

2025/9/24 STRセミナー2025

貪欲LZ-Start-End分解の 線形時間アルゴリズム

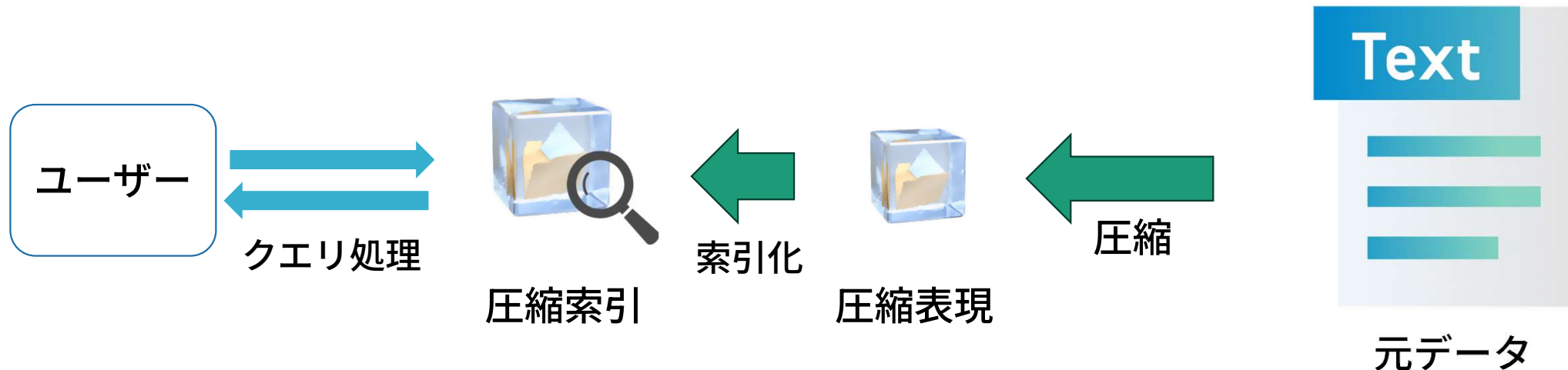
柴田 紘希（九州大学）

データ圧縮と圧縮索引

■ データ圧縮: データを小さな圧縮表現に変換する手法

- 例: ランレングス圧縮・LZ圧縮・文法圧縮・etc...

■ 圧縮索引: 圧縮表現の構造を利用した省領域な索引構造



圧縮性能と圧縮索引化のしやすさの関係

- 圧縮性能・圧縮索引化のしやすさはトレードオフ
- 小さい圧縮表現であるほど、サイズを維持した圧縮索引化は難しい



LZ77圧縮・文法圧縮の関係性

LZ77圧縮・文法圧縮のトレードオフ:

