北海道大学
Hokkaido University

情報知識ネットワーク特論
「情報検索とパターン照合」

情報科学研究科 コンピュータサイエンス専攻
情報知識ネットワーク研究室
喜田拓也

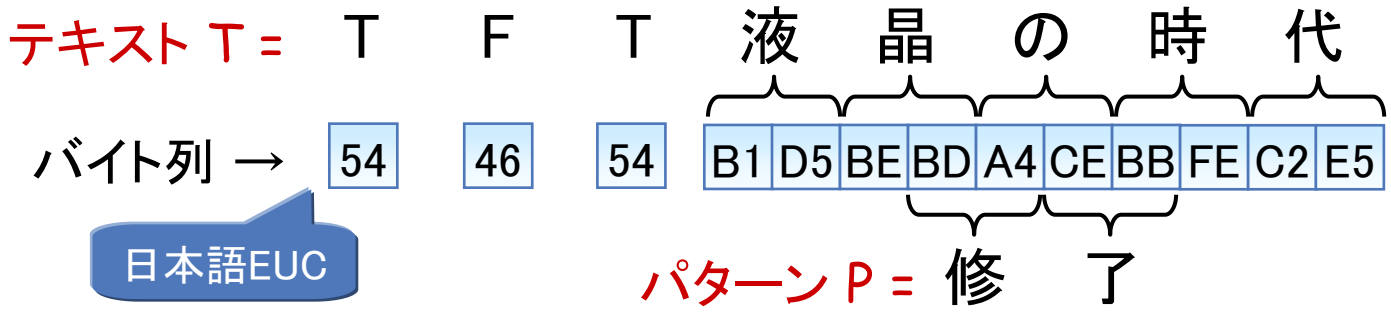
第7回（最終回）
パターン照合技術の今後

多バイトコードへの対応方法
知的なパターン照合を目指して：
1. XMLに対する照合アルゴリズム
2. Arc注釈付きテキストに対する照合
3. 分類階層を考慮したパターン照合

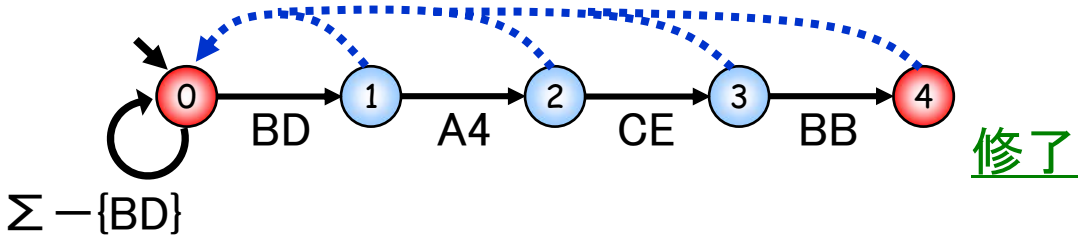
付録：Randomizedアルゴリズム

多バイトコード(日本語文書)への対応方法

- 符号語の同期問題
  - 日本語テキストをASCII単位(バイト単位)で照合すると誤検出が生じる
  - Huffmanコードと同様に、文字の区切りを判別する必要がある



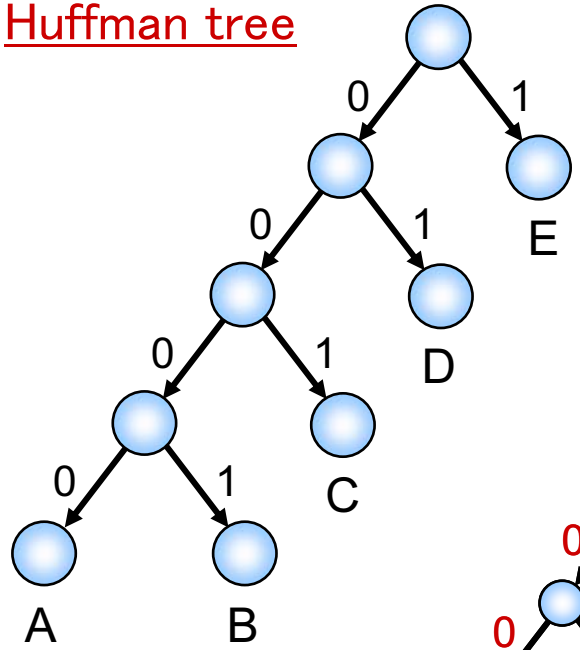
パターンP="BD A4 CE BB"(修了)を照合するAC照合機械



復習：Huffman符号化テキストの場合の解決法

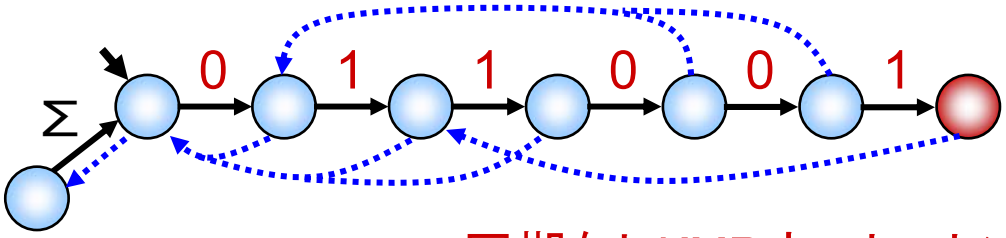
M. Miyazaki, S. Fukamachi, M. Takeda: Speeding up the pattern matching machine for compressed texts (in Japanese), Trans. IPSJ, Vol. 39, No. 9, pp.2638-2648, 1998.

Huffman tree

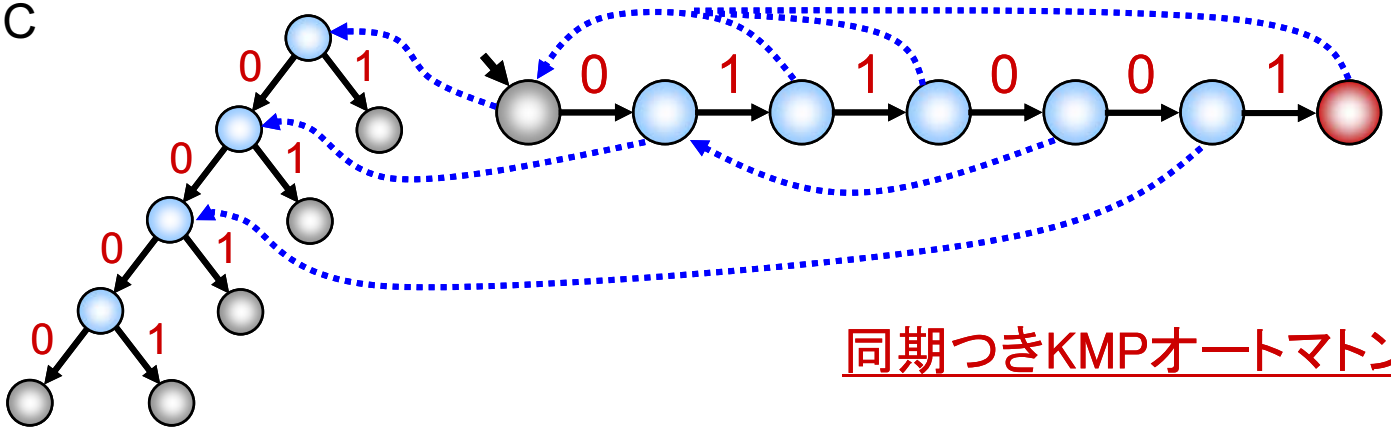


パターン P = DEC

Huffman符号化した
パターン E(P) = 011001



同期なしKMPオートマトン



同期つきKMPオートマトン

テキストT = ABECA・・・　Huffman符号化テキストE(T) = 0000000110010000・・・

同期付きオートマトンによる多バイト文字対応

M. Takeda, et al.: Processing Text Files as Is: Pattern Matching over Compressed Texts, Multi-Byte Character Texts, and Semi-Structured Texts, Proc. of SPIRE2002, LNCS2476, pp.170-186, 2002.

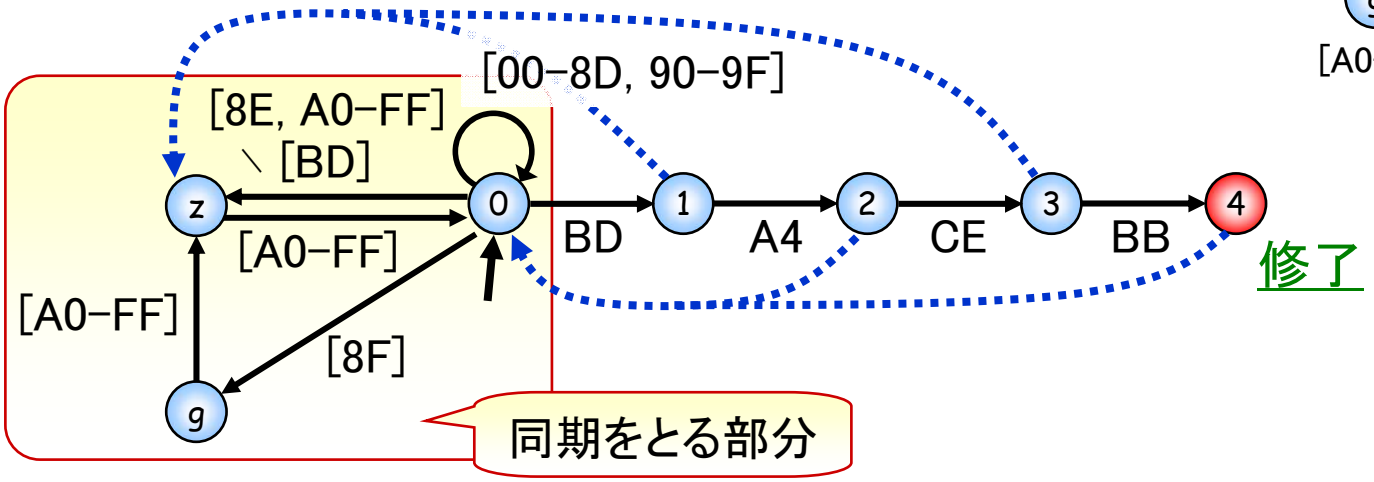
テキスト T = T F T 液 晶 の 時 代

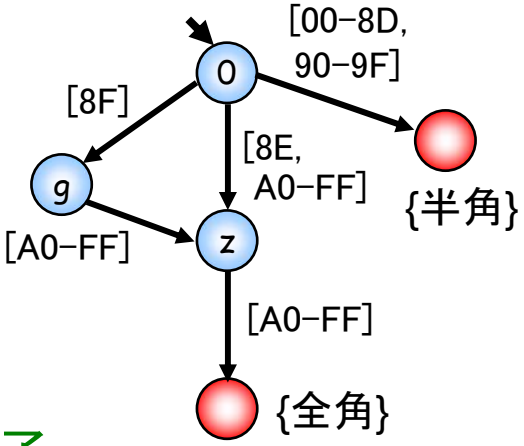
バイト列 → 54 46 54 B1 D5 BE BD A4 CE BB FE C2 E5

