



HOKKAIDO UNIVERSITY

Title	ビット並列計算を用いた高度なパターン照合
Author(s)	金田, 悠作
Description	ERATO湊離散構造処理系プロジェクト : 2010年度初冬のワークショップ (ERATO合宿) . 2010年11月29日 (月) ~12月1日 (水) . 札幌北広島クラッセホテル.
Relation	2010年度科学技術振興機構ERATO湊離散構造処理系プロジェクト講究録. p.404-412.
Issue Date	2011-06
Doc URL	https://hdl.handle.net/2115/48352
Type	conference presentation
File Information	05.06_kaneta.pdf



ビット並列計算を用いた 高度なパターン照合

北海道大学
博士後期課程2年

金田 悠作

17th String Processing and Information Retrieval Symposium
(SPIRE2010)

Fast Bit-Parallel Matching for Network and Regular Expressions

Yusaku Kaneta, Shin-ichi Minato, and Hiroki Arimura

Graduate School of Information Science and Technology
Hokkaido University, Japan

Background

- Regular expression matching problem
 - Fundamental problem in string processing
- Its new applications have emerged such as
 - NIDS (Network Intrusion Detection Systems)
 - ESP (Event Stream Processing)
- There are demands for large-scale pattern matching systems
 - However, these systems have to cope with ...

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合-金田 悠作 (北海道大学)

3

Example of real regular expressions in NIDS applications

Three Perl compatible regular expressions (PCREs) in SNORT 2.8
within the class **EXNET** of extended network expressions of depth 1

```
R1: /\^If-Modified-Since\^x3a(?:((Sun|Mon|Tue|Wed|Thu|Fri|Sat),[0-9]?[0-9](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}){2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thurs|Fri|Sat)day,[0-9]{2})[- ](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[- ]([0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thu|Fri|Sat)(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}([0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2})/sm
R2: /\^Last-Modified\^x3a(?:((Sun|Mon|Tue|Wed|Thu|Fri|Sat),[0-9]?[0-9](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}){2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thurs|Fri|Sat)day,[0-9]{2})[- ](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[- ]([0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thu|Fri|Sat)(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}([0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2})/sm
R3: /\^Date\^x3a(?:((Sun|Mon|Tue|Wed|Thu|Fri|Sat),[0-9]?[0-9](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}){2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thurs|Fri|Sat)day,[0-9]{2})[- ](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[- ]([0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thu|Fri|Sat)(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}([0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2})/sm"
```

Example of real regular expressions in NIDS applications

Three Perl compatible regular expressions (PCREs) in SNORT 2.8
within the class **EXNET** of extended network expressions of depth 1

Massive
(hundreds to thousands)

Complex
(regular expressions)

Hi-speed
(data streams with 1 to 10Gbps)

R1: /\^If-Modified-Since\^x3a(?:((Sun|Mon|Tue|Wed|Thu|Fri|Sat),[0-9]?[0-9](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}){2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thurs|Fri|Sat)day,[0-9]{2})[-](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[-]([0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thu|Fri|Sat)(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}([0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2})/sm
R2: /\^Last-Modified\^x3a(?:((Sun|Mon|Tue|Wed|Thu|Fri|Sat),[0-9]?[0-9](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}){2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thurs|Fri|Sat)day,[0-9]{2})[-](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[-]([0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thu|Fri|Sat)(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}([0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2})/sm
R3: /\^Date\^x3a(?:((Sun|Mon|Tue|Wed|Thu|Fri|Sat),[0-9]?[0-9](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}){2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thurs|Fri|Sat)day,[0-9]{2})[-](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[-]([0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thu|Fri|Sat)(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}([0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2})/sm"

Example of real regular expressions in NIDS applications

Three Perl compatible regular expressions (PCREs) in SNORT 2.8
within the class **EXNET** of extended network expressions of depth 1

Massive
(hundreds to thousands)

Complex
(regular expressions)

Hi-speed
(data streams with 1 to 10Gbps)

There is much attention to

- Large-scale pattern matching on hardware
- Hardware-oriented matching algorithms