- 圧縮性能: LZ77圧縮の方が良い
- 圧縮索引化のしやすさ: 文法圧縮の方が良い

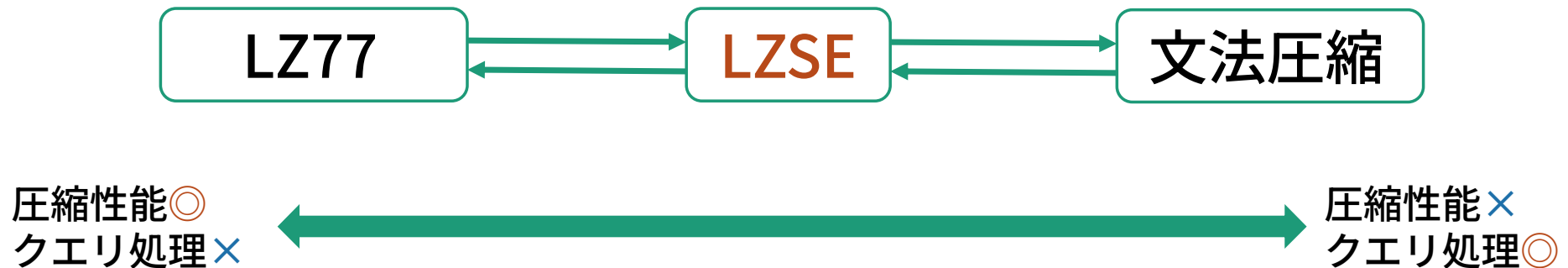
→ 圧縮性能を維持しながら、圧縮索引化にも対応した手法が欲しい！



本研究の成果: LZ-Start-End (LZSE) 圧縮の提案

LZSE: 圧縮性能とクエリ処理能力を両立した制約付きLZ77

- 圧縮性能: LZ77～文法圧縮の間に位置
- 圧縮索引: 圧縮前データへの高速アクセスが線形サイズの圧縮索引で可能



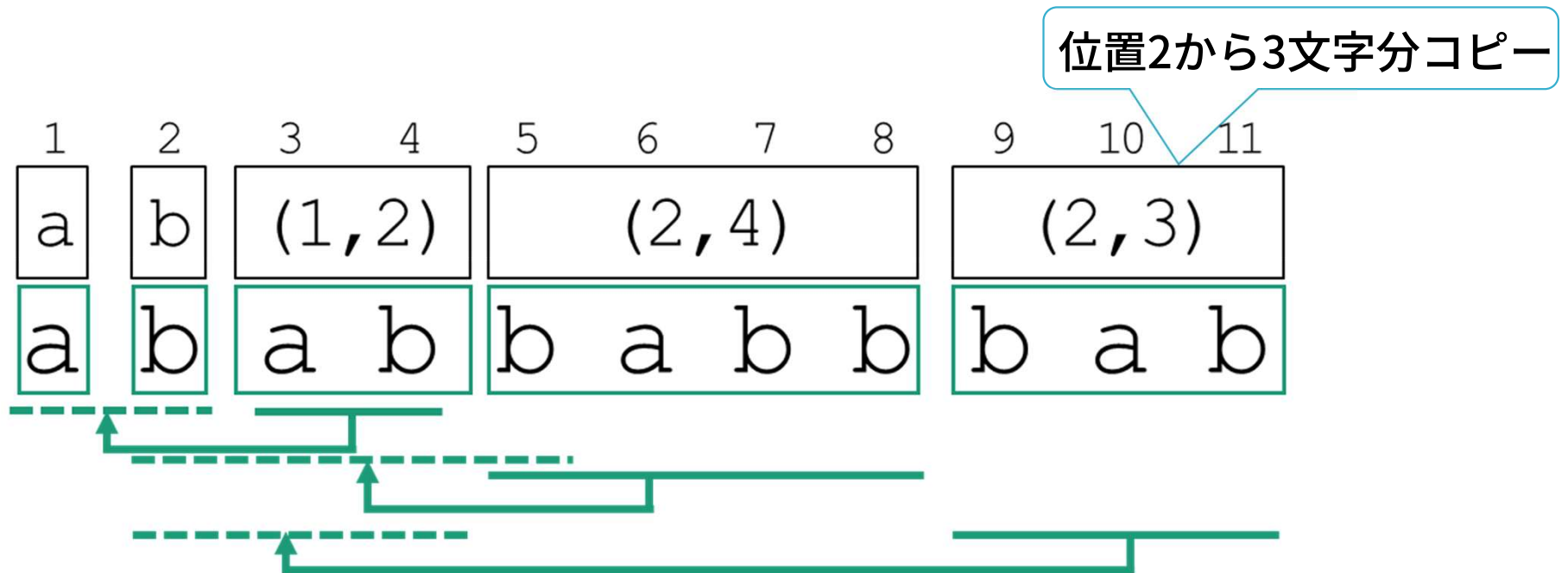
研究の成果・発表の内容

1. LZSEと他の圧縮手法の関係性の解析
 - LZSE $\leftrightarrow$ 文法圧縮の変換手法
 - LZSEと文法圧縮の圧縮力の差の理論解析
 2. 分解サイズの線形領域での圧縮索引を設計
 3. 貪欲LZSE分解の線形時間アルゴリズムを提案
 4. 貪欲LZSEの非最適性と最適LZSEとの差の下界を示す
- etc...

LZ77圧縮 [Ziv and Lempel, 1977]

データを以下のいずれかを表すfactorの列（**LZ77分解**）で表現:

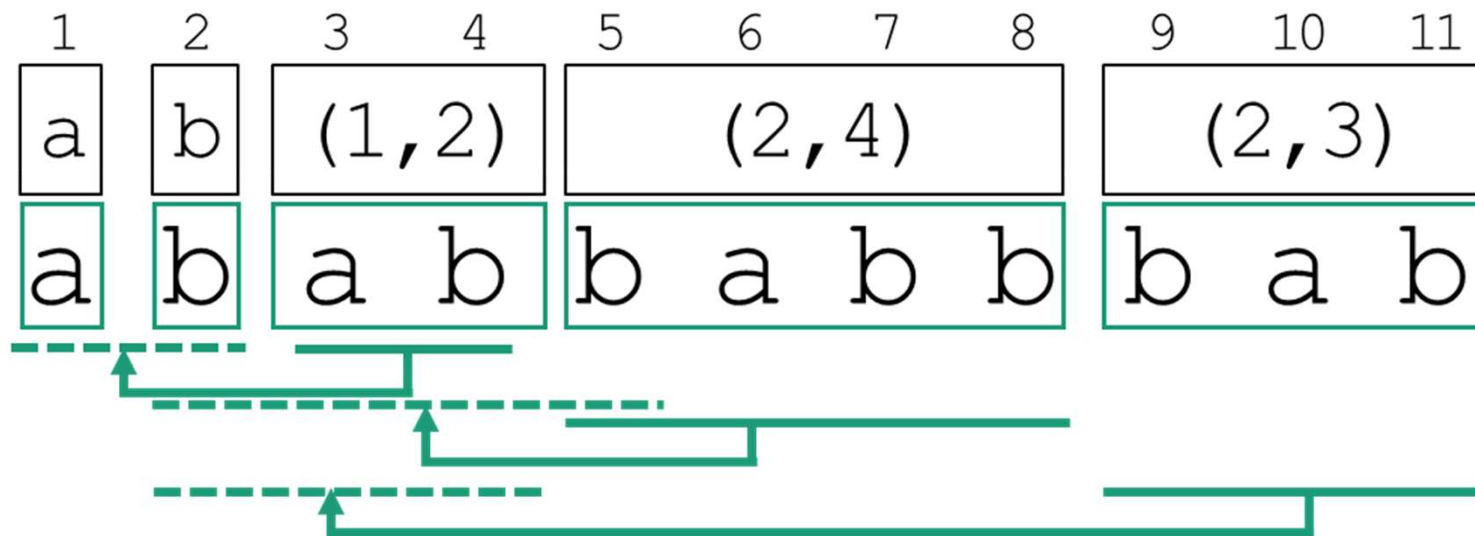
1. 1文字の保存
2. 左側に現れる同じ部分文字列へのコピー



LZ77圧縮 [Ziv and Lempel, 1977]

■ 1つの文字列に対するLZ77分解は一般に複数存在する

- 最適LZ77分解: factor数最小のLZ77分解
- 貪欲LZ77分解: 左側から貪欲に一番長いfactorを選ぶLZ77分解



LZ77圧縮 [Ziv and Lempel, 1977]