日本語EUC

パターン P = 修 了

（EUCエンコードの）パターンP="修了" を
正しく照合する同期付きAC照合機械





EUCコードを受理する
コード・オートマトン

復習：ビットパラレル手法のアイデア

パタン P: a b a b b

テキスト T: a b a b a b b a

1 →	0	1	0	1	0	1	0	0	1
2 →	0	0	1	0	1	0	1	0	0
3 →	0	0	0	1	0	1	0	0	0
4 →	0	0	0	0	1	0	1	0	0
5 →	0	0	0	0	0	0	0	1	0

Mask table M

	a	b
a	1	0
b	0	1
a	1	0
b	0	1
b	0	1

0	1	1	1
1	0	0	0
0	1 &	1	1
0	0	0	0
0	0	0	0

$$R_i = (R_{i-1} \ll 1 \mid 1) \ \& \ M(T[i])$$

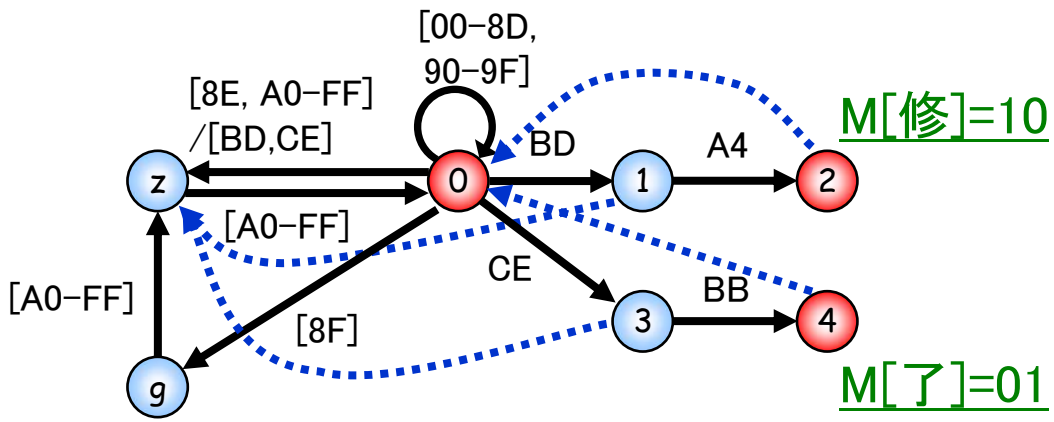
O(1)時間で計算可能

※つまり、マスクビット列Mと「&」をとることで、正しい遷移だけを残している！

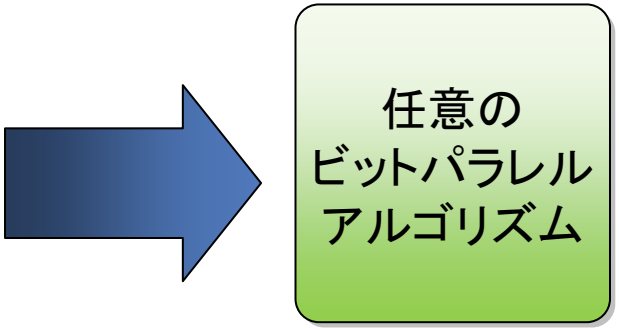
ビットパラレル手法の多バイト文字対応

Heikki Hyyrö, Jun Takaba, Ayumi Shinohara, and Masayuki Takeda: On Bit-Parallel Processing of Multi-byte Strings, *Proc. of Asia Information Retrieval Symposium*, pp.190-196, 2004.

- アイデア
  - 文字コードの境界を判別でき、かつパターン中に含まれる文字を認識する照合機械（コード・オートマトン）を構築する
  - テキストをバイト単位で読み出しつつコード・オートマトンを走らせ、パターン中の文字が見つかったら対応するMaskビット列を出力する
  - T[i]の代わりにコード・オートマトンの出力M(T[i])を入力とみなして、ビットパラレルアルゴリズムを動作させる



EUCの境界を判別し、
「修」と「了」を認識するコード・オートマトン



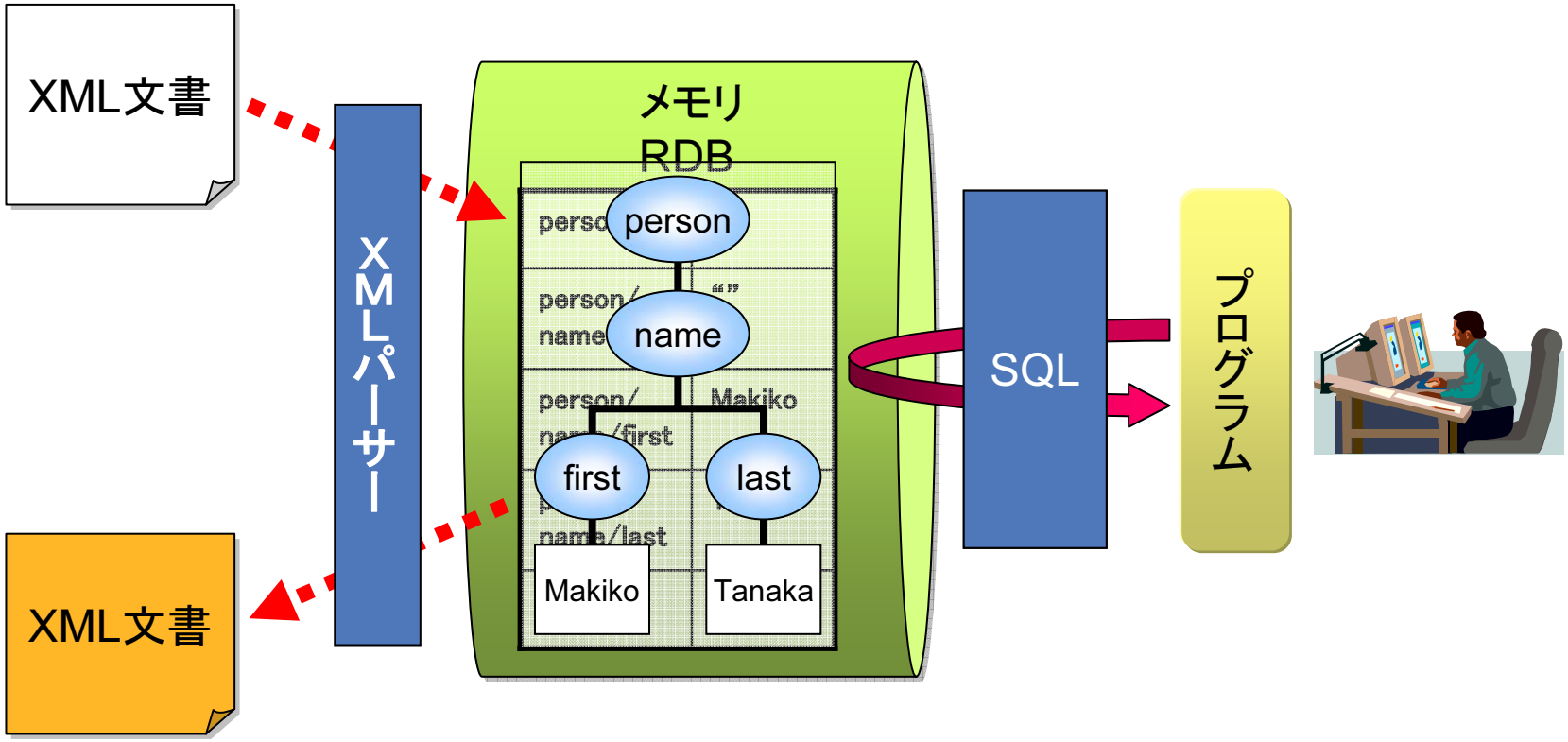
$R_i = (R_{i-1} << 1 \mid 1) \,\&\, M(T[i])$

知的なパターン照合を目指して

- これまで …
  - テキスト = 単なる記号の列
    （テキストに関する背景知識や文の意味などは無視！）
  - Fast! Fast! Fast!

