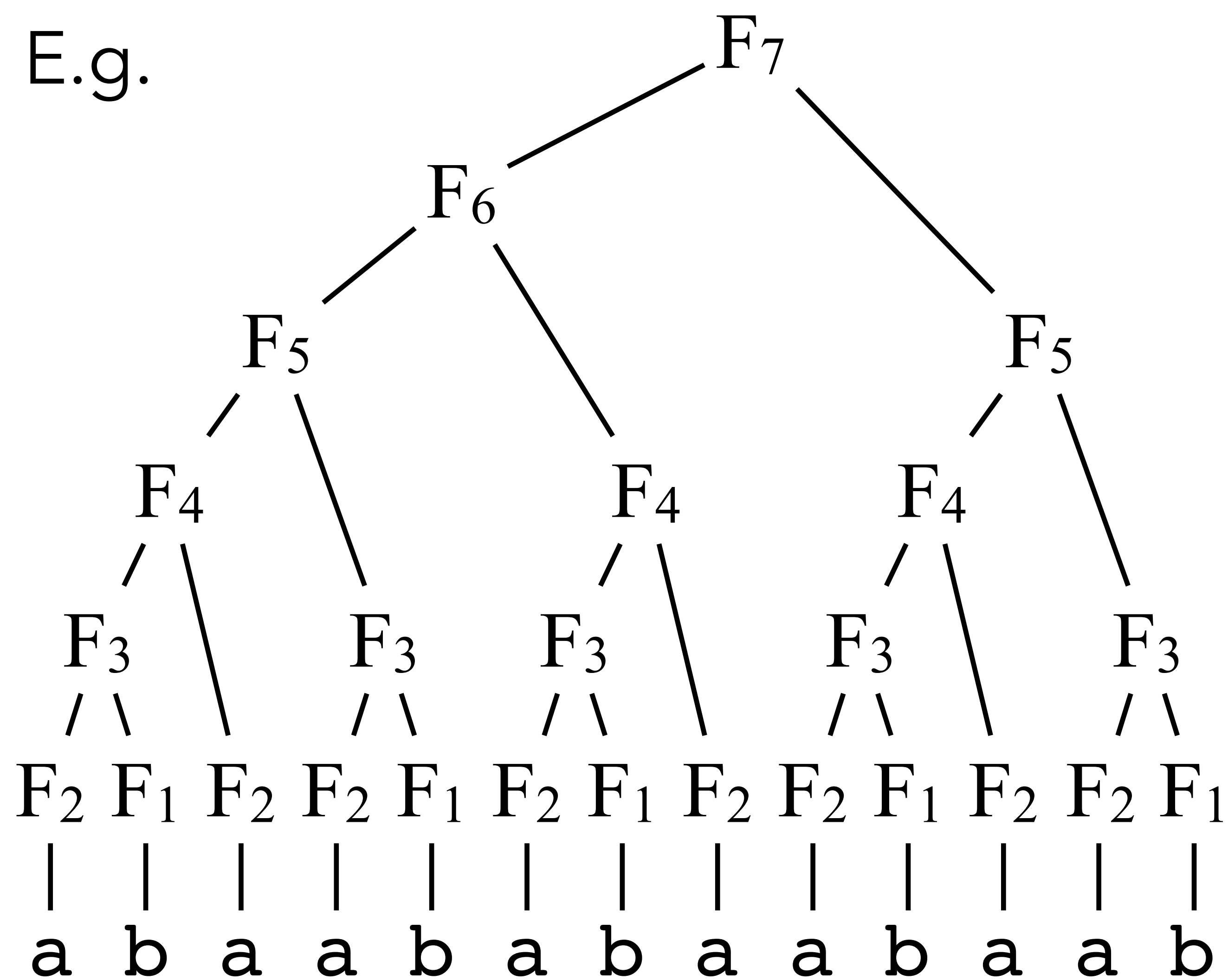
Smallest Grammar Size of Fibonacci Words

There is a size-n grammar by the definition of F_n.

$F_1 = \texttt{b}$, $F_2 = \texttt{a}$,

$F_i = F_{i-1}F_{i-2}$ $(i \geq 3)$.

E.g.



The remaining task is to prove that there is no grammar of size $< n$.

Relation Between Grammar and LZ-factorization

To evaluate the grammar size of F_n, we utilize the **LZ-factorization**.

Definition (LZ-factorization (without self-reference))

The **LZ-factorization** of string w is a sequence $(p_1, \ldots, p_z)$ of strings such that $w = p_1 \cdots p_z$ and each p_i is either a fresh symbol or the longest prefix of $p_i \cdots p_z$ which occurs in $p_1 \cdots p_{i-1}$.

