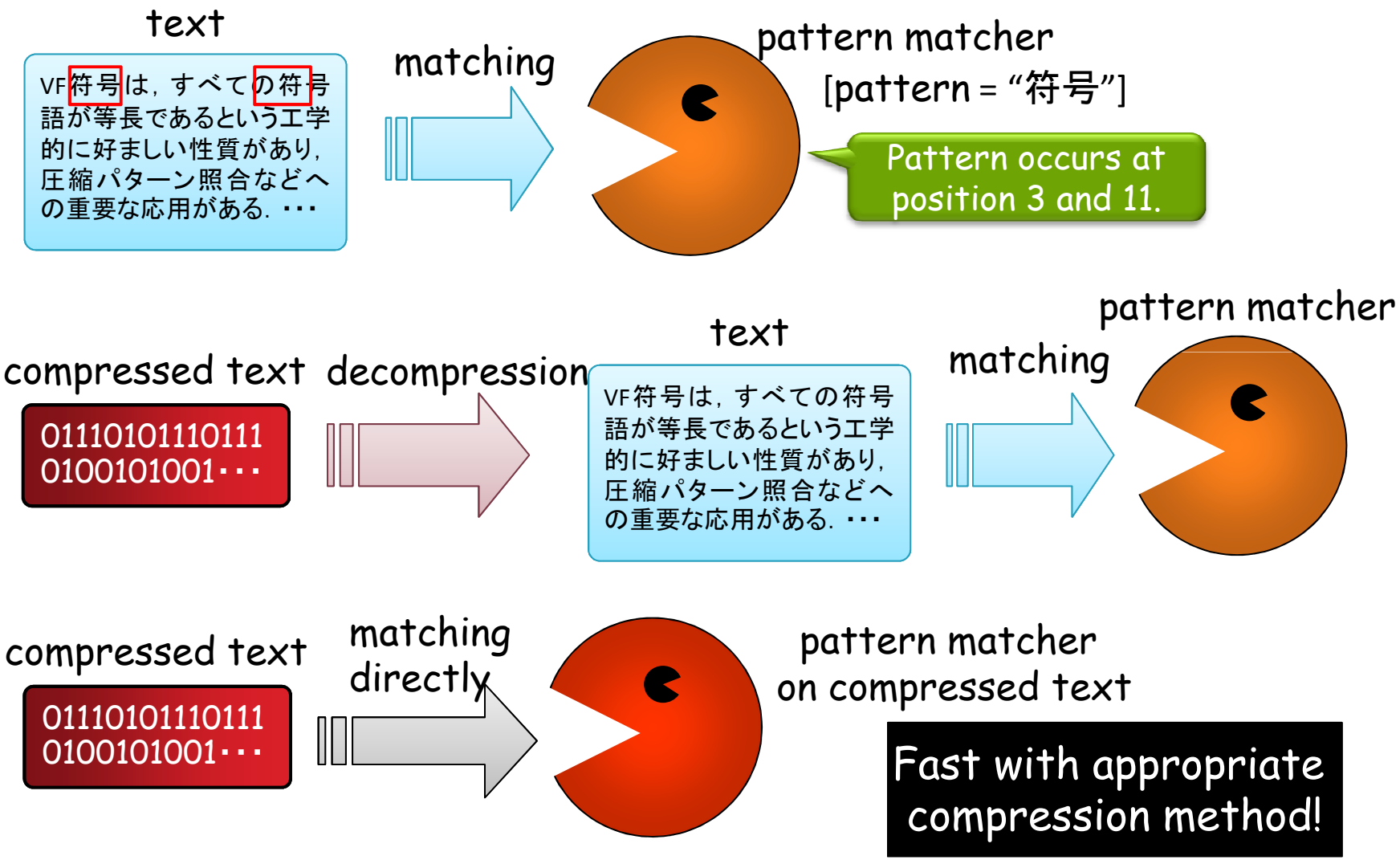
VF符号と算術符号の多段圧縮アプローチに関する考察

Satoshi Yoshida and Takuya Kida

Graduate School of Information Science and Technology,
Hokkaido University

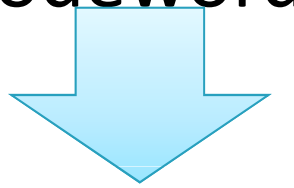
※一部Data Compression Conference 2011にてポスター発表済み

Background: Compressed Pattern Matching



Background: Compression Method Suitable for Compressed Pattern Matching

- The dictionary is static and compact.
- The boundary of codewords is clear.