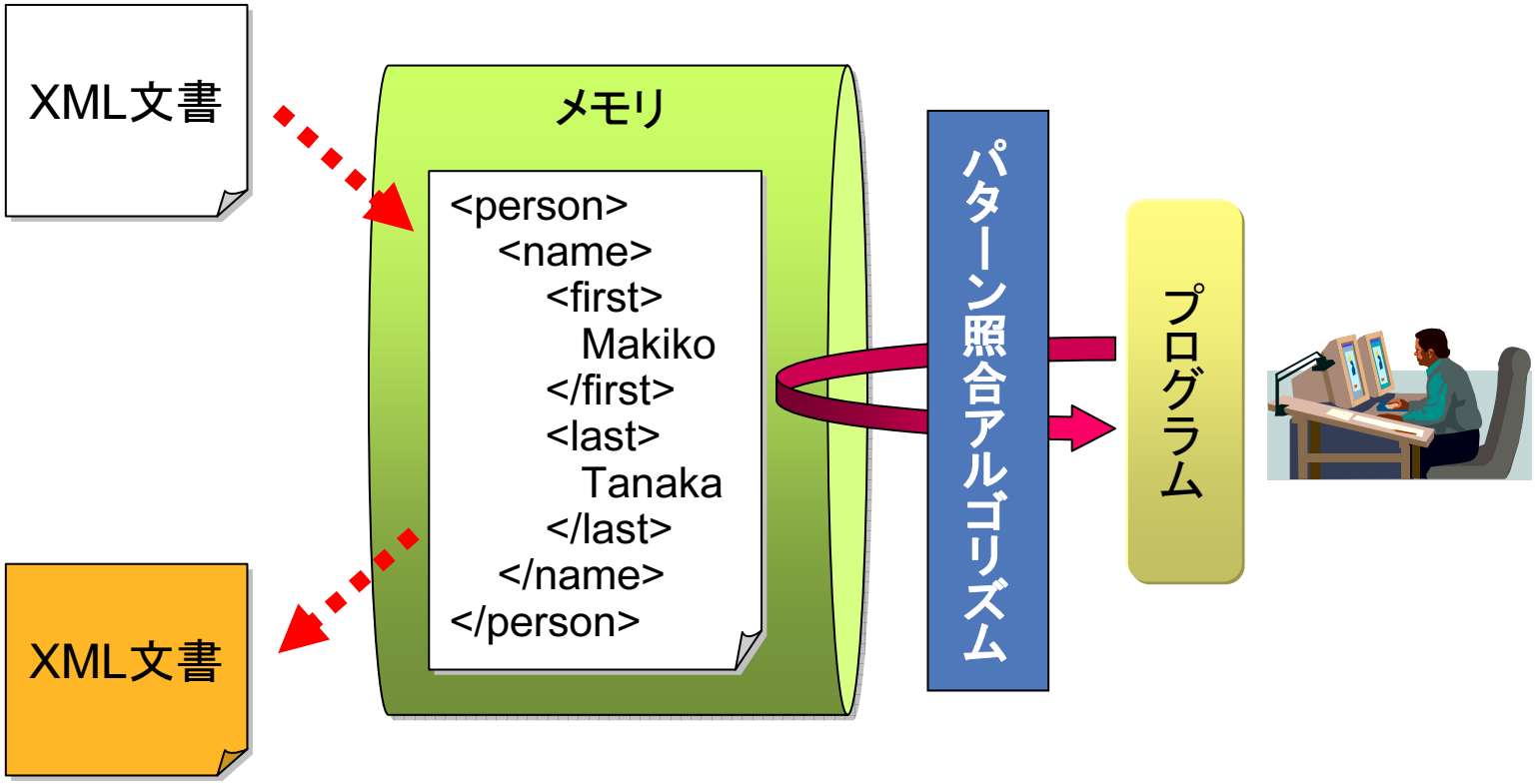
- これから …
  - テキスト = 意味や構造を持った文のつらなり
  - より知的な（できれば高速な！）パターン照合が求められる
    - テキストの構造を考慮した照合
      - XMLテキストに対するパターン照合
      - Arc注釈付きテキストに対するパターン照合
      - etc…
    - テキストの意味を考慮した照合（オントロジーデータとの連携）
      - 分類階層を考慮したパターン照合
      - Thesaurus, Inductive rules, etc…

XMLテキストに対する照合：既存の手法

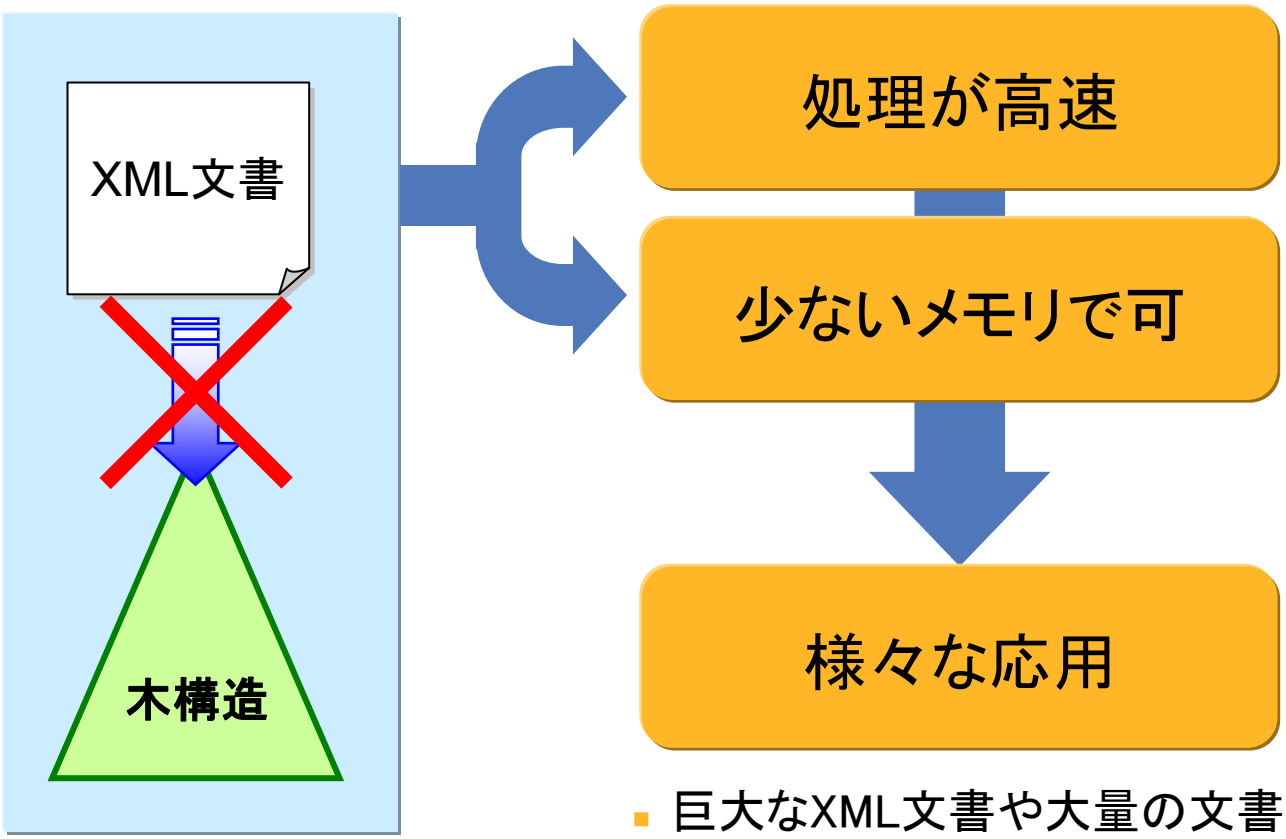


XMLテキストに対する照合：我々のアプローチ

M. Takeda, et al.: Processing Text Files as Is: Pattern Matching over Compressed Texts, Multi-Byte Character Texts, and Semi-Structured Texts, *Proc. of SPIRE2002*, LNCS2476, pp.170-186, 2002.



パターン照合による手法の利点



- 巨大なXML文書や大量の文書群を一括に処理
- 複数の質問を同時に処理

単純なパターン照合での問題点

- タグ名の一部分とマッチしてしまう可能性がある

パターンΠ = {other, mother}

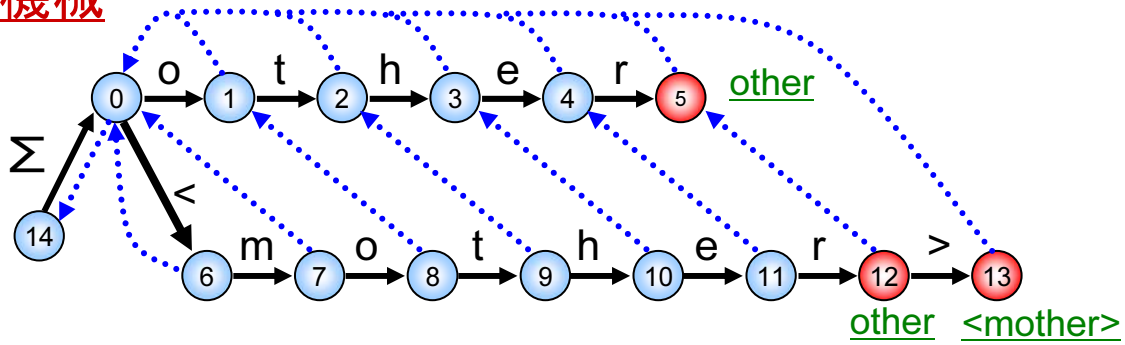
```
<body>
  <h1>あのTVCM</h1>
  <p>
    <mother>
      mother
    </mother>
    mを取ったらother、
    <other>
      他人
    </other>
    です.
  </p>
</body>
```