■ 1つの文字列に対するLZ77分解は一般に複数存在する

- **最適LZ77分解**: factor数最小のLZ77分解
- **貪欲LZ77分解**: 左側から貪欲に一番長いfactorを選ぶLZ77分解

定理

任意の文字列に対して、貪欲LZ77 = 最適LZ77 が成り立つ

定理

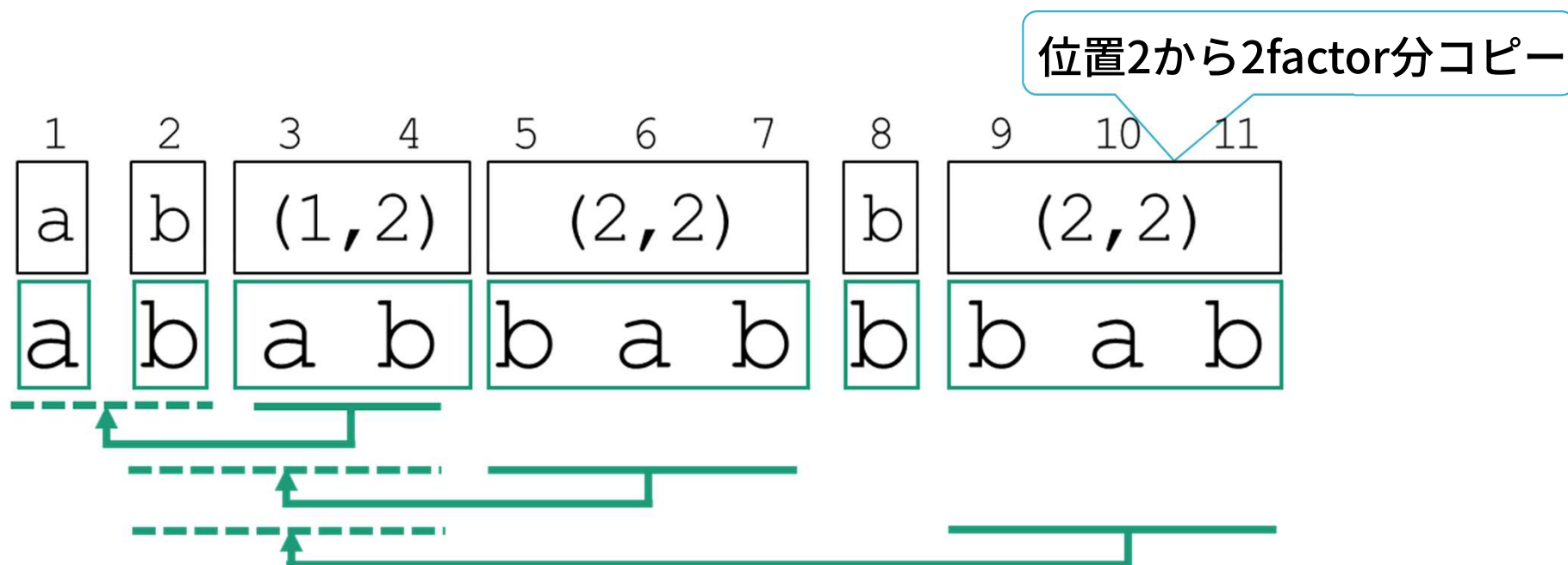
貪欲LZ77分解 = 最適LZ77分解は $O(n)$ 時間で計算できる

n : 入力文字列の長さ

新たな圧縮手法: LZ-Start-End (LZSE) 圧縮

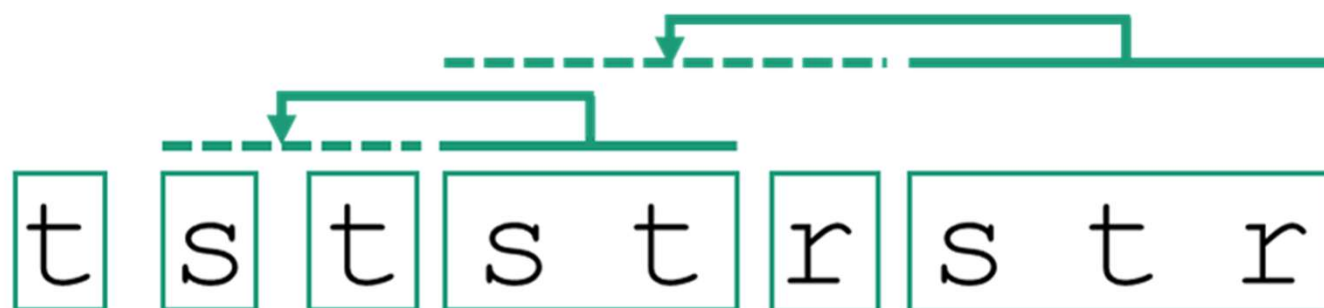
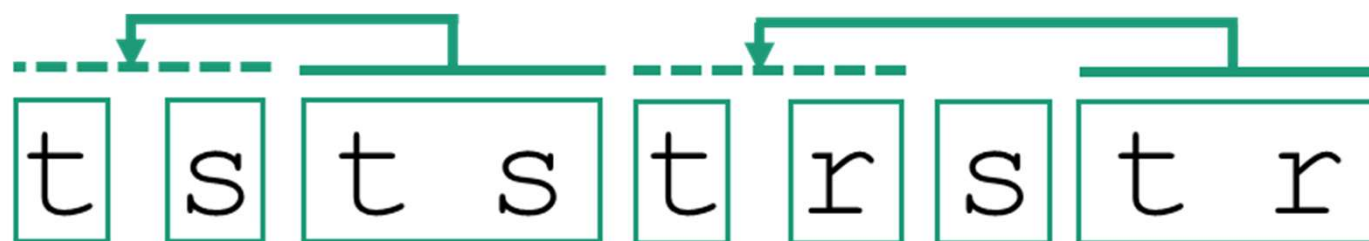
データを以下のいずれかを表すfactorの列（**LZSE分解**）で表現:

- 1文字の保存
- 左側に現れる連続するfactorへのコピー



貪欲LZSE分解 vs 最適LZSE分解

- LZ77分解と同様に、1つの文字列に対するLZSE分解は一般に複数存在する



定理

貪欲LZSE分解 = 最適LZSE分解 が常に成り立つとは限らない

最適LZSE分解の困難性

- 一方で、以下の定理が成り立つ:

定理

最適LZSE分解の計算はNP困難

- 最適解の計算が難しいため、高速に計算できる貪欲アルゴリズムが重要

貪欲LZSE分解の高速計算の難しさ

- **目標:** 貪欲LZSE分解を $O(n)$ 時間で計算したい！
- **難しさ:** LZSEでは、コピーできる文字列が扱いづらい構造を持っている！
 1. コピー可能な文字列の個数が大きく、最悪で $O(n^2)$ 要素になってしまう
 - ▲ LZ78・LZMWではこれが $O(n)$ に収まる → コピー可能な文字列集合を陽に持てた！
 2. コピー可能な文字列の条件である「連続factorの連結」に良い性質がない
 - ▲ LZ77ではコピーの条件が単純 → 陽に持たずとも簡単に扱えた！