VF code

		input text	
		Fixed length	Variable length
codeword	Fixed length	**FF code**	**VF code**
	Variable length	**FV code**	**VV code**

Research Goal

is to realize a compression method that enables

high compression ratio and

fast pattern matching.

Research Results until Now

- Improved AIVF code [Yamamoto and Yokoo 2001]
  - reduced the size of dictionary by 80%.
  - Presented in IEEE DCC2010 (Yoshida, Kida)
- Improved compression ratio by training
  - improved compression ratio (same as gzip)
  - takes much time (100 times slow as gzip)
  - SPIRE2010 (Uemura, Yoshida, Kida, Asai, Okamoto)
- Performance evaluation of STVF [kida 2009] + Range Coder
  - improved compression ratio
  - slow pattern matching
  - Presented in IEEE DCC2011 (Yoshida, Kida)

Main Results of DCC 2011

Performance of a combination of STVF code [Kida 2009] and Range Coder [Martin 1979]

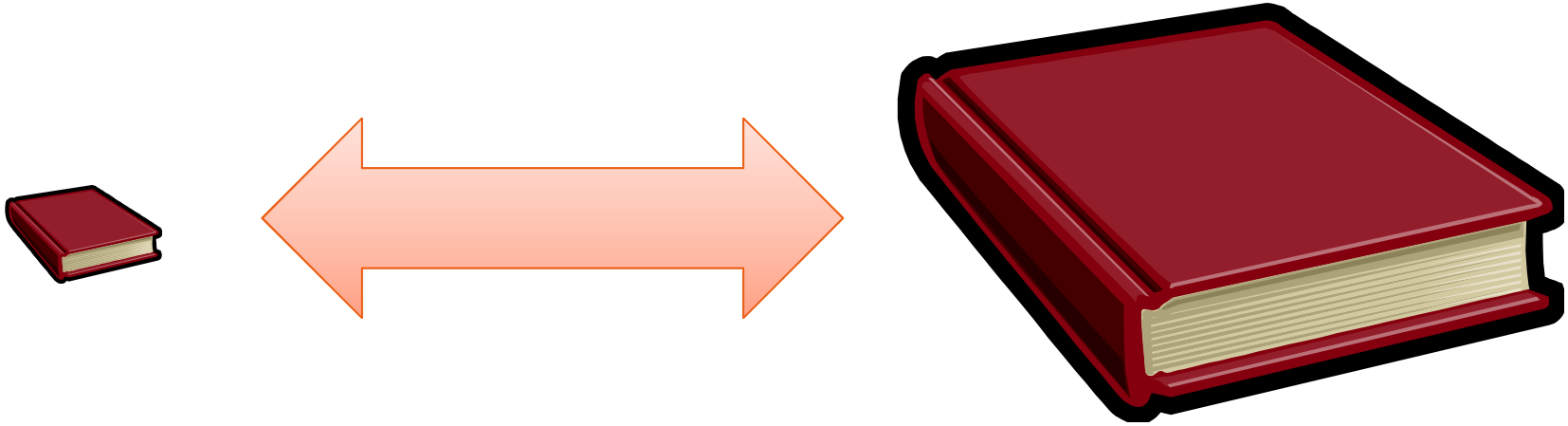
- Compression ratio:

  STVF(12) + range coder slightly improved STVF(16)

- Pattern matching time:

  slower by decompression of Range Coder.

Codeword Length in VF Codes



	short	codeword length	long	
	small	size of dictionary	large	
	low	compression ratio	high	☺
☺	low	cost of construct/hold	high	
☺	low	preparation cost of pattern matching	high	