R1: /\^If-Modified-Since\^x3a(?:((Sun|Mon|Tue|Wed|Thu|Fri|Sat),[0-9]?[0-9](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}){2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thurs|Fri|Sat)day,[0-9]{2})[-](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[-]([0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thu|Fri|Sat)(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}([0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2})/sm
R2: /\^Last-Modified\^x3a(?:((Sun|Mon|Tue|Wed|Thu|Fri|Sat),[0-9]?[0-9](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}){2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thurs|Fri|Sat)day,[0-9]{2})[-](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[-]([0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thu|Fri|Sat)(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}([0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2})/sm
R3: /\^Date\^x3a(?:((Sun|Mon|Tue|Wed|Thu|Fri|Sat),[0-9]?[0-9](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}){2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thurs|Fri|Sat)day,[0-9]{2})[-](Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[-]([0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}((ECMP|S|GM)T|((Sun|Mon|Tue|Wed|Thu|Fri|Sat)(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[0-9]{2}|[0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2}([0-9]{4})[0-9]{2}\^x3a[0-9]{2}\^x3a[0-9]{2}){2})/sm"

Background

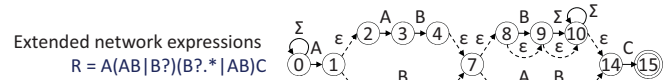
- Large-scale pattern matching systems have to cope with requirements
 - Massive (hundreds to thousands)
 - Complex patterns (regular expressions)
 - High-speed data streams (with 1 to 10 Gbps)
- Hardware-oriented matching algorithms
 - DFA-based, NFA-based, CAM-based ...
 - None of these satisfies all of the above three requirements
- In this research, we focus on **bit-parallel methods for efficient regular expression matching**

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

7

Classes of regular expressions

- We study the regular expression matching problem for the following subclasses of regular expressions
 - Regular expressions (REG)**
 - Network expressions (NET)**: The subclass of REG without Kleene-star [Myers, '96]
 - Extended network expressions (EXNET)**: (our main target class) Network expressions over the class EXT of extended strings
 - which are concatenations of letter α , class of letters $[ab\dots]$, optional letter $a?$, bounded repeat $a\{lo, hi\}$, and unbounded repeats a^* and a^+ .
 - These subclasses are widely used in real world applications such as NIDS and ESP



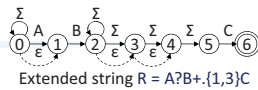
2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

8

Bit-Parallel method

SHIFT-AND approach

- SHIFT-AND method** [Baeza-Yates & Gonnet, '92]
 - Bit-parallel matching algorithm for the class STR of strings
 - Boolean (&, |) operations only
 - Simple and efficient
- Extended SHIFT-AND method** [Navarro & Raffinot, '01]
 - A nice extension to the class EXT of extended strings
 - Boolean (&, |, ~, $\oplus$) and arithmetic (-) operations
 - Still simple and efficient
- Challenge: Expressiveness of matching patterns**
- Open problem: Can we extend Extended SHIFT-AND method to more general subclasses of REG?**



2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

9

Related work

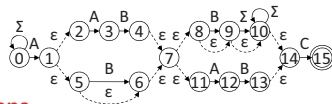
- Bit-parallel Thompson** [Wu & Manber, '92]
 - Bit-parallel matching algorithm for the class REG
 - Boolean (&, |) operations
 - Table-lookup
- Myers's algorithm** [Myers, '92]
 - Matching algorithm for the class REG
 - Table-lookup
 - Module decomposition of Thompson NFA
- Bille's algorithm** [Bille, ICALP2006]
 - Bit-parallel matching algorithm for the class REG
 - Boolean (&, |) and arithmetic (-) operations
 - Module decomposition of Thompson NFA

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

10

Our results

- We developed **Extended² SHIFT-AND method**
- Fast bit-parallel matching algorithm for extended network expressions** that runs in
 - $O(ndm/w)$ time
 - $O(dm/w)$ space
 - $O(dm)$ preprocessing
- Extension for **regular expressions**
- Experimental results on hardware implementation of our algorithm
- Keys**
 - Bi-monotonicity lemma** for Thompson NFA
 - Bit-parallel operations **Scatter**, **Gather**, and **Propagate** for simulating ϵ -moves



m : size of R , n : size of T , d : depth of R , w : word length

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

11

Comparison: Time & Space complexities

- Our **Extended² SHIFT-AND method** is the **first extension** of SHIFT-AND and Extended SHIFT-AND methods to the following classes:
 - Extended network expressions (EXNET)**
 - Regular expressions (REG)**