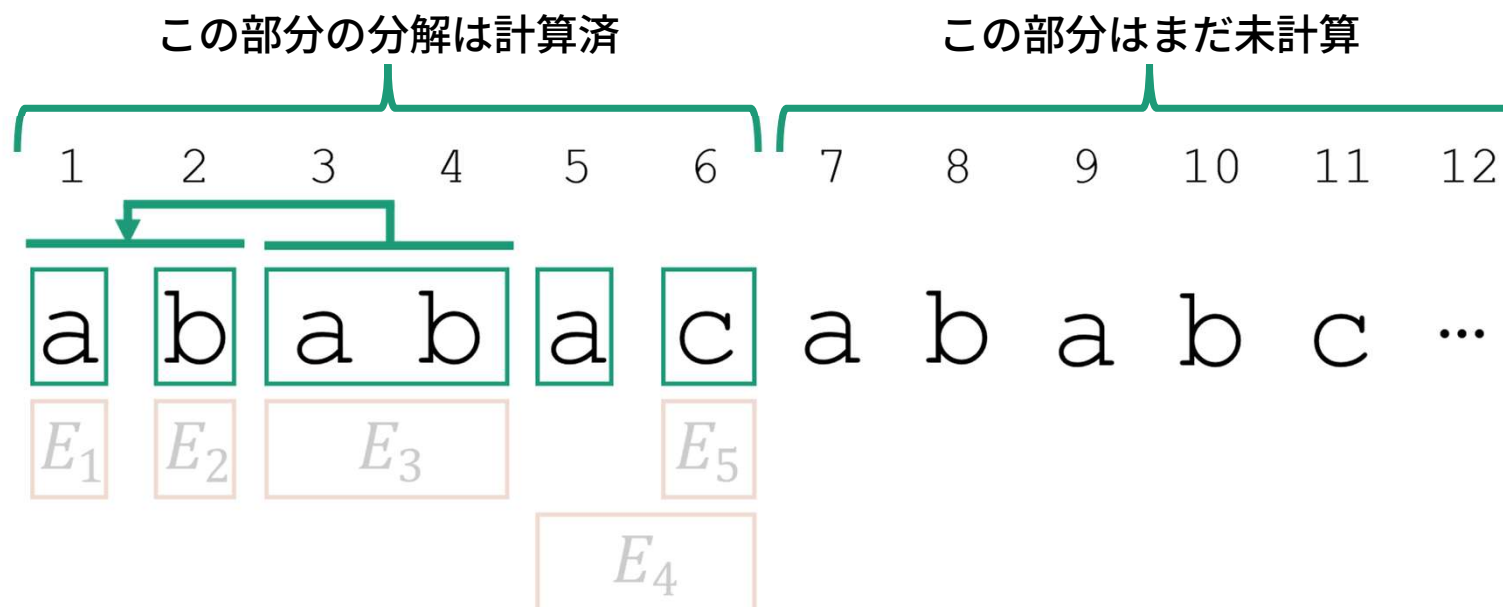
解決策: 貪欲LZSE分解特有の性質を用いて参照元候補の数を絞る！

Extended Factor

p 文字目までの貪欲LZSE分解 $T[1..p] = F_1 \cdots F_k$ が求まっている状態を考える

■ **Extended Factor:** 以下のように定義される文字列 $E_1, E_2, \dots, E_k$

- F_i が $E_1, \dots, E_{i-1}$ の中出现していないなら $E_i = F_i$
- 出現しているなら $E_i = F_i F_{i+1}$