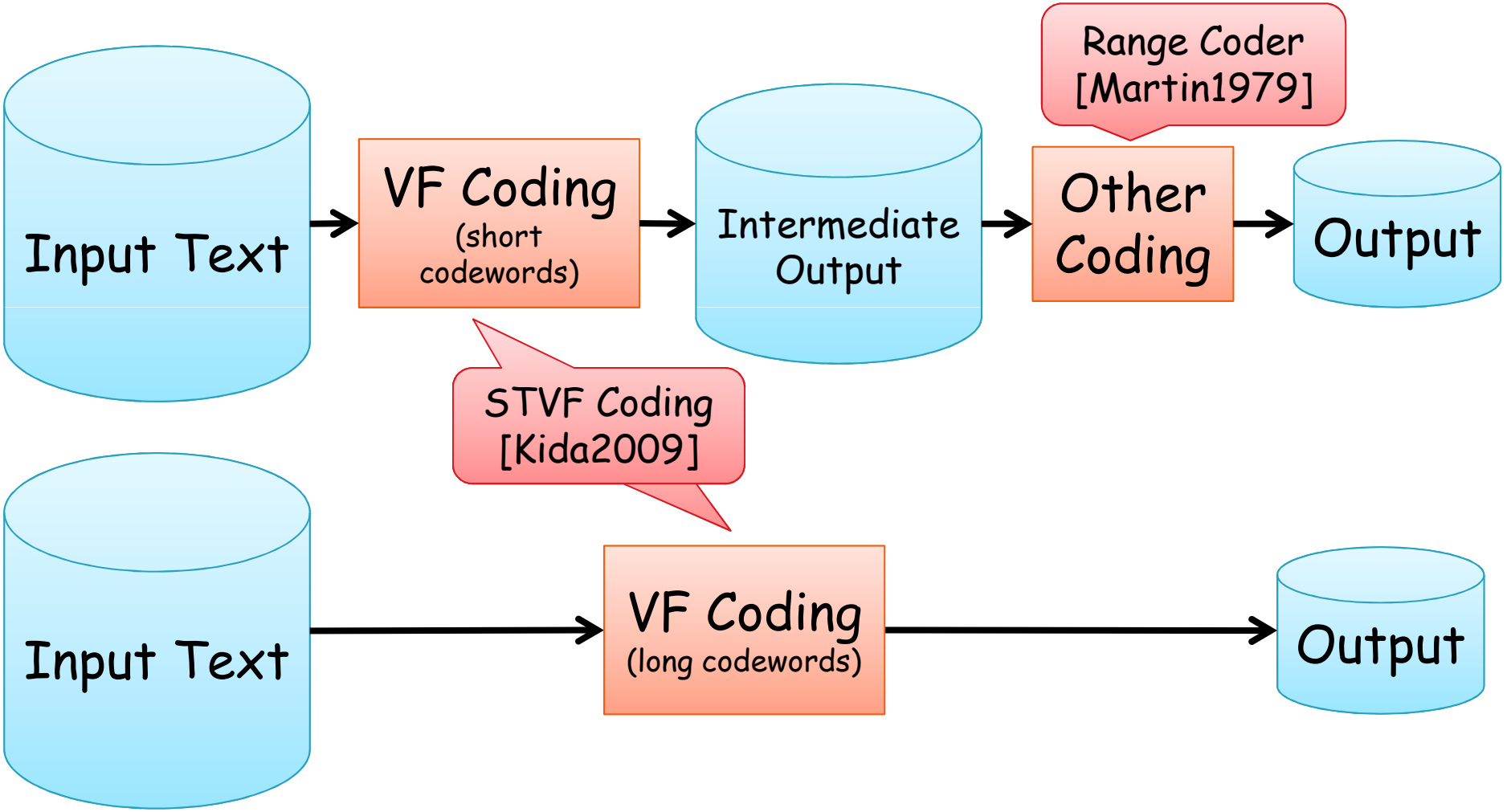
Motivation

Obtaining

a high compression ratio

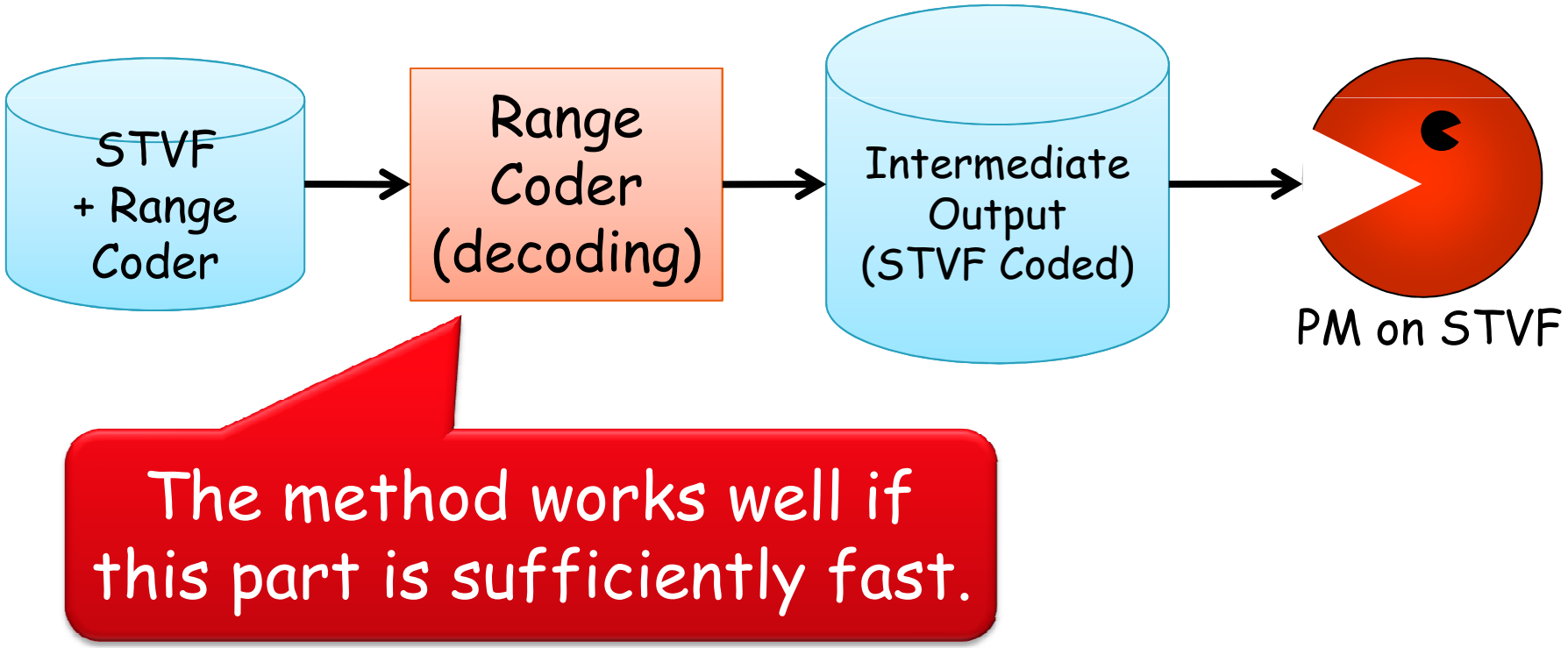
with small dictionary

Two Approaches



Pattern Matching on STVF + Range Coded Text

After decoding compressed text with range coder, we get STVF coded text.