タグの
中？外？

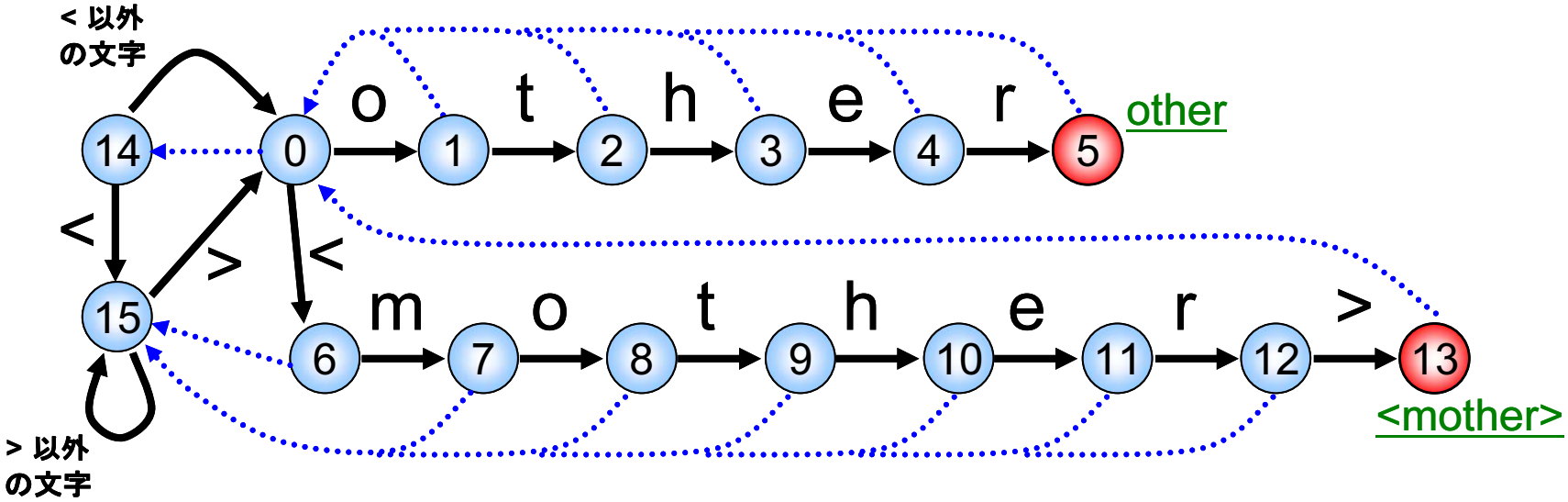
誤った検出

解決策

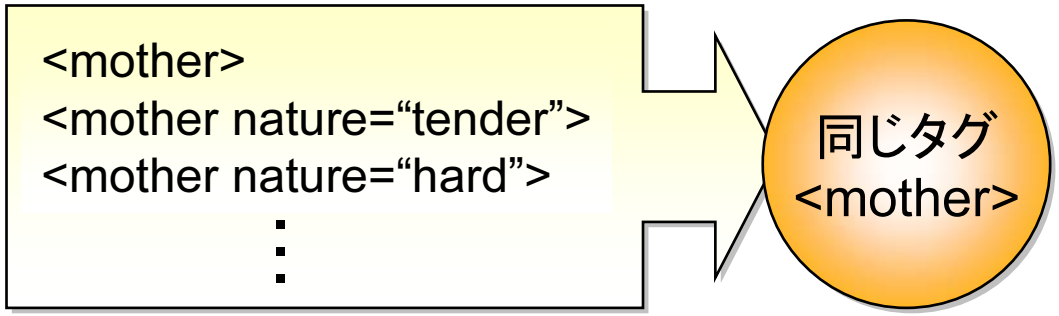
通常のAC照合機械

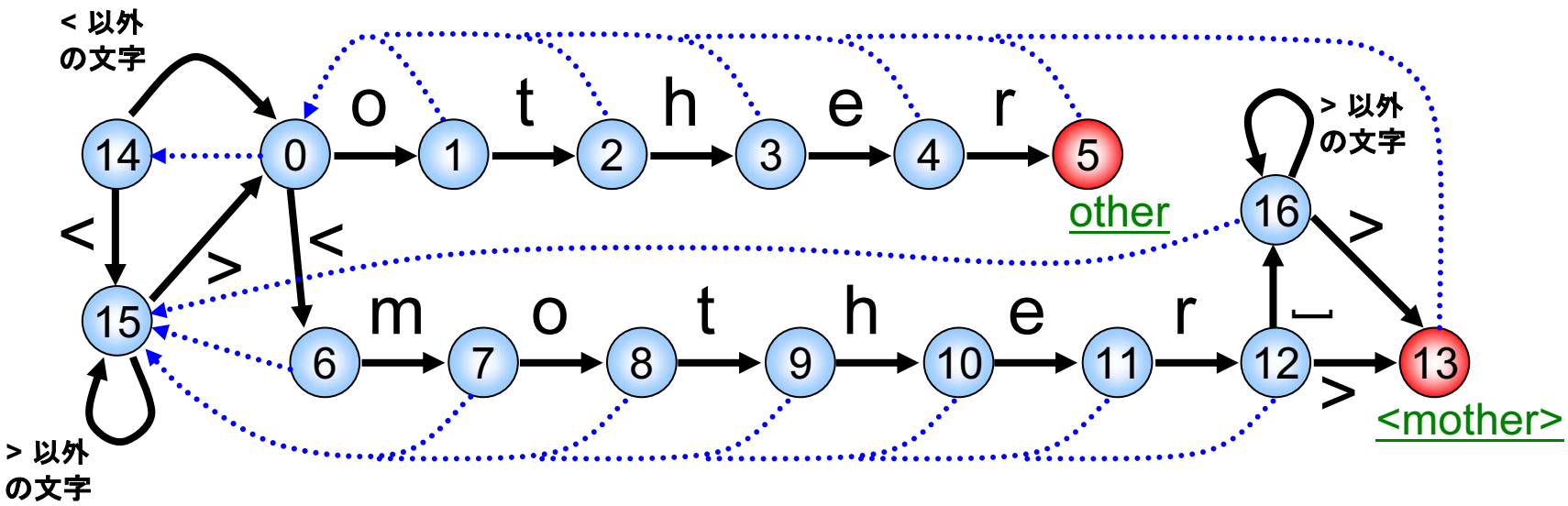


XMLのタグを考慮したAC照合機械



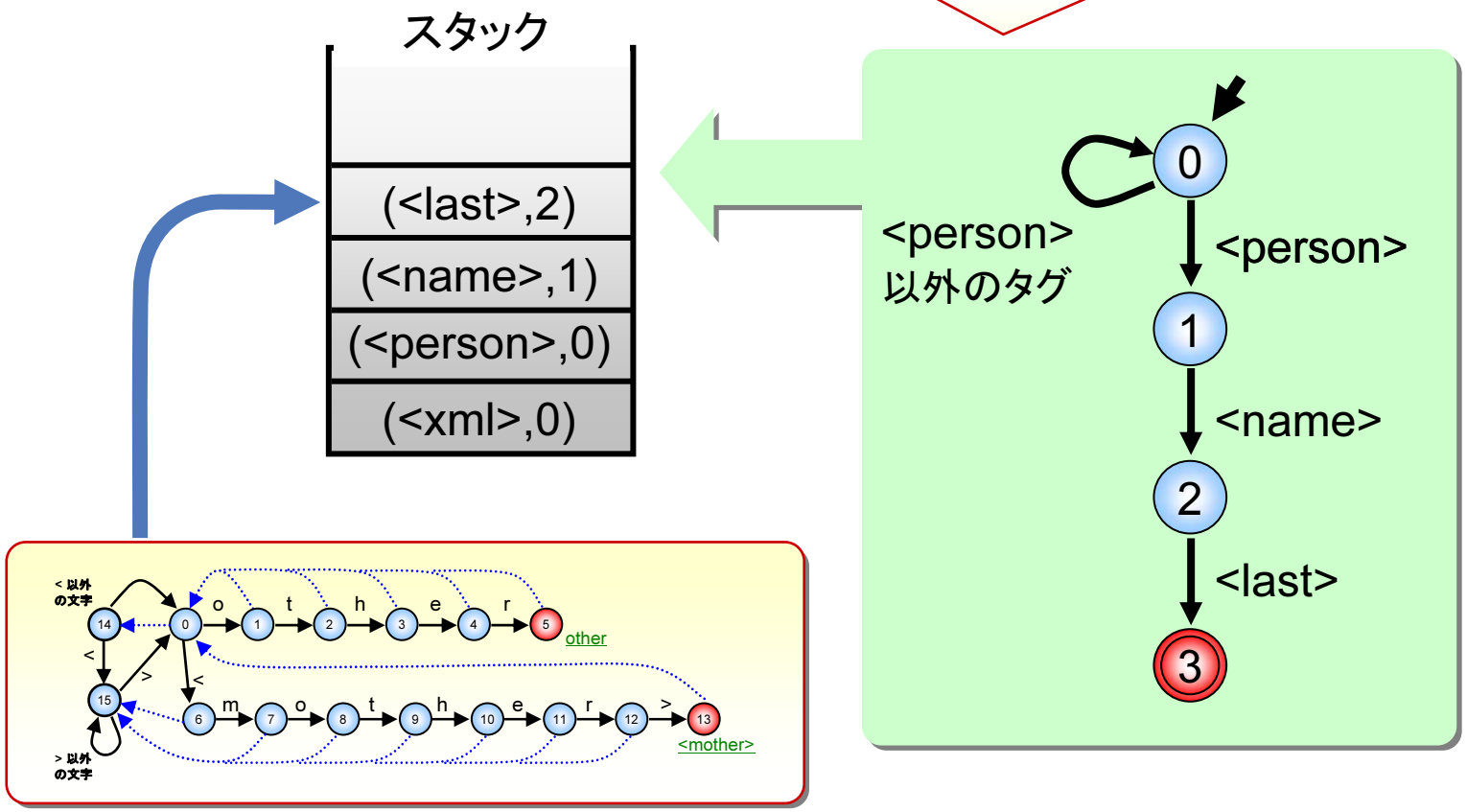
属性の取り扱い


<mother>
<mother nature="tender">
<mother nature="hard">
⋮
同じタグ
<mother>


< 以外
の文字
> 以外
の文字
o
t
h
e
r
m
o
t
h
e
r
_
<
>
<
>
>
0
1
2
3
4
5
other
6
7
8
9
10
11
12
13
<mother>
14
15
16

XMLのパスを考慮した照合

苗字が「田中」の人を探したい！
（Xpathだと，要素 //person/name/last/ が「田中」）



Λ={<person>,</person>,<name>,</name>,<last>,</last>,…}
∏={Tanaka}

処理可能なXPathのサブセット

- 文字列照合による手法の限界
  - 先行ノードの指定はできない！
  - 複雑なフィルタの指定は照合速度を著しく低下させる。

```
LocationPath        ::= '/' RelativeLocationPath
RelativeLocationPath ::= Step
                       | RelativeLocationPath '/' Step
Step                 ::= AxisSpecifier NodeTest
AxisSpecifier ::= AxisName '::'
AxisName        ::= 'attribute'
                       | 'child'
                       | 'descendant'
                       | 'descendant-or-self'
                       | 'following'
                       | 'following-sibling'
                       | 'self'
                       | 'namespace'
```

```
NodeTest ::= QName
                       | NodeType '(' ')'
NodeType ::= 'node'
                       | 'text'
                       | 'comment'
                       | 'processing-instruction'
```

/descendant::cars/child::car/attribute::node()

//cars/car/@*

Sgrepとの速度比較実験

- Sgrep(J. Jaakkola and P. Kilpeläinen)との比較

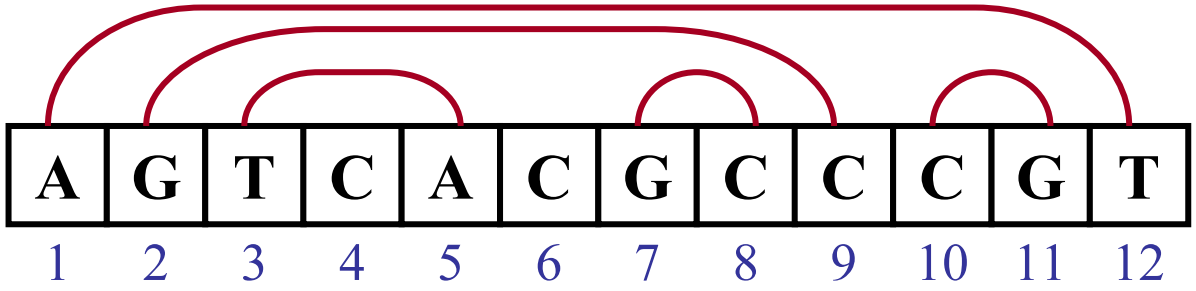
パターン	//text/"summers"	//test//"summers"	/site/regions/africa/item/location/"United_States"
Sgrep	38.44	37.02	51.85
Takeda et al. [2002]	12.40	12.30	12.23