Extended Factorのもつ良い性質

性質1

貪欲LZSE分解では、Extended Factorの各要素は
高々2回しか重複して出現しない

性質2

貪欲LZSE分解の各Factorの参照元（複数あり得る場合最左のもの）は
必ずExtended Factorを接頭辞として含んでいる

証明はスライドのAppendixに載せてます 😊

性質2の嬉しさについて

性質2

貪欲LZSE分解の各Factorの参照元（複数あり得る場合最左のもの）は必ずExtended Factorを接頭辞として含んでいる



対偶をとる

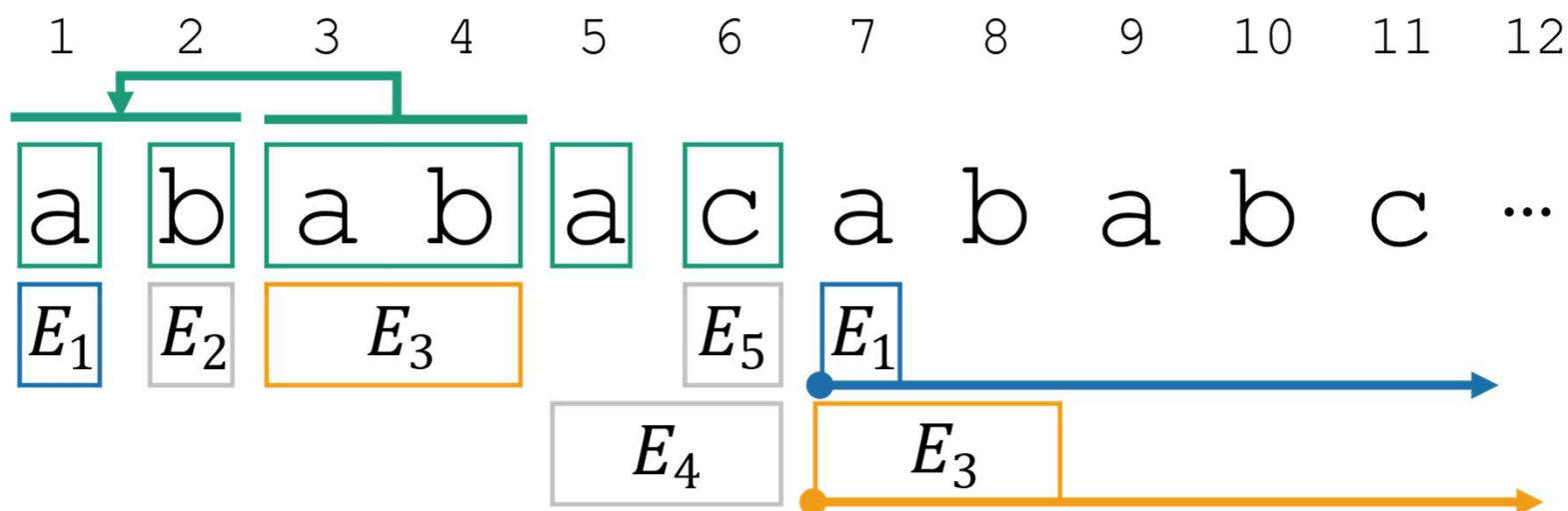
性質2*

Extended Factorを接頭辞として含まないような部分文字列は貪欲LZSE分解の各Factorの参照元（のうち最左のもの）にならない

性質2の嬉しさについて

性質2*

Extended Factorを接頭辞として含まないような部分文字列は
貪欲LZSE分解の各Factorの参照元（のうち最左のもの）にならない

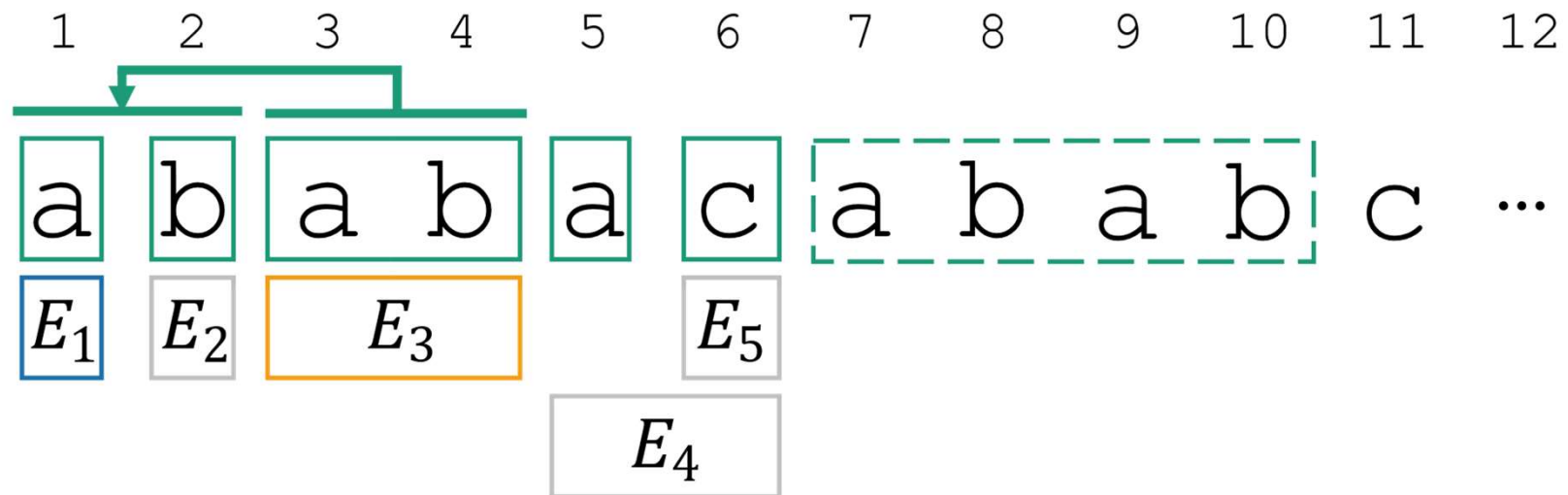


上図の例では、 E_1, E_3 から始まる2つしか候補にしなくてよい！

最長Factorを求める高速なアルゴリズム

p 文字目までの貪欲LZSE分解が既知のとき、次のようにして次のFactorを求められる:

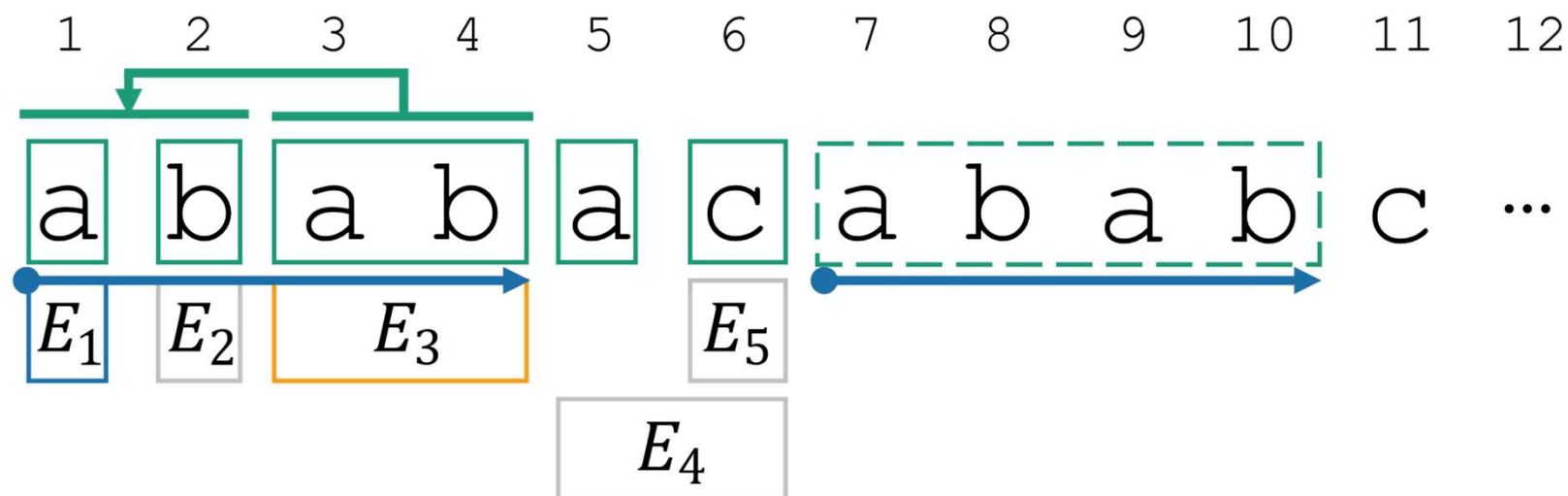
1. $T[p + 1..]$ の接頭辞として現れるExtended Factor (参照開始位置の候補) を列挙する
2. 各候補について、そのExtended Factorを参照元の始点にした場合のFactor長を求める
3. 2. の手続きで得られた最長Factorを F_{k+1} とする



最長Factorを求める高速なアルゴリズム

p 文字目までの貪欲LZSE分解が既知のとき、次のようにして次のFactorを求められる:

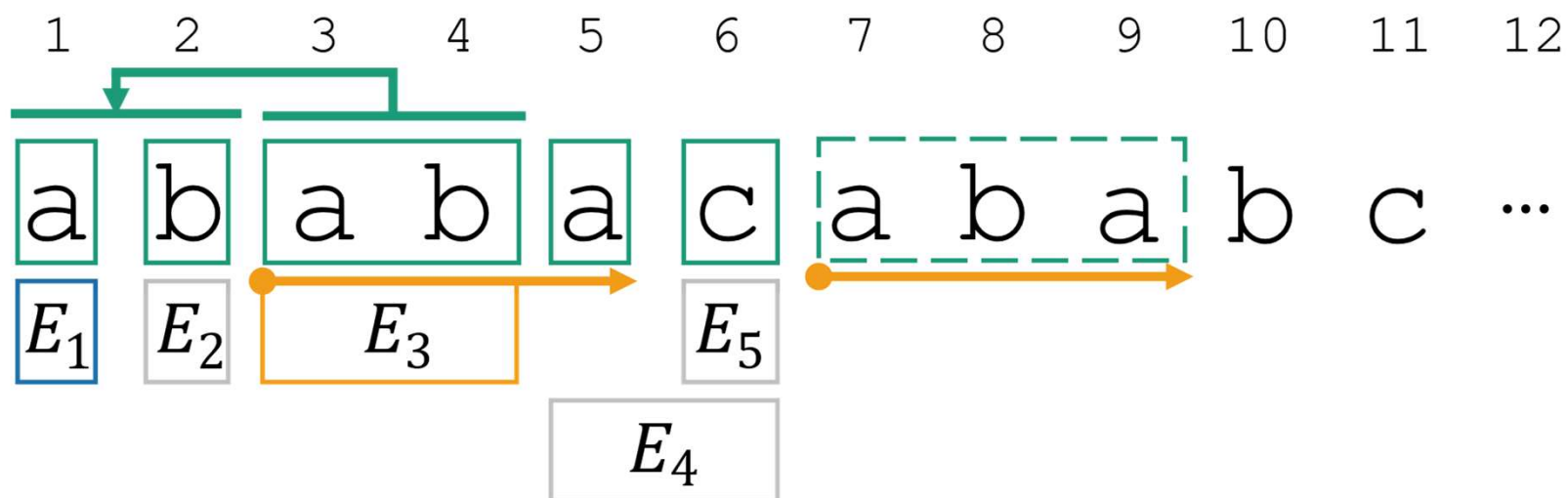
1. $T[p + 1..]$ の接頭辞として現れるExtended Factor (参照開始位置の候補) を列挙する
2. 各候補について、そのExtended Factorを参照元の始点にした場合のFactor長を求める
3. 2. の手続きで得られた最長Factorを F_{k+1} とする