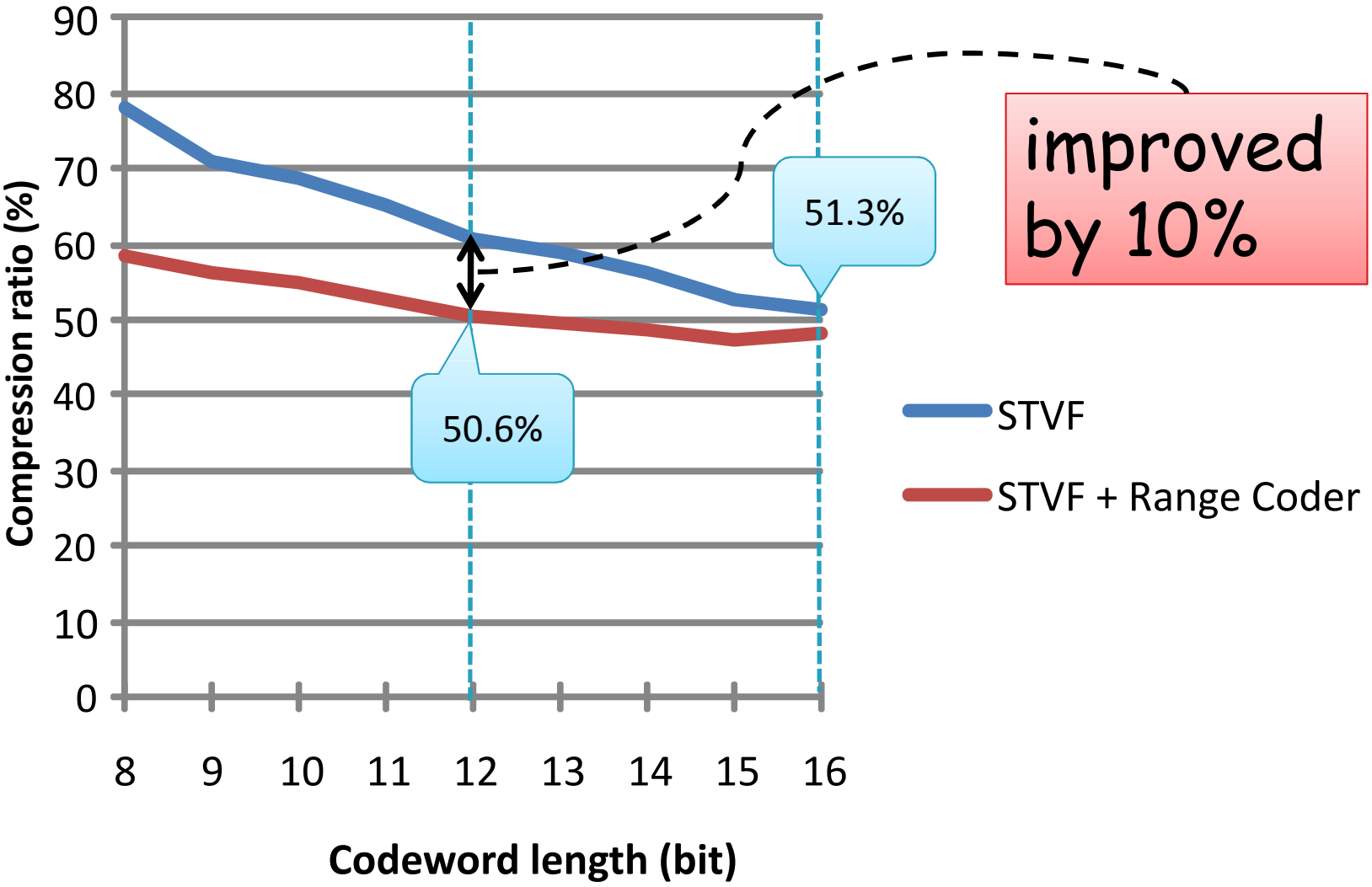


Experiments

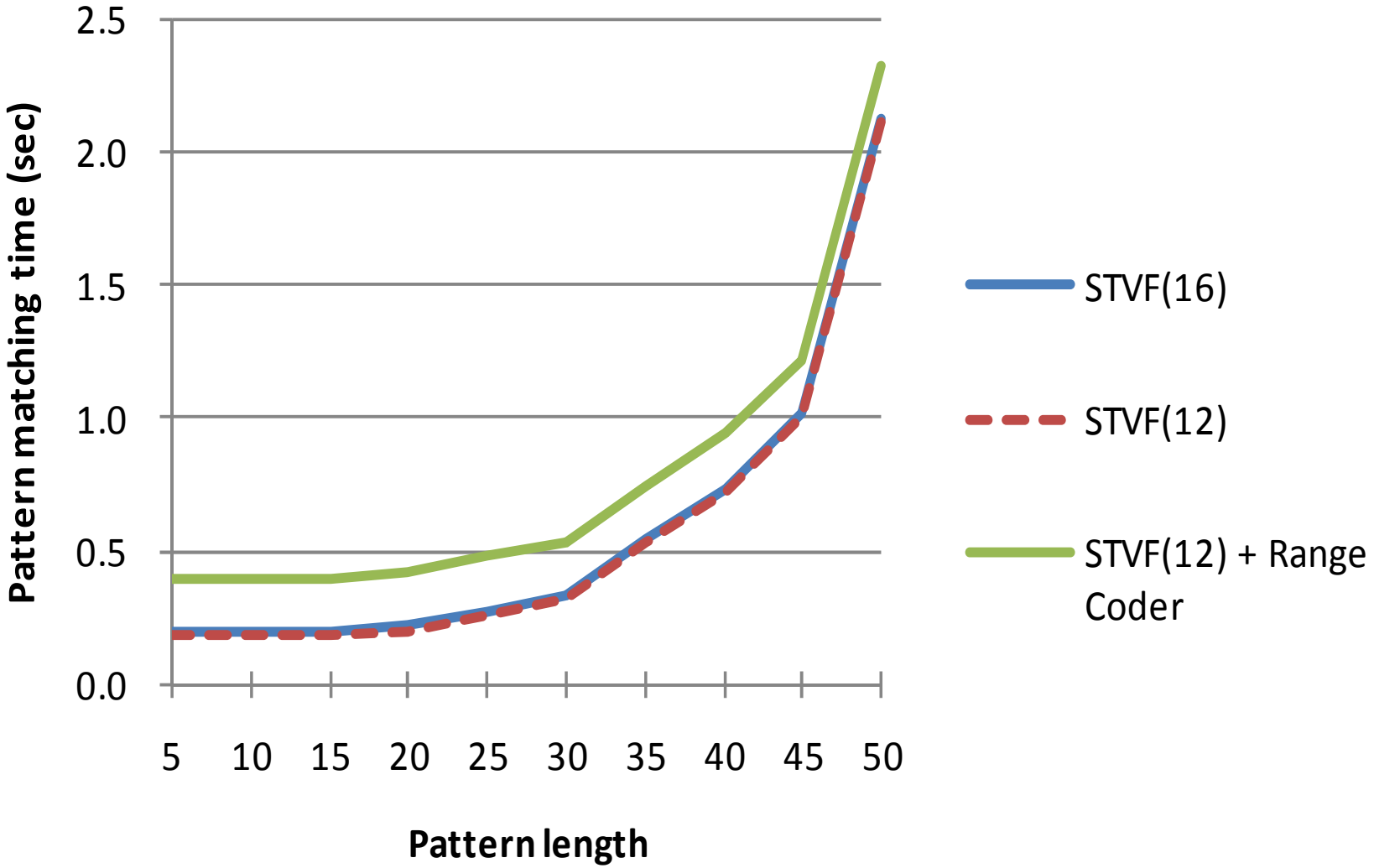
- Compression methods
  - STVF Coding + Range Coder
  - STVF Coding
- Data
  - English Text (brown corpus, 6.8MB, $|\Sigma|=96$)
- Environments
  - CPU: Intel® Xeon® processor 3.00GHz dual core
  - Memory: 12GB
  - OS: Red Hat Enterprise Linux ES Release 4
- Codeword Length
  - l = 8-16 bits