CPU時間（秒）

テキスト：110MB(英文テキスト)
CPU　：Celelon 366MHz
メモリ　：128MB
OS　：Kondara/MNU Linux 2.1 RC2

Arc注釈付きテキストに対するパターン照合

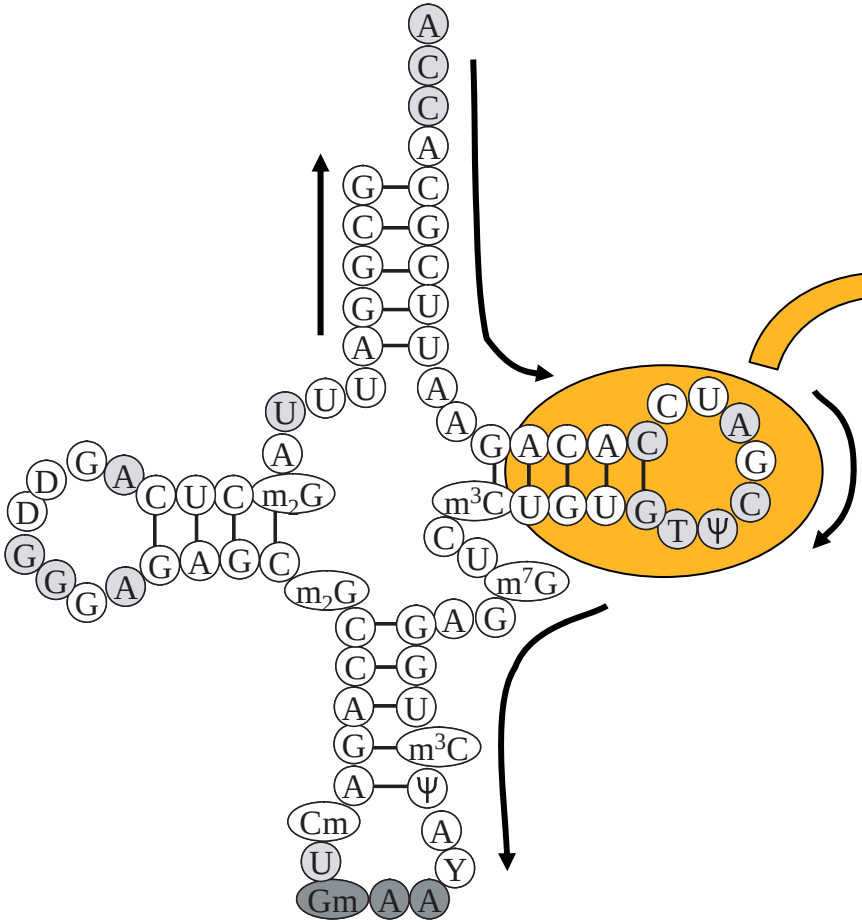
Arc注釈付きテキストの例：



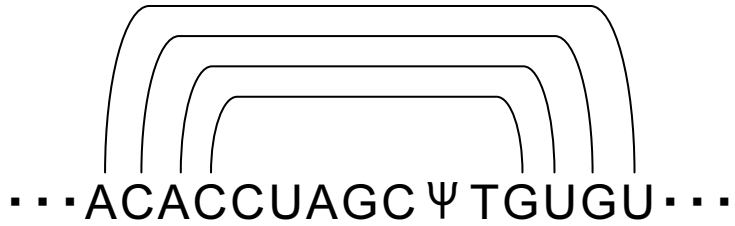
定義：

- **列Sに付随するアーク注釈Aとは、整数{1, 2, . . . , |S|}の二つ組みの集合**
- **各要素 (iL, iR)∈A をアークと呼ぶ**
  - S[iL] と S[iR] をそれぞれ左端点、右端点と呼ぶ
  - 任意のアークについて iL < iR が成り立つと仮定する
  - また、任意のアークどうしは同じ整数を共有しない
  - すなわち任意のアークどうしは同じ端点を共有しない

Arc注釈付きテキストの実例



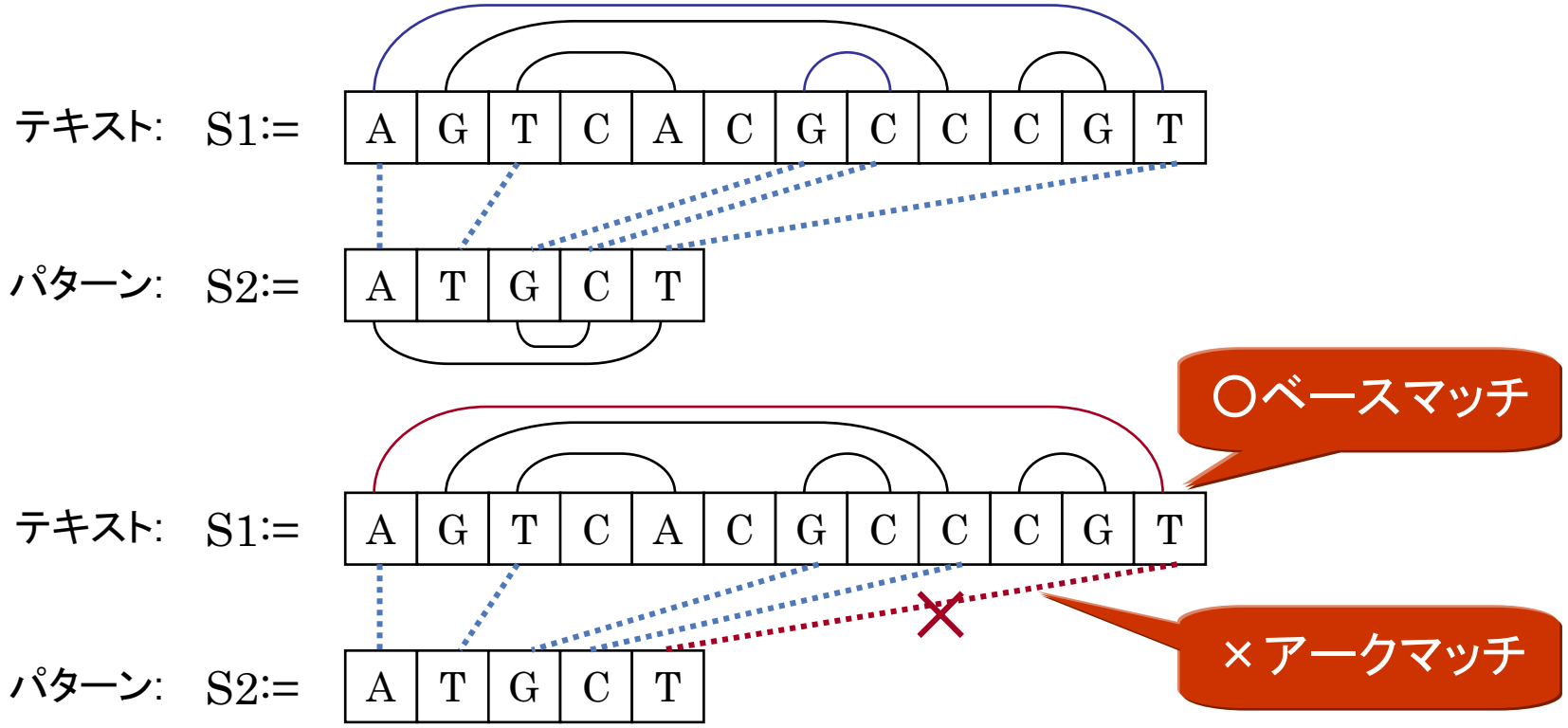
tRNA(tRNAPhe) 二次元構造の例



ネストされたアークをもつ文字列

Arc-preserving subsequence(APS) 問題

- テキストS1 = S1[1 : n] とパターンS2 = S2[1 : m] がそれぞれアーク注釈A1とA2を伴って与えられたとき、APS問題とは以下を満たしているかどうかを答える問題
  - S2がS1の部分列
  - その部分列にアークがある場所にはパターンにも対応するアークがあり、かつ逆も成り立つ

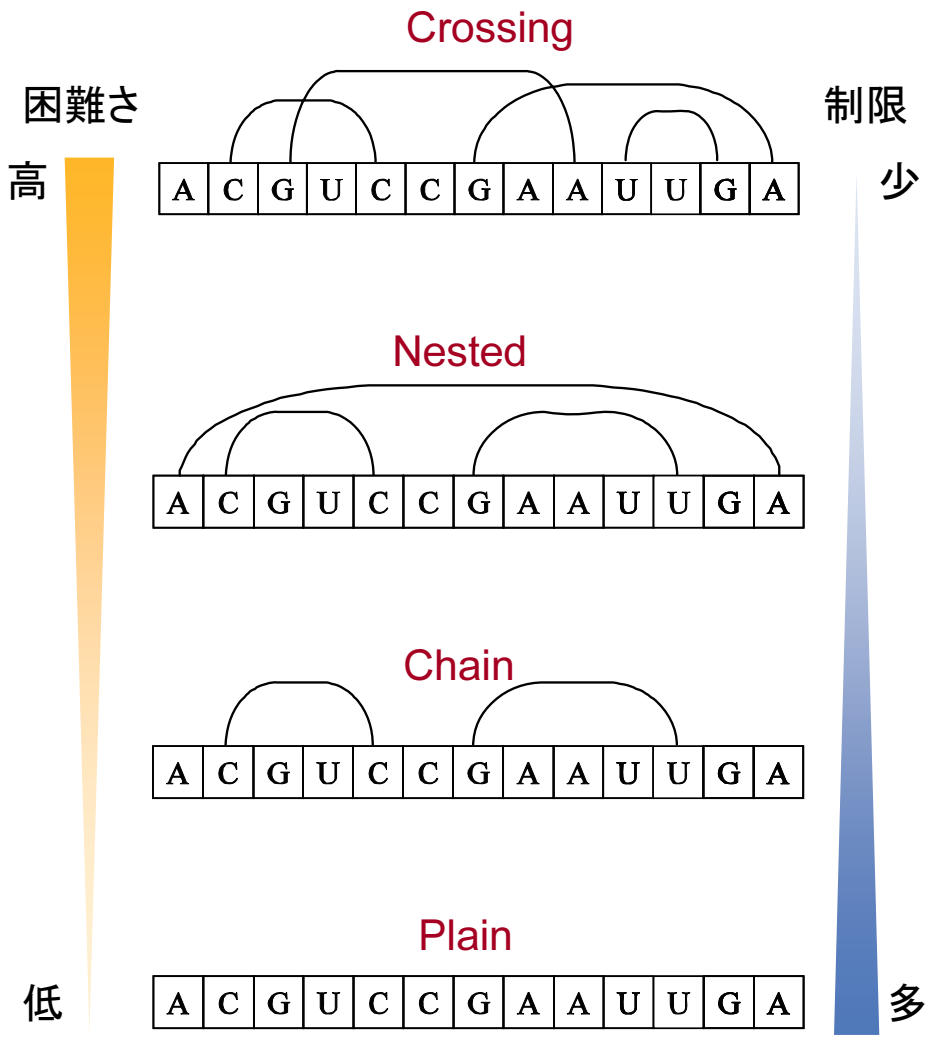


APS(TYPE1, TYPE2)