Let $z(w)$ be the number of factors in the LZ-factorization of w.

E.g.

$$w = \mathtt{a}|\mathtt{b}|\mathtt{a}|\mathtt{a\,b\,a}|\mathtt{a\,b}|\mathtt{c}|\mathtt{a\,b\,a\,a}|\mathtt{c\,a\,b\,a\,a}|\mathtt{b\,a}$$

The size of LZ-factorization is $z(w) = 9$

Theorem 2 [Rytter, '03]

For any string w, $\boldsymbol{z(w) \le g^*(w)}$ holds.

$g^*(w)$: the size of the <u>smallest</u> grammar of w.

Proof of Lemma 1

Theorem 2 [Rytter, '03]

For any string w, $\boldsymbol{z(w) \leq g^*(w)}$ holds.

With a modification to the proof of the Theorem 2, we obtain a slightly tighter lower bound on the smallest grammar size:

Theorem 3 [This work]

For any string w, $\boldsymbol{z(w) - 1 + \sigma_w \leq g^*(w)}$ holds
where σ_w is the number of characters appearing in w.

Since $z(\mathrm{F}_n) = n - 1$ holds for Fibonacci words,
$z(\mathrm{F}_n) - 1 + \sigma_w = n - 1 - 1 + 2 = \boldsymbol{n \leq g^*(\mathrm{F}_n)}$ holds by Theorem 3.
Namely, the aforementioned grammar size n is the smallest. $\square$

Lemma 1

The size of the smallest grammar of F_n is n.

Lemmas for Our Main Result

Lemma 1

The size of the smallest grammar of F_n is n.

Lemma 2 ◀ NEXT

The size of **any** RePair grammar of F_n is n.

By Lemmas 1 and 2, RePair$(\mathrm{F}_n) \subseteq$ Opt(F_n) holds.

Lemma 3

The size of **any non-RePair grammar** of F_n is at least $n + 1$.

By combining Lemma 3, we obtain RePair$(\mathrm{F}_n) =$ Opt(F_n).

Most Frequent Bigrams in Fibonacci Words

The most frequent bigrams in F_n are ab and/or ba.

n	F_n	most frequent bigram(s)
3	ab	ab
4	aba	ab, ba
5	abaab	ab
6	abaababa	ab, ba
7	abaababaabaab	ab
8	abaababaabaababaababa	ab, ba
9	abaababaabaababaababaabaababaabaab	ab
10	abaababaabaababaababaabaababaabaababaababaabaababaababa	ab, ba

E.g. # of occurrences of bigrams in $\mathrm{F}_7 =$ abaababaabaab
ab: 5, ba: 4, aa: 3, bb: 0

There are two possible cases, depending on which bigram is chosen.

When ab is Chosen

Let's observe how Fibonacci words change when ab is replaced to another symbol.

E.g. The most frequent bigram in F_7 is (only) ab.

$$F_7 = \texttt{abaababaabaab}$$

Replace all ab's with symbol X, then the string changes to

$$X\ \texttt{a}\ X\ X\ \texttt{a}\ X\ \texttt{a}\ X$$

$$F_6 = \texttt{a}\ \texttt{b}\ \texttt{a}\ \texttt{a}\ \texttt{b}\ \texttt{a}\ \texttt{b}\ \texttt{a}$$

This is *isomorphic* to $F_6 = \texttt{abaababa}$.

In general, Lemma 4 holds.

Lemma 4

By replacing all occurrences of ab in F_n with a new symbol, we obtain a string which is isomorphic to F_{n-1}.

When ab is Chosen

Lemma 4

By replacing all occurrences of ab in F_n with a new symbol, we obtain a string which is isomorphic to F_{n-1}.

Lemma 4 can be easily proven by using another definition of F_n:

$$\mathrm{F}_n = \phi^{n-1}(\mathtt{b})$$

where ϕ is the *string morphism* such that ϕ(a) = ab and ϕ(b) = a.

E.g.

$$\mathrm{F}_6 = \text{a b a a b a b a}$$

$$\phi \downarrow \qquad \uparrow \text{Replace ab} \rightarrow \text{a (and then a} \rightarrow \text{b)}$$

$$\mathrm{F}_7 = \text{abaababaabaab}$$

Essentially, the replacement in Lemma 4 is the **inverse of ϕ**.