We compared compression ratios and pattern matching times between the two methods.

Compression Ratios (comp. size / original size)



Pattern Matching Time



Conclusion

- STVF + Range Coder improves compression ratios.
- STVF + Range Coder depreciates pattern matching speed.

Future Work

- Combine with other methods whose decompression speeds are fast such as gzip.
- Implement Boyer Moore type algorithm to improve pattern matching speed when pattern length is long.

現状の問題点と今後の研究課題

- 現状のVF符号は圧縮率が悪い，圧縮率を改良すると圧縮に時間がかかる
  - 高速かつ圧縮率が高い圧縮方法の開発
- パターン照合の前処理に時間がかかる
  - 圧縮パターンマッチの高速化
- STVF符号で使用するSuffix Treeはメモリを消費する
  - 大規模テキストへの対応

研究計画

- 高速かつ圧縮率が高い圧縮方法の開発
  - 文法変換に基づく方法
  - LZ圧縮に基づく方法
  - 貪欲なトレーニング手法
  - AISTVFのVMA化
- 圧縮パターンマッチの高速化
  - 大量なパターンに対するパターンマッチ
  - 並列パターンマッチ
- 大規模テキストへの対応
  - 多重分節木のmixing

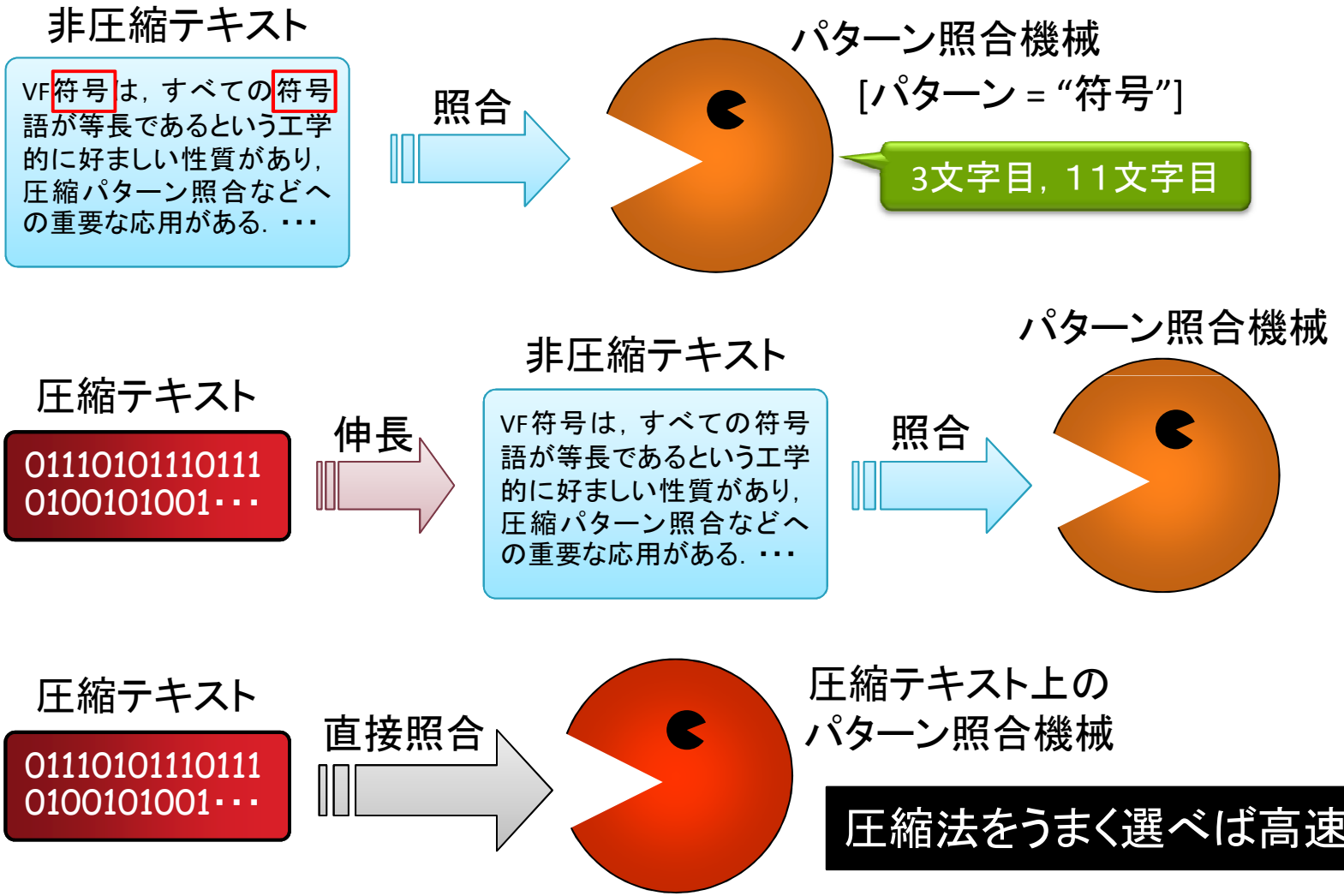
年次計画

	前期	後期
2011年度	※ 貪欲トレーニング	LZ法に基づく手法 文法変換に基づく手法 →DCC2012
2012年度	AISTVFのVMA化	分節木のMixing →DCC2013
2013年度	並列パターンマッチ 大量パターン	博士論文執筆

※Improving Parse Trees for Efficient Variable-to-Fixed Length Codes というタイトルで
IPSJ情報爆発特集号に投稿済み