Algorithm	Class	Time	Space (in words)
SHIFT-AND [Baeza-Yates & Gonnet, '92]	STR	$O(nm/w)$	$O(m/w)$
Extended SHIFT-AND [Navarro & Raffinot, '01]	EXT	$O(nm/w)$	$O(m/w)$
Extended² SHIFT-AND (in this talk)	EXNET	$O(ndm/w)$	$O(dm/w)$
	REG	$O(nd\log(m)m/w)$	$O(d\log(m)m/w)$

m : size of R , n : size of T , d : depth of T , w : word length

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

12

Hardware implementation

• Merit of our Extended² SHIFT-AND method

- **Simple and Efficient:** Our algorithm is particularly efficient for **expression of small depth and regular structure**. Therefore, it is suitable to applications such as NIDS and ESP.
- **Hardware friendly:** Our algorithm is suitable to **modern parallel hardware, such as FPGA and GPGPU** since it uses only simple Boolean and arithmetic operations (+, −) avoiding the heavy use of table-lookup.

FPGA = Field Programmable Gate Array
GPGPU = General Purpose GPU

• Implementation

- We implemented our algorithm on FPGA.
- The update formulae of bit-parallel matching are transformed into a fixed circuitry on FPGA in advance.
- In preprocessing, bitmasks are loaded to block RAMs on FPGA.
- In runtime, The NFA equivalent to a given expression is simulated by the circuitry.

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

13

Hardware implementation

• Experimental settings

- Our circuit written in Verilog HDL is compiled and simulated on Xilinx Virtex-5 FPGA LX330 (51,840 slices and 1 MB block RAMs) using Xilinx ISE Design Suite and Synopsys VCS.

• Experimental results

- 128 patterns of length 32 can be installed on the FPGA
- Our algorithm implemented on the FPGA (0.6 GHz clock) achieves the high throughput of 0.5 Gbps.

Algorithm	Class	#pat	#op	#add	Throughput
Extended ² SHIFT-AND (in this talk)	EXNET	128	20	9	0.5 Gbps

#pat, #op, and #add are the number of input expressions, 32-bit Boolean-operations, and 32-bit integer additions, respectively

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

14

Algorithms

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

15

Our bit-parallel algorithm

• Outline of algorithm

Algorithm Extended² SHIFT-AND

Preprocessing

- Transform an input regular expression R to the equivalent NFA N_R (Thompson NFA, TNFA).
- Construct a set of bitmasks for TNFA.

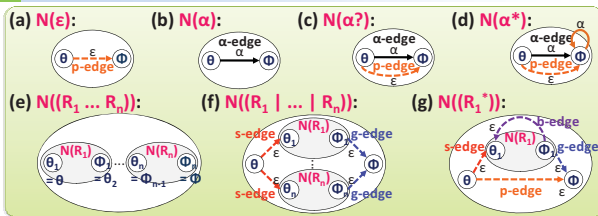
Runtime:

- Simulate TNFA N_R on an input text by using the bitmasks based on bit-parallel method.

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

16

Thompson NFA [Thompson, CACM1968]



- A NFA equivalent to a regular expression R
 - The **source** θ and the **sink** ϕ
- **Depths** $d(S)$ and $d(x)$ of subexpressions S and states x
 - The **number of nesting of Union** " $|$ "
- TNFA for EXNET has three types of ϵ -edges
 - **Scatter edges**, **Gather edges**, and **Propagate edges**
- TNFA for REG has one more type of ϵ -edges
 - **Back edges**

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

17

Simulation of Thompson NFA

• We use bit-parallel method

- Encode a state set of TNFA in a bitmask D

• Procedure RunTNFA:

- Simulates α -moves $\text{Move}_N(D, \alpha)$ in the same way as SHIFT-AND approaches [Baeza-Yates & Gonnet, '92][Navarro & Raffinot, '01]
- Simulates ϵ -closure $\text{EpsClo}_N(D)$ by combining bit-parallel operations **Scatter**, **Gather**, and **Propagate** [This talk]

```

Procedure RunTNFA( $T = t_1 \dots t_n$ : input text)
1.  $D \leftarrow \text{Init}_N$ ;
2.  $D \leftarrow \text{EpsClo}_N(D)$ ;
3. for  $t \leftarrow t_1, \dots, t_n$  do
4.   if  $D \& \text{Accept}_N \neq 0$  then
5.     output match  $i-1$ ;
6.    $D \leftarrow \text{Move}_N(D, t)$ ;
7.    $D \leftarrow \text{EpsClo}_N(D)$ ;
8. end for
9. return  $D$ ;
  