- **APS問題はアーク注釈の構造によって困難さが変化する**

- **APS(TYPE1, TYPE2)**
  - TYPE1：テキスト側のアークの構造
  - TYPE2：パターン側のアークの構造

- **例： APS(nested, chain)**
  - テキストのアーク構造がnested
  - パターンのアーク構造がchain



喜田2005の結果

Kida: Faster Pattern Matching Algorithm for Arc-Annotated Sequences, *Proc. of Federation over the Web*, LNAI (to appear)

APS問題の先行研究:

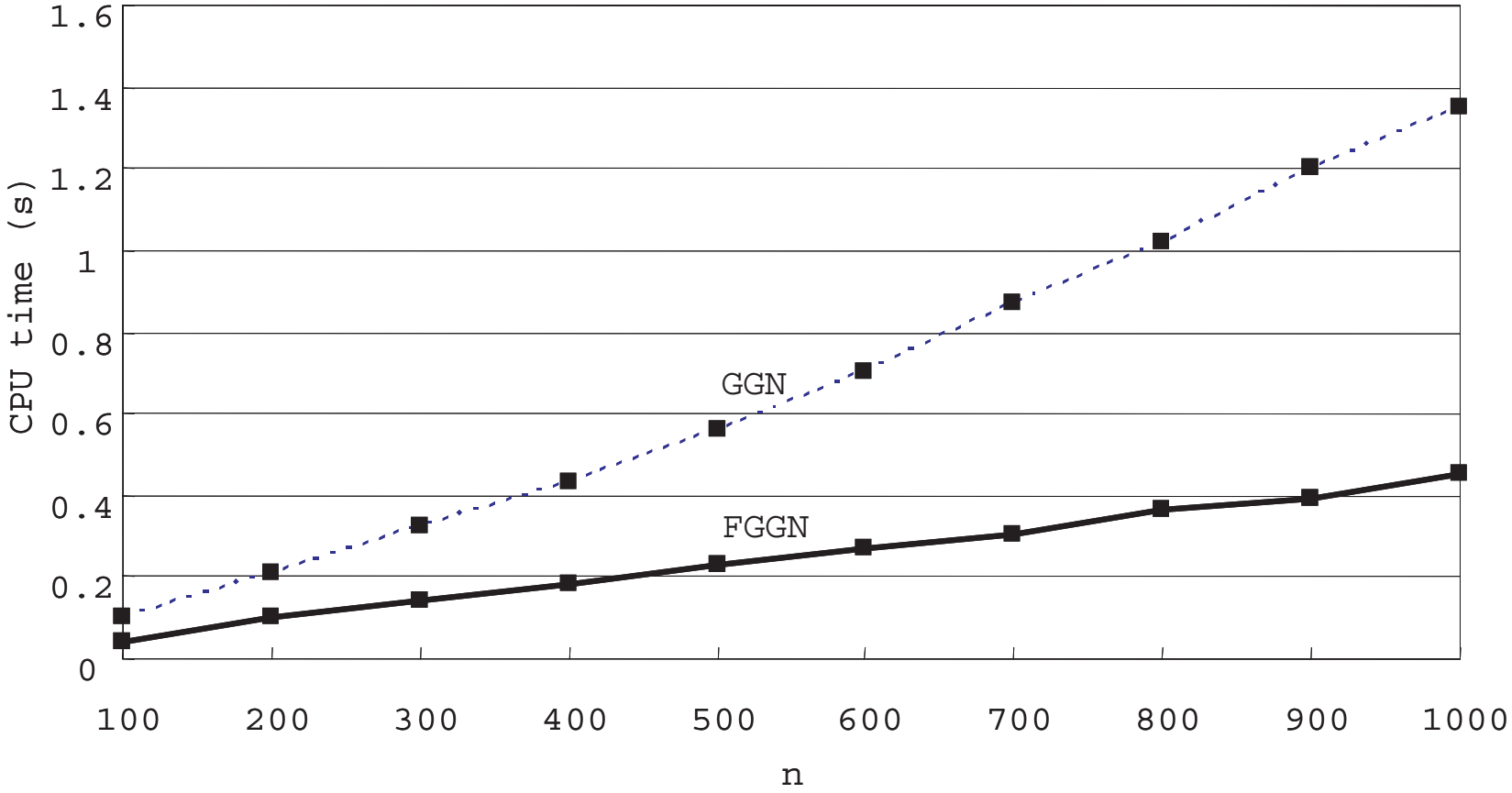
- J. Gramm, J. Guo, and R. Niedermeier. "Pattern matching for arc-annotated sequences." In *Proc. 22nd FSTTCS*, volume 2556 of LNCS, pages 182-193. Springer, 2002.

APS(nested, nested)がO(nm)

喜田[2005]の結果:

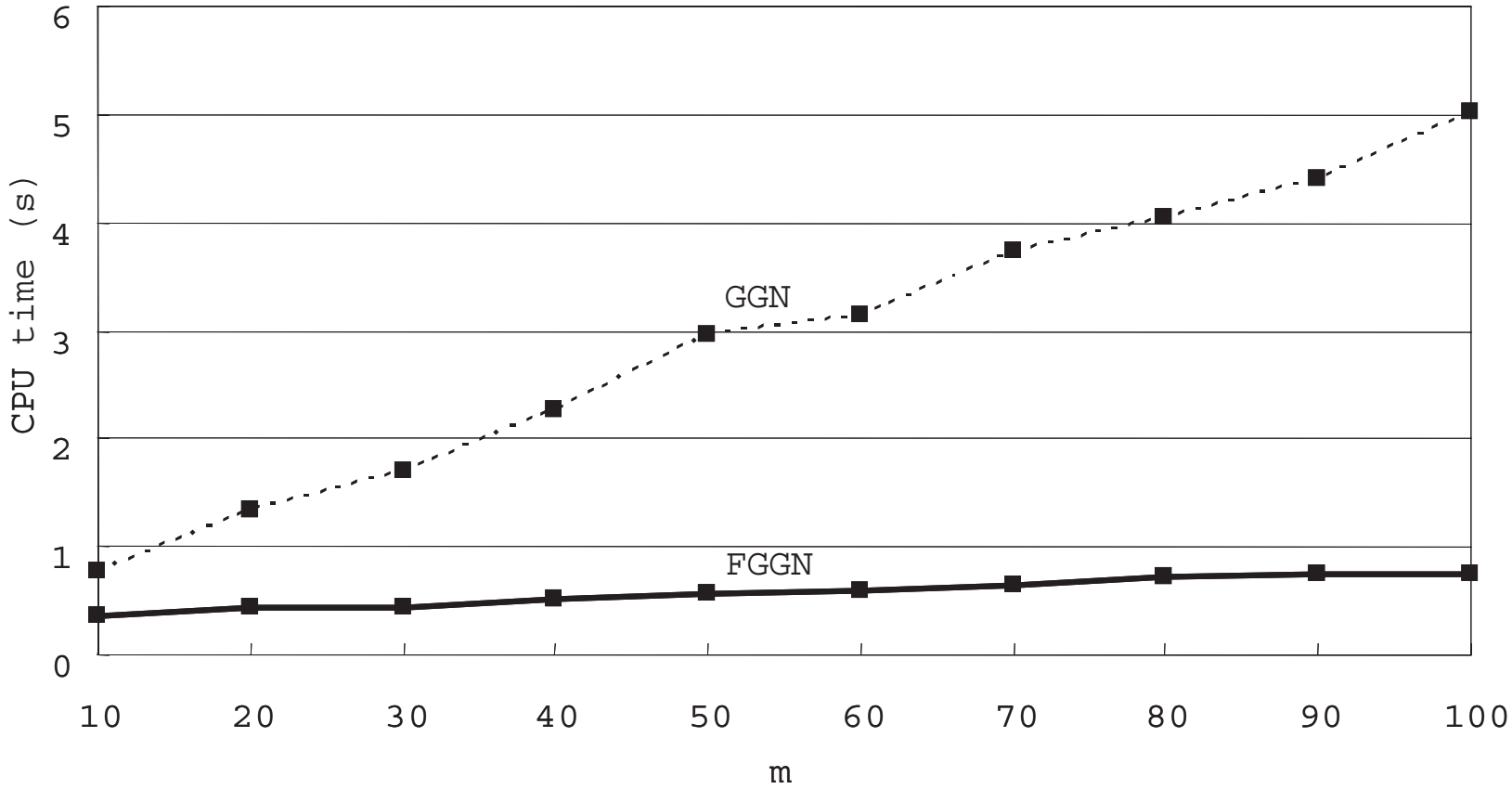
- GGNアルゴリズムに基づいた改良アルゴリズムを提案
  - ただし、最悪時の計算量はGGNアルゴリズムと同様
- Gramm-Guo-Niedermeier (GGN)アルゴリズムを修正
  - 元のGGNアルゴリズムはエラーが含まれている
- 実装と実験を行った
  - 提案アルゴリズムはGGNアルゴリズムより 2～5倍高速

テキスト長 n に対する変化



$|A_1|$=20% of n, m=20, $|A_2|$=4

パターン長 m に対する変化



$|A_2|$=20% of m, n=1000, $|A_1|$=100

ちょっと、ひといき・・・

- ここまでのまとめ
  - 多バイトコードへの対応方法
    - 同期付きオートマトンをAC照合機械へ組み込む
    - Maskビット列を出力するコード・オートマトンをビットパラレル手法に組み合わせる
  - テキストの構造を考慮した照合
    - XMLテキストに対するパターン照合
    - Arc注釈付きテキストに対するパターン照合