→ Of course, the resulting string is isomorphic to F_{n-1}. □

When ba is Chosen

Let's observe how Fibonacci words change when ba is replaced with another symbol.

E.g. The most frequent bigrams in F_8 are ab and ba.

$$F_8 = \texttt{abaababaabaababaababa}$$

Replace all ba's with symbol X, then the string changes to

$$\texttt{a X a X X a X a X X a X X}$$

$$F_7 = \boxed{}\ \texttt{a b a a b a b a a b a a}\ \boxed{\texttt{b}}$$

This is isomorphic to the ***right-rotation*** R_7 of F_7.

$$R_n := F_n[|F_n|] \cdot F_n[1..\, |F_n|-1]$$

In general, Lemma 5 holds.

Lemma 5

By replacing all occurrences of ba in F_{2k} with a new symbol, we obtain a string which is isomorphic to the right-rotation R_{2k-1} of F_{2k-1}.

Applying RePair to Right-rotation of Fibonacci Words (1/2)

n	$R_n := F_n[\lvert F_n\rvert] \cdot F_n[1..\lvert F_n\rvert{-}1]$	most frequent bigram
3	ba	ba
4	aab	ab
5	babaa	ba
6	aabaabab	ab
7	babaababaabaa	**ba**
8	aabaababaabaababaabab	ab
9	babaababaabaababaababaabaababaabaa	ba
10	aabaababaabaababaababaabaababaabaababaababaabaababaabab	ab

$R_7 =$ babaababaabaa
$\rightarrow$ XXaXXaXa
(aabaabab)

Lemma 6

By replacing all occurrences of ba in R_{2k+1} with a new symbol, we obtain a string which is isomorphic to R_{2k}.

Applying RePair to Right-rotation of Fibonacci Words (2/2)

n	$R_n := F_n[\lvert F_n\rvert] \cdot F_n[1..\lvert F_n\rvert-1]$	most frequent bigram
3	ba	ba
4	aab	ab
5	babaa	ba
6	aabaabab	**ab**
7	babaababaabaa	ba
8	aabaababaabaababaabab	ab
9	babaababaabaababaababaabaababaabaa	ba
10	aabaababaabaababaababaabaababaabaababaababaabaababaabab	ab

$R_6 =$ aabaabab
$\to$ aXaXX
(babaa)

Lemma 7

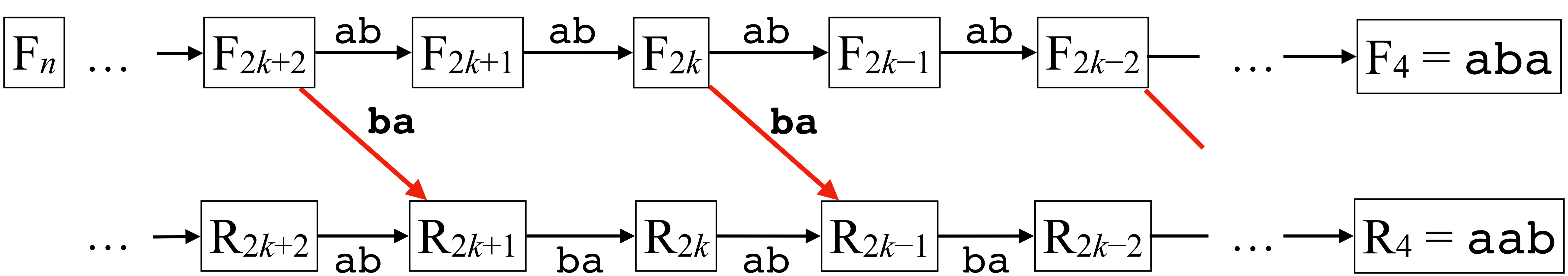
By replacing all occurrences of ab in R_{2k} with a new symbol, we obtain a string which is isomorphic to R_{2k-1}.

"RePair Graph" for Fibonacci words (1/2)