最長Factorを求める高速なアルゴリズム

p 文字目までの貪欲LZSE分解が既知のとき、次のようにして次のFactorを求められる:

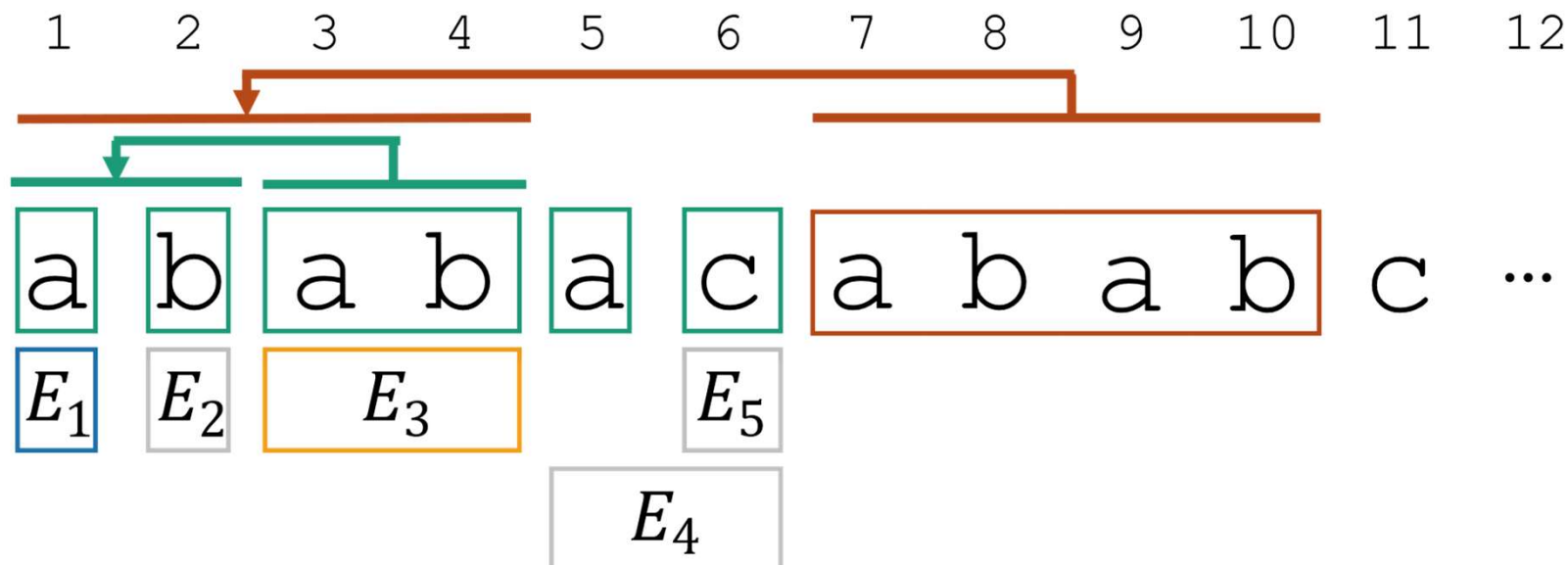
1. $T[p + 1..]$ の接頭辞として現れるExtended Factor (参照開始位置の候補) を列挙する
2. 各候補について、そのExtended Factorを参照元の始点にした場合のFactor長を求める
3. 2. の手続きで得られた最長Factorを F_{k+1} とする



最長Factorを求める高速なアルゴリズム

p 文字目までの貪欲LZSE分解が既知のとき、次のようにして次のFactorを求められる:

1. $T[p + 1..]$ の接頭辞として現れるExtended Factor (参照開始位置の候補) を列挙する
2. 各候補について、そのExtended Factorを参照元の始点にした場合のFactor長を求める
3. 2. の手続きで得られた最長Factorを F_{k+1} とする



アルゴリズムの計算量

- 参照元候補になるExtended Factorの列挙: 合計で $O(n)$ 時間
 - 実は参照元候補のExtended Factorの個数の総和が $O(n)$ で抑えられる！
 - ▲ 「同じ文字列を表すExtended Factorが高々2個」（性質1）から示せる！
 - 列挙は適切なデータ構造で1つあたり定数時間になる（詳細略）
- 参照元の位置を決めた際の最長Factor長計算: 1つあたり **定数時間**
 - 最長共通接頭辞の計算ができればよく、これも適切なデータ構造が使える

定理

貪欲LZSE分解は $O(n)$ 時間で計算できる

発表のまとめ

■ 研究の成果: LZ圧縮の制約付き版であるLZSE圧縮を提案

- LZ圧縮～文法圧縮の中間に位置し、文法圧縮より強い圧縮性能を持つ
- 文法圧縮と同じ速度で圧縮領域でのデータアクセスが可能
- 貪欲LZSE分解は線形時間で計算できる

■ Open Problem: 貪欲LZSE vs 最適LZSE の分解サイズ比の最大値は？

- 論文中に挙げた例では2倍
- Thue-Morse文字列の接頭辞でもう少し良い比が出るが、それも定数倍っぽい
- オーダーレベルで差が出る例は存在するのか？

Appendix

1. 性質1の証明
2. 性質2の証明
3. 文法 $\rightarrow$ LZSE の（簡単な）変換手法と変換によるサイズ変化
4. LZSE $\rightarrow$ 文法の変換手法
5. LZSE / 文法 の圧縮能力の差