水中を翔るキングペンギン（旭山動物園にて '05.8.12）

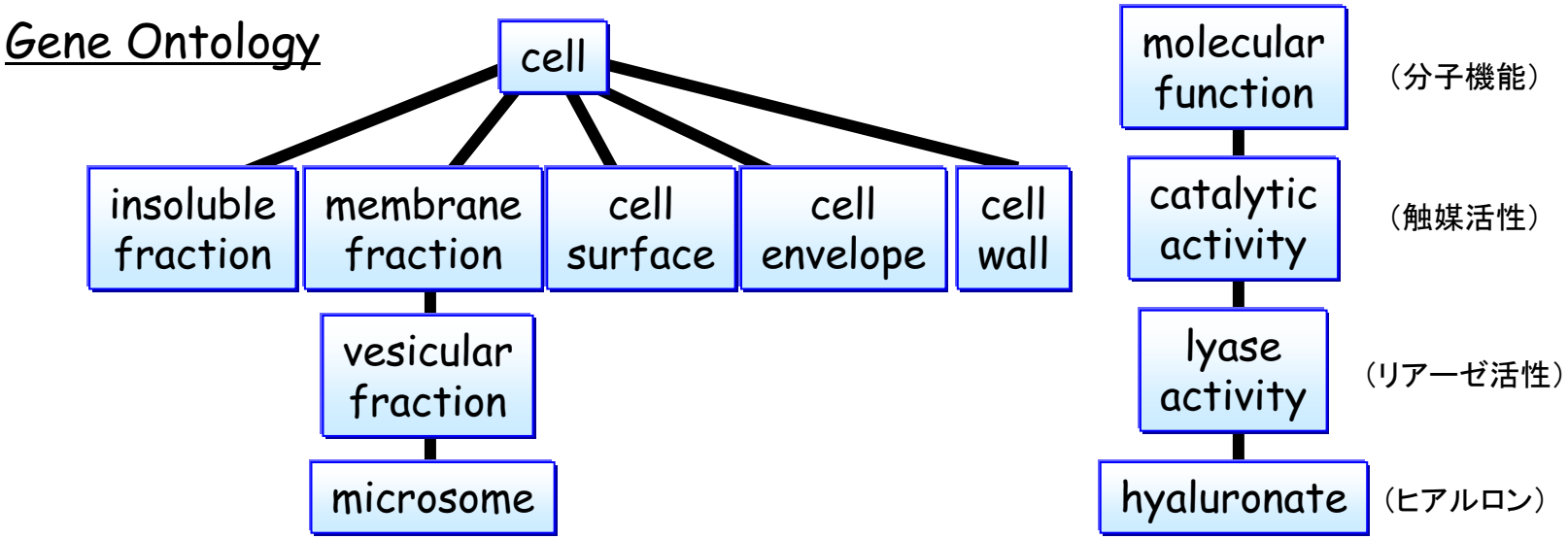
～トリビア～

整数 x と y が長さ m ビットのビット列で表現されているとき条件分岐を行わずに min(x,y) を求める方法

$$S \leftarrow ((x \mid 10^m) - y) \,\&\, 10^m,$$
$$S \leftarrow S - (S \gg m),$$
$$\min(x,y) \leftarrow (\sim S \,\&\, x) \mid (S \,\&\, y)$$

（ただし、この方法だとm+1ビット必要）

分類階層を考慮した照合問題(PMTX)の例



パターンP: (cell) (receptor) (for) (catalytic activity)

テキストT:

Pub:1: Cell. 1990 Jun 29;61(7):1303-13.
Title:CD44 is the principal cell surface receptor for hyaluronate.
Authours:Aruffo A, Stamenkovic I, Melnick M, Underhill CB, Seed B.

Kida&Arimura[2004]の結果

T. Kida and H. Arimura: Pattern Matching with Taxonomic Information, *Proc. of Asia Information Retrieval Symposium* (AIRS2004), pp. 265-268, Oct. 2004.

- 前処理 O(m+mh/w) 時間
- 領域 O(m|Σ|/w)
- テキスト走査 O(mn/w) 時間

m < w のとき、
うまく働く

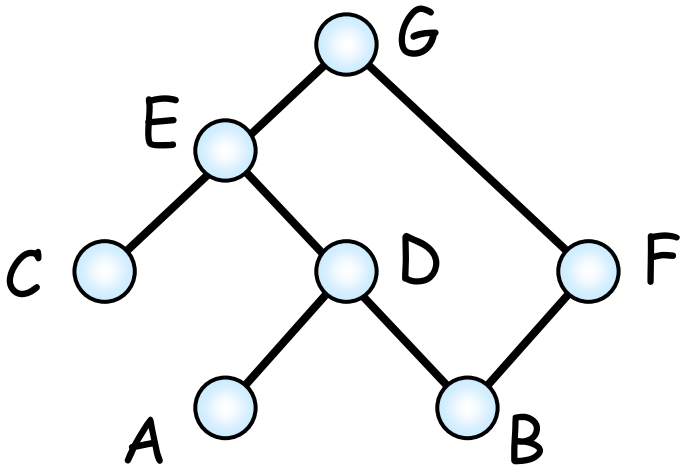
- 前処理 O(m+h) 時間
- 領域 O(|Σ|)
- テキスト走査 O(n) 時間

  - m: パターン P∈Σ* の長さ
  - n: テキスト T∈Σ* の長さ
  - h: 分類階層情報Hの大きさ
  - |Σ|: 概念集合Σの大きさ
  - w: ワード長(現行では 32 or 64)

分類階層情報とソート付きアルファベット

- ソート付きアルファベット（Σ, ≥）
  - Σ: 有限アルファベット（概念の集合）
  - ≥: 半順序関係

(Σ,≥) を表すDAG Hの例



※ 別名、ハッセ図（Hasse diagram）

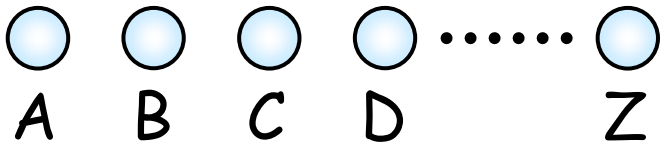
パターンとテキストは
概念の列 P∈Σ*とT∈Σ*
で与えられるものとする

パターンP:= A B E F

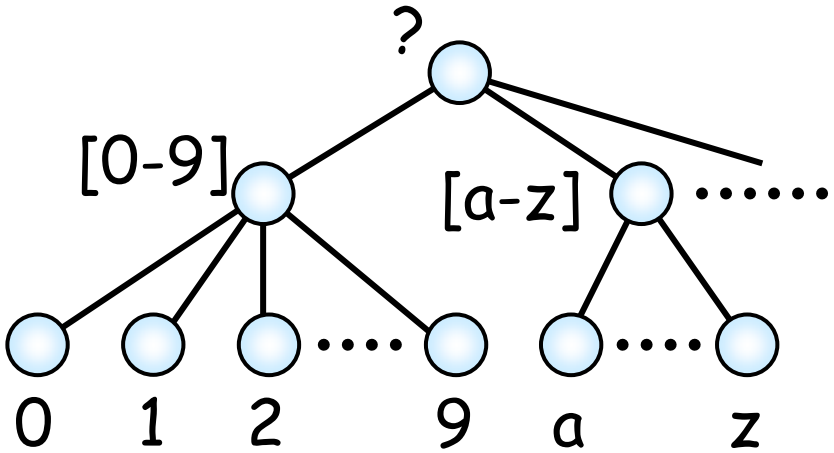
テキストT:= A B C B D F C B

概念Eは文字クラス
[A,B,C,D,E]と振る舞いが同じ

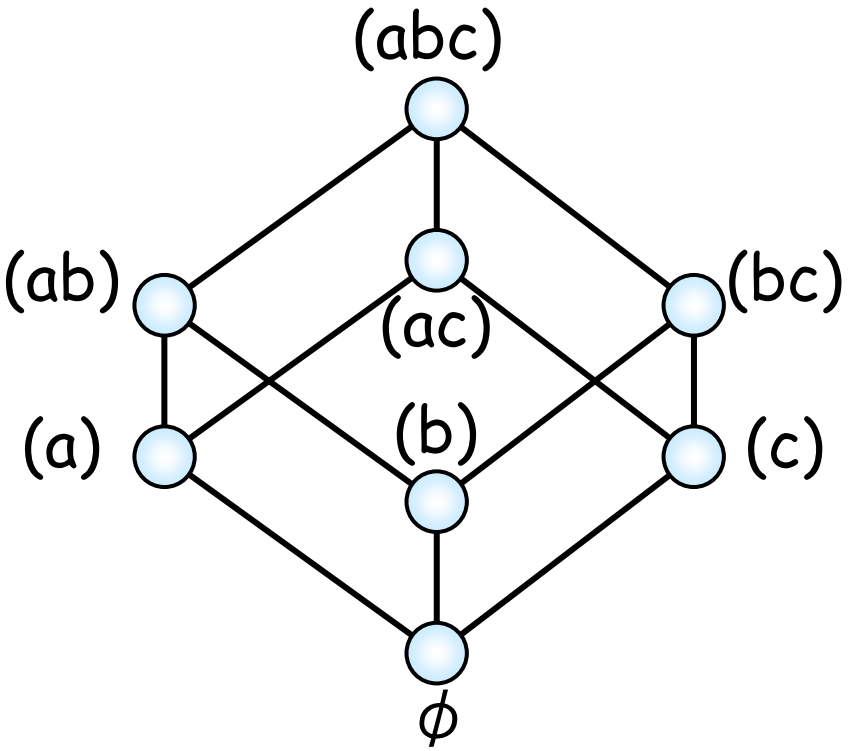
ソート付きアルファベットの例



(1) flat alphabet



(2) class of characters



(3) letter-sets alphabet

Shift-And法のアイデアが使える！？

パターン P: a b [a b] b b

テキスト T: a b a b b b b a

1 →	0	1	0	1	0	0	0	0	1
2 →	0	0	1	0	1	0	0	0	0
3 →	0	0	0	1	0	1	0	0	0
4 →	0	0	0	0	1	0	1	0	0
5 →	0	0	0	0	0	0	0	1	0

Mask table M

	a	b
a	1	0
b	0	1
[ab]	1	1
b	0	1
b	0	1

これだけ！

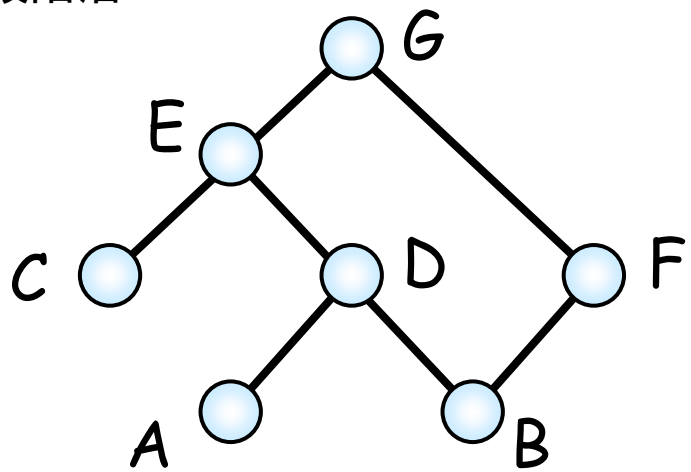
0	1	0	0
1	0	1	0
0	1 &	1	1
1	0	1	0
0	1	0	0

ここは同じ

$$R_i = (R_{i-1} \ll 1 \mid 1) \,\&\, M(T[i])$$

分類階層への対応

分類階層 H:



Mask table M'

	A	B	C	D	E	F	G
A ▶	1	0	0	0	0	0	0
B ▶	0	1	0	0	0	0	0
C ▶	0	0	1	0	0	0	0
D ▶	1	1	0	1	0	0	0
E ▶	1	1	1	1	1	0	0
F ▶	0	1	0	0	0	1	0

パターンP:= A B E F

テキストT:= A B C B D F C B

O(mh) ?