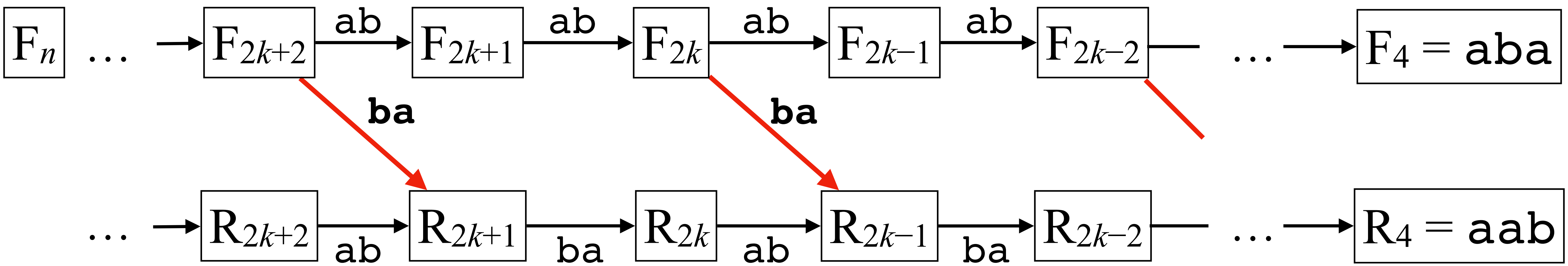
From Lemmas 4–7, the strings that appear during the execution of RePair for a Fibonacci word are strings that are isomorphic to

- some Fibonacci word F_i , or
- the right-rotation R_i of some Fibonacci word.

The **changes** of strings when RePair is applied to Fibonacci words can be represented by the following "***RePair graph***".



"RePair Graph" for Fibonacci words (2/2)



The size of a grammar equals **the length of the corresponding path** from the left-end (F_n) to the right-end (F_4 or R_4) of the RePair graph.
→ That always equals $\boldsymbol{n}$, i.e., the smallest size.

The number of RePair grammars equals **the number of paths** from the left-end (F_n) to the right-end (F_4 or R_4) of the RePair graph.
→ That equals $\boldsymbol{n-2}$ if n is even, or $\boldsymbol{n-1}$ otherwise. □

Lemmas for Our Main Result

Lemma 1 (omit the details in this talk)

The size of the smallest grammar of F_n is n.

Lemma 2

The size of **any** RePair grammar of F_n is n.

By Lemmas 1 and 2, $\mathsf{RePair}(\mathrm{F}_n) \subseteq \mathsf{Opt}(\mathrm{F}_n)$ holds.

Lemma 3 ◀ NEXT

The size of **any non-RePair grammar** of F_n is at least $n+1$.

By combining Lemma 3, we obtain $\mathsf{RePair}(\mathrm{F}_n) = \mathsf{Opt}(\mathrm{F}_n)$.

16 Strategies in non-RePair Algorithms

Bigram to replace / Target string	aa		ab		ba	
	all	not all	all	not all	all	not all
F_{2k}	1	2	RePair	3	RePair	4
F_{2k+1}			RePair		5	6
R_{2k}	7	8	RePair	9	10	11
R_{2k+1}	12	13	14	15	RePair	16

For all **16 strategies** to replace bigrams which do not satisfy the RePair conditions, the resulting grammars are non-smallest.

To prove this, we heavily utilize the fact "The smallest grammar size is lower-bounded by the LZ77 size" (Theorem 3).

Lemmas for Our Main Result

Lemma 1 (omit the details in this talk)

The size of the smallest grammar of F_n is n.

Lemma 2

The size of **any** RePair grammar of F_n is n.

By Lemmas 1 and 2, $\mathsf{RePair}(\mathrm{F}_n) \subseteq \mathsf{Opt}(\mathrm{F}_n)$ holds.

Lemma 3

The size of **any non-RePair grammar** of F_n is at least $n+1$.

By combining Lemma 3, we obtain $\mathsf{RePair}(\mathrm{F}_n) = \mathsf{Opt}(\mathrm{F}_n)$.

Conclusion and Future Work

Conclusion

We showed that the **smallest grammars** are the same as the **RePair grammars** of Fibonacci words F_n.

The smallest grammars of F_n are characterized completely.

Theorem 1 [This work]

For every $n \geq 1$, $\mathsf{Opt}(\mathrm{F}_n) = \mathsf{RePair}(\mathrm{F}_n)$ holds. Also, for every $n \geq 4$, $|\mathsf{Opt}(\mathrm{F}_n)| = n - 2$ if n is even, and $|\mathsf{Opt}(\mathrm{F}_n)| = n - 1$ if n is odd.

Future Work

To investigate whether it is possible to characterize the smallest grammars of other binary words, including <u>Thue-Morse words</u> TM_n and <u>Period-doubling words</u> PD_n. We have confirmed that

$$\mathsf{Opt}(\mathrm{TM}_n) \neq \mathsf{RePair}(\mathrm{TM}_n) \text{ and } \mathsf{Opt}(\mathrm{PD}_n) \neq \mathsf{RePair}(\mathrm{PD}_n).$$