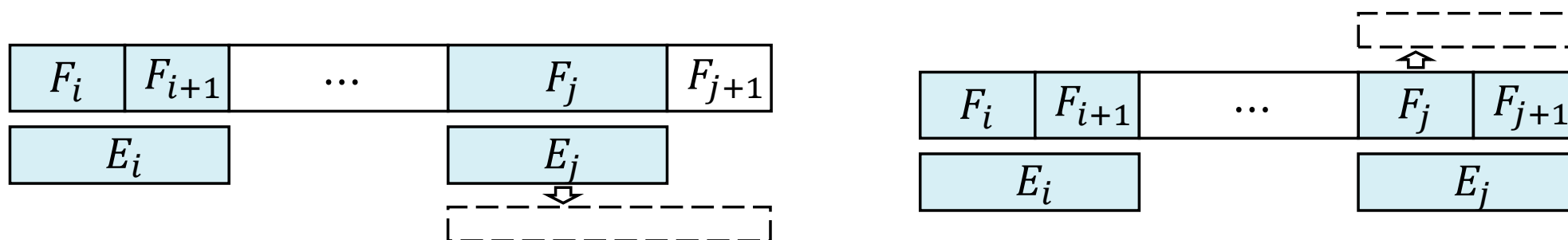
性質1の証明

性質1

貪欲LZSE分解では、Extended Factorの各要素は
高々2回しか重複して出現しない

証明: 隣接しないExtended Factorの組 E_i, E_j ($i + 1 < j$) が一致しないことを背理法で示す:

1. $E_i = E_j = F_j$ の場合: $F_j = E_i$ の場合 E_j は $E_j = F_j F_{j+1}$ と定義されるため、定義と矛盾
2. $E_i = E_j = F_j F_{j+1}$ の場合: F_j としてより長いFactor E_j を選べるため、貪欲に分解したことに矛盾



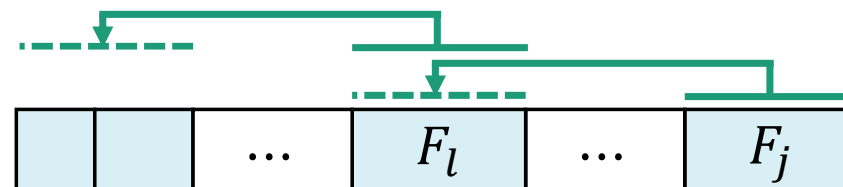
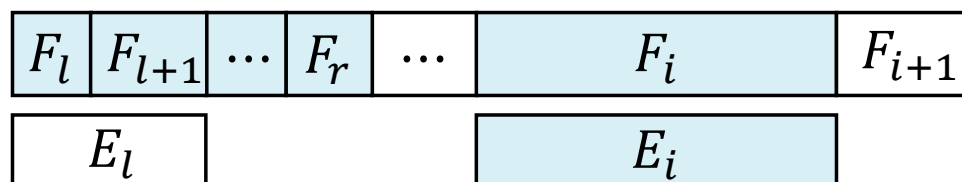
性質2の証明

性質2

貪欲LZSE分解の各Factorの参照元（複数あり得る場合最左のもの）は必ずExtended Factorを接頭辞として含んでいる

証明: Factor F_i が区間コピーのfactorであり、その最左の参照元が $F_l \cdots F_r = F_i$ だと仮定して示す:

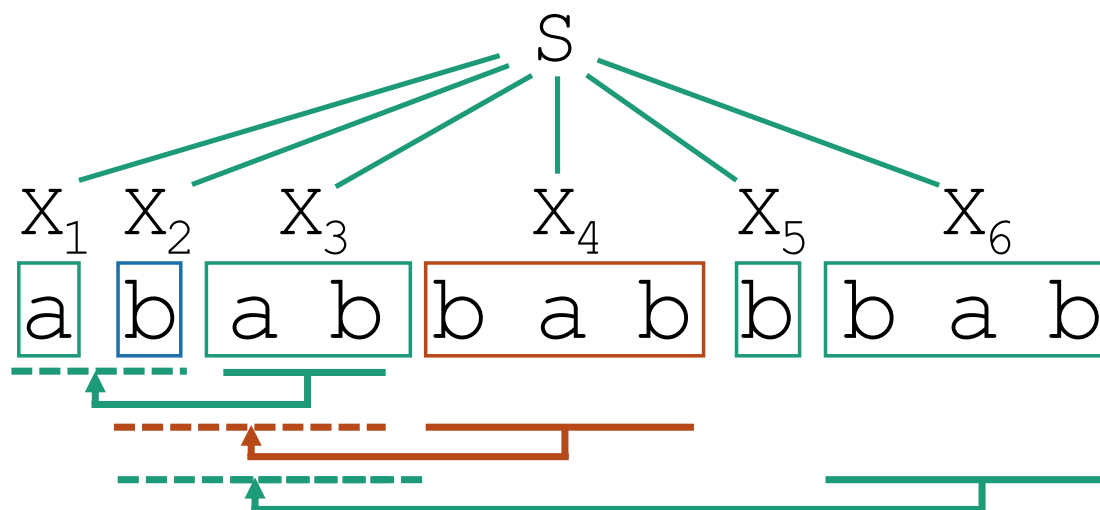
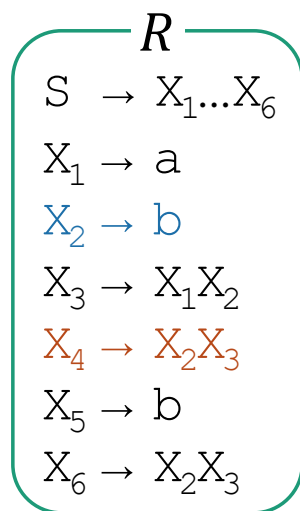
1. コピー元が複数Factorの場合（ $l \neq r$ の場合）: Extended Factorの定義から必ず成り立つ
2. コピー元が単一Factor F_l の場合: F_l が F_i の最左の参照元であるはずだが、 F_l のコピー元にも $F_i = F_l$ と同じ文字列が連続Factorの連結として現れるため、最左の参照元を取ったことに矛盾