M'(a)の計算(補題)

- **補題1**
  (Σ,≥)をソート付きアルファベットとし、パターンP∈Σ*が与えられたとする。このとき、任意の a∈Σ について

$$M'(a) = \bigcup_{x \in Upb(a)} M(x)$$

  が成り立つ。

- **補題2**
  (Σ,≥)をソート付きアルファベットとし、パターンP∈Σ*が与えられたとする。このとき、任意の a∈Σ について

$$M'(a) = M(a) \cup \bigcup_{x \in Par(a)} M'(x)$$

  が成り立つ。

M’(a)を計算するアルゴリズムの擬似コード

```
Preprocess_M’ (P=p1…pm)  /* 分類階層Hはグローバル */
    M(a)を次のように初期化;
        M(a)={1≦i≦m | P[i]=a};
    for each a∈Σ do
        CalculateM’(a);
    end of for

Function CalculateM’(a)
    if M’(a)は計算済み then return M’(a)
    else do
        M’(a) = M(a);
        for each x∈Par(a) do
            M’(a)=M’(a)∪(CalculateM’(x));
        end of for
    return M’(a);
```

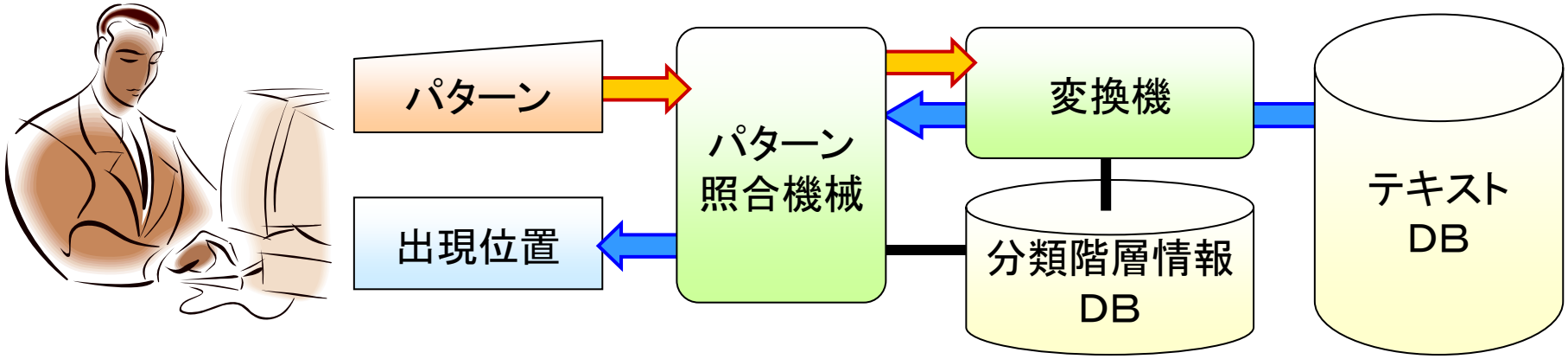
O(m)

O(h)

Total O(m+mh/w)

O(m/w)

PMTXアルゴリズムによる検索システムの概要



テキストを概念の列に切り分ける必要がある

変換機

Replace Automaton（有川・白石[1984]）

O(h+n)

もしくは茶筅のような自然言語解析機を使う

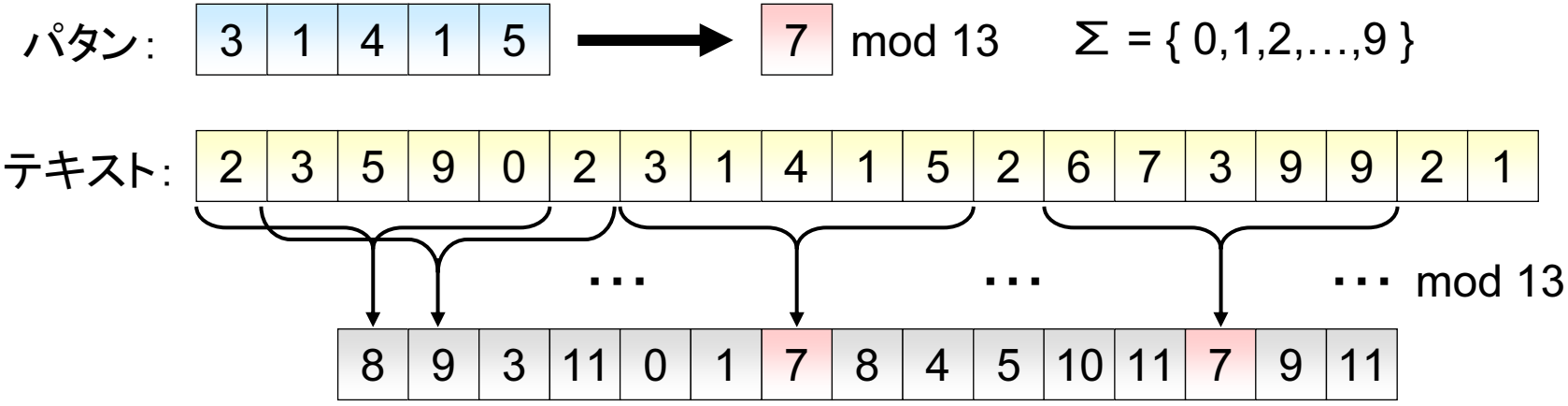
第7回 まとめ

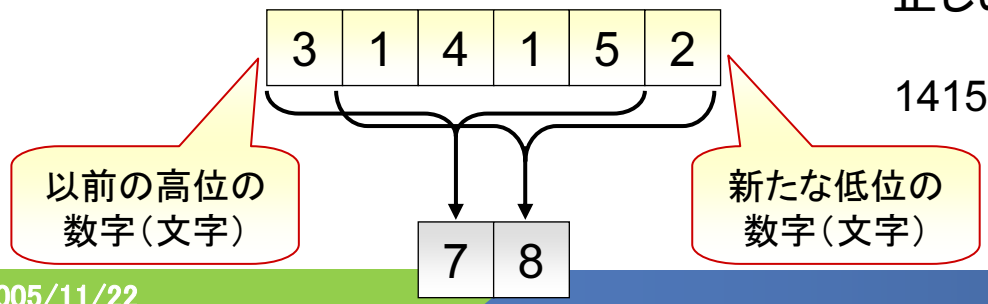
- 多バイトコードへの対応方法
  - 同期付きオートマトンをAC照合機械へ組み込む
  - Maskビット列を出力するコード・オートマトンをビットパラレル手法に組み合わせる
- 知的なパターン照合を目指して:
  - テキストの構造を考慮した照合
    - XMLテキストに対するパターン照合
    - Arc注釈付きテキストに対するパターン照合
  - テキストの意味を考慮した照合(オントロジーデータとの連携)
    - 分類階層を考慮したパターン照合
- 次回から有村博紀先生が担当します。
  - 情報検索のための効率のよい索引データ構造
  - ウェブからのデータマイニングほか

Karp-Rabinアルゴリズム

KARP R.M., RABIN M.O., Efficient randomized pattern-matching algorithms. IBM J. Res. Dev. 31(2):249-260, 1987.

- Hashingを使ったRandomizedアルゴリズム
  - 文字列を一個の整数とみなして照合する！
- 最悪時O(mn)時間かかるが、平均時はO(n+m)時間
- Extra spaceがO(1)！





$$14152 \equiv (31415 - 3 \times 10000) \times 10 + 2 \pmod{13}$$
$$\equiv (7 - 3 \times 3) \times 10 + 2 \pmod{13}$$
$$\equiv 8 \pmod{13}$$

擬似コード

```
Karp-Rabin (P, T, d, q)
    m ← length[P].
    n ← length[T].
    h ← d^(m−1) mod q.
    p ← 0.
    t_0 ← 0.
    for i ← 1 to m do
        p ← (d・p + P[i]) mod q;
        t_0 ← (d・t_0 + T[i]) mod q.
    for s ← 0 to n – m do
        if p = t_s then
            if P[1…m] = T[s+1…s+m] then
                report an occurrence at s;
        else if s < n – m then
            t_(s+1) ← (d・(t_s – T[s+1]・h)+T[s+m+1]) mod q.
```

正しい出現かどうかを確認

FFTを利用した確率的近似文字列照合

K. Baba, A. Shinohara, M. Takeda, S. Inenaga, and S. Arikawa. A Note on Randomized Algorithm for String Matching with Mismatches. *Nordic Journal of Computing*, 10(1):2-12, 2003.

- Fast Fourier Transform (FFT)は
  ハードウェア上で高速に計算可能
- 文字列を数値に置き換え、スコアベクトルの列を
  FFTにより高速に計算することで、(近似)文字列
  照合を行う

馬場さん(九州大)

i :	1	2	3	4	5	6	7	8	9	10
T[i] =	a	c	b	a	b	b	a	c	c	b
P =	a	b	b	a	c					
		a	b	b	a	c				
			a	b	b	a	c			
				a	b	b	a	c		
					a	b	b	a	c	
						a	b	b	a	c
c_i =	3	1	1	5	2	0				

スコアベクトル

$$\delta(a,b) = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases}$$

$$c_i = \sum_{j=1}^{m} \delta(t_{i+j-1}, p_j)$$