いまやってること

研究背景：圧縮パターン照合



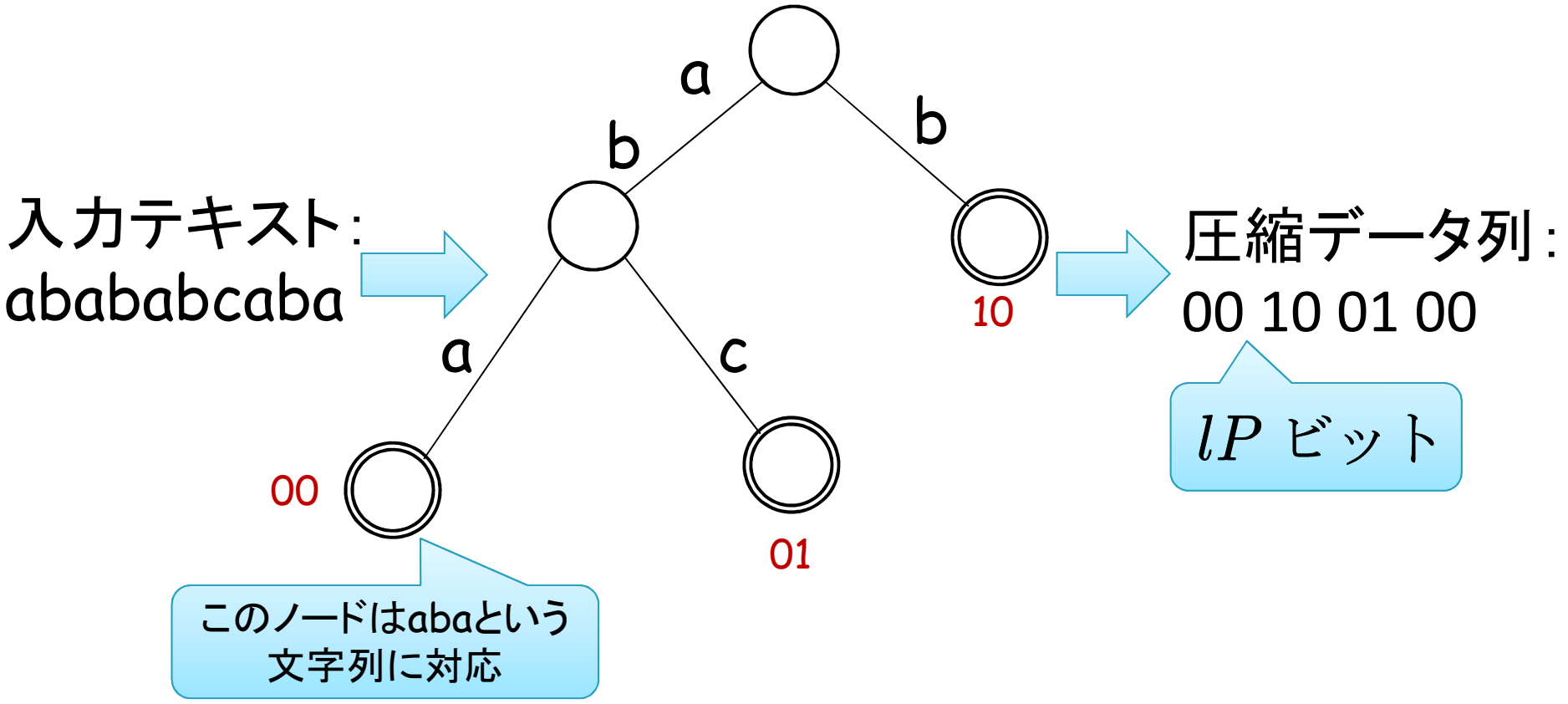
研究背景：圧縮検索に適する圧縮方法

- 辞書が静的でコンパクトである
  - 動的に辞書が変化するとテキストスキャン時に余計な時間がかかる
- 固定長符号である
  - 符号語が容易に切り分けられるほうがよい

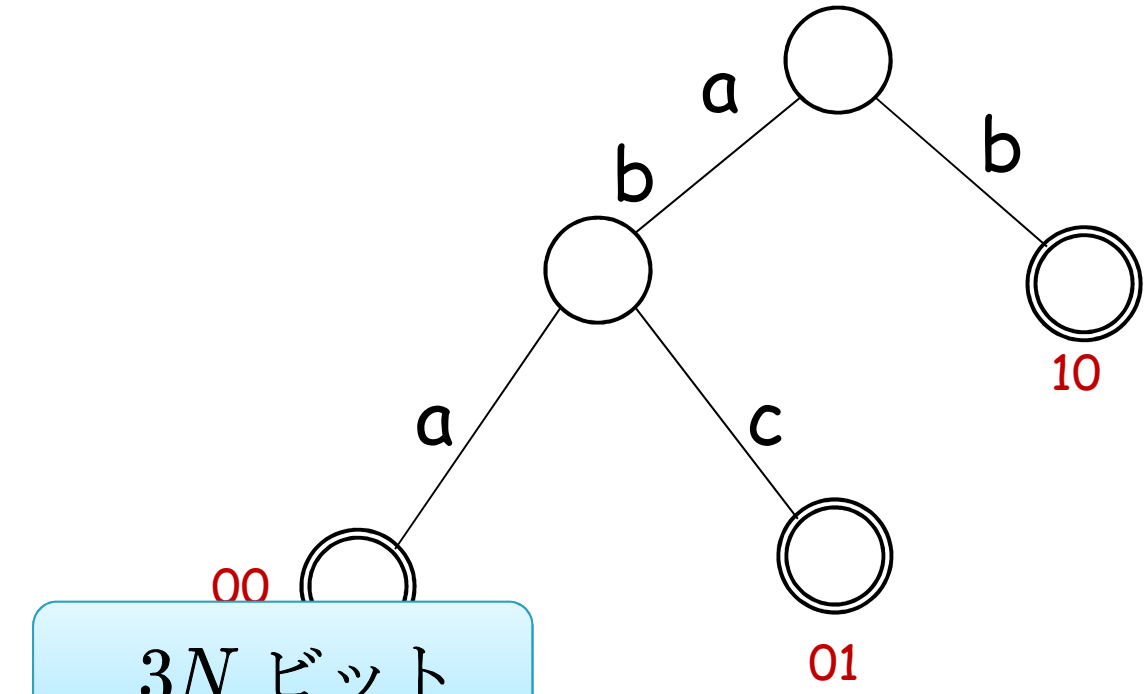
VF符号

		入力テキスト（情報源記号）	
		固定長 (F)	可変長 (V)
圧縮テキスト（符号語）	固定長 (F)	**FF符号** 等長符号	**VF符号** Tunstall符号
	可変長 (V)	**FV符号** Huffman符号	**VV符号** LZ符号

VF符号と分節木


入力テキスト:
abababcaba
a
b
b
a
c
10
00
01
圧縮データ列:
00 10 01 00
lP ビット
このノードはabaという
文字列に対応

分節木のサイズ



$3N$ ビット

木構造: (0(0(1)(1))(1))

ラベル列: ab#a#c#b#

$(N-1+L)\lceil \lg(|\Sigma|+1) \rceil$ ビット

圧縮したデータの総合計

木のサイズ+圧縮データ列の長さ=

$$lP + 3N + (N - 1 + L)\lceil \lg(|\Sigma| + 1) \rceil$$

→最小化

NP完全？