LZSE→文法の（簡単な）変換手法と圧縮索引化

各factorを1つの非終端記号に対応付けて、LZSE分解から文法を構成できる

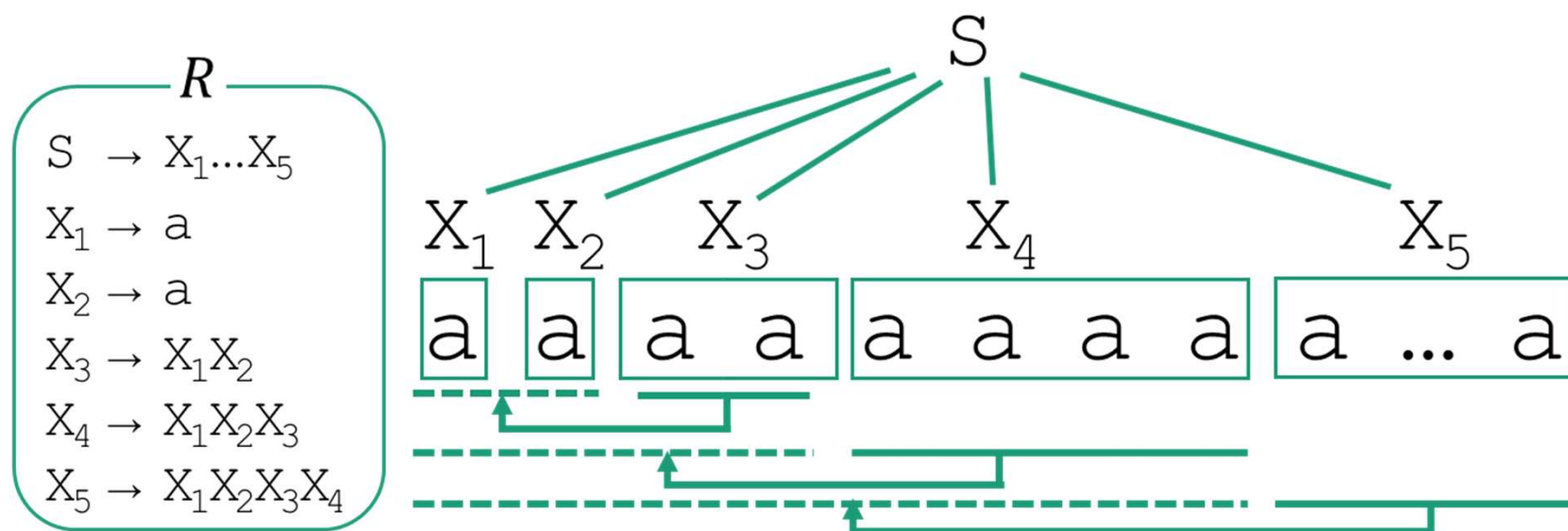
→ 文法に似た構造を持つことから、文法への手法の拡張で圧縮索引が作れる！
 （詳細は略）



文法への変換によるサイズの変化

前ページの方法で構成した文法はLZSEよりサイズが大きくなるかも

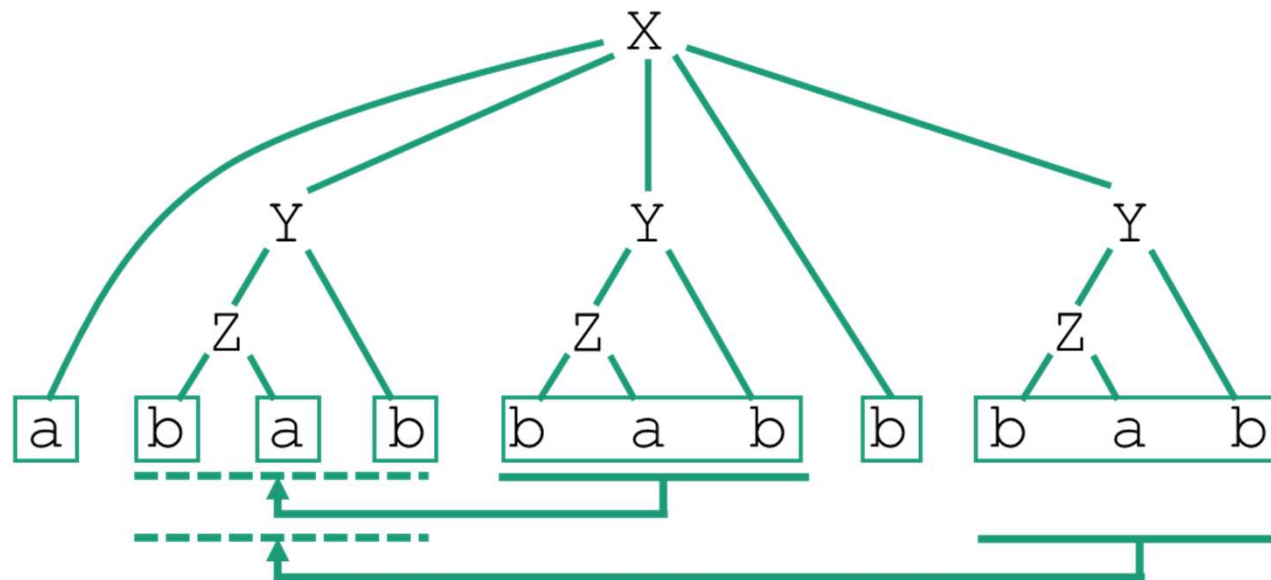
■ 下のような例が最悪ケースで、サイズが二乗オーダーで増えてしまう



文法 → LZSE分解

Grammar Decomposition [Rytter, 2003]: 文法→LZ分解の変換手法

- 各非終端記号の非最左出現を、その記号の最左出現へのコピーとして表す
- 変換で得られたLZ分解のサイズは、元の文法圧縮のサイズ以下
- 実は、この手法で得られるLZ分解がLZSE分解になっている！



文法圧縮・LZSE圧縮の圧縮性能の差

定理

十分大きな整数 m について、以下の条件を満たす長さ $\Theta(m^2)$ の文字列 T が存在する:

- T の貪欲LZSE分解のサイズが $\Theta(m)$
- T の最小文法のサイズが $\Omega(m\alpha(m))$

この定理より、LZSEが文法圧縮より（オーダーレベルで）**真に強い圧縮性能を持っている**といえる！

- 任意の文字列について、最小LZSE分解サイズ $\leq$ 最小文法圧縮サイズ
- 特定の文字列について、最小LZSE分解サイズ $\ll$ 最小文法圧縮サイズ

実はこの比はタイトで、同じ問題からの帰着により文法 $\rightarrow$ LZSEを（前ページより良い方法で）構成できる

$\alpha(x)$: アッカーマン関数の逆関数