```

$\text{Move}_N(D, \alpha)$
 $\equiv D \leftarrow (((D \ll 1) \& \text{CHR}[\alpha]) | 1)$
 $| (D \& \text{REP}[t]);$

$\text{Init}_N \equiv 0^{m-1};$

$\text{Accept}_N \equiv 10^{m-1};$

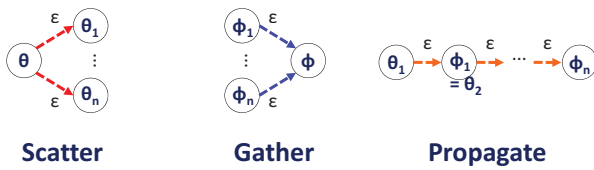
In the following, we assume that a constant alphabet Σ , where $|\Sigma| = O(1)$

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

18

Our bit-parallel algorithm

- **Key:** Computation of ϵ -closure $\text{EpsClo}_N(D)$
- Consists of two components:
 1. **Bi-monotonicity lemma** for Thompson NFA
 2. Bit-parallel implementation of three ϵ -move operations: **Scatter**, **Gather**, and **Propagate**



2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

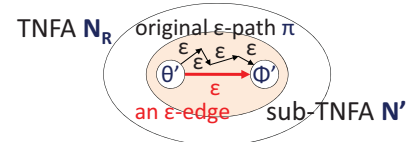
19

Bimonotonicity lemma

- Let $N_R = (V, E, \theta, \phi)$ be a TNFA with source θ and sink ϕ

Procedure Bypassing

1. Visit every **sub-TNFA** N' of N_R whose source θ' and sink ϕ' are connected by an ϵ -path π
2. Add **an ϵ -edge** directly connecting θ' and ϕ'



- Bypassing can be done in $O(m)$ time in preprocessing
- **Expand(R)** denotes the NFA obtained from N_R by bypassing

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

20

Bimonotonicity lemma

- Let x, y be any states in TNFA N_R for **Expand(R)**
- For any states x, y , define $d(x) \leq_1 d(y)$ iff $d(x) - d(y) \leq 1$

Lemma 1. (Bi-monotonicity lemma)

If there is an ϵ -path π from x to y , then **there also exists some bi-monotone ϵ -path** $\pi' = (x_1 = x, \dots, x_n = y)$ from x to y in N_R such that $d(x_1) \geq_1 \dots \geq_1 d(x_k)$ and $d(x_k) \leq_1 \dots \leq_1 d(x_n)$ (Eq1)

Errata: There is a **mistake** in the definition of the bimonotonicity in Page 379 of the conference proceedings. **The direction of inequality $\leq_1$ is opposite to the correct one :-)** Please correct it as in the above (Eq1).

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

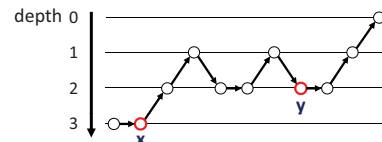
21

Bimonotonicity lemma

- Let x, y be any states in TNFA N_R for **Expand(R)**
- For any states x, y , define $d(x) \leq_1 d(y)$ iff $d(x) - d(y) \leq 1$

Lemma 1. (Bi-monotonicity lemma)

If there is an ϵ -path π from x to y , then **there also exists some bi-monotone ϵ -path** $\pi' = (x_1 = x, \dots, x_n = y)$ from x to y in N_R such that $d(x_1) \geq_1 \dots \geq_1 d(x_k)$ and $d(x_k) \leq_1 \dots \leq_1 d(x_n)$



Step 0. Suppose we have a TNFA N_R with an ϵ -path π from state x to state y

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

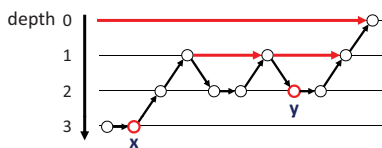
22

Bimonotonicity lemma

- Let x, y be any states in TNFA N_R for **Expand(R)**
- For any states x, y , define $d(x) \leq_1 d(y)$ iff $d(x) - d(y) \leq 1$

Lemma 1. (Bi-monotonicity lemma)

If there is an ϵ -path π from x to y , then **there also exists some bi-monotone ϵ -path** $\pi' = (x_1 = x, \dots, x_n = y)$ from x to y in N_R such that $d(x_1) \geq_1 \dots \geq_1 d(x_k)$ and $d(x_k) \leq_1 \dots \leq_1 d(x_n)$



Step 1. Applying **bypassing transformation** to the TNFA N_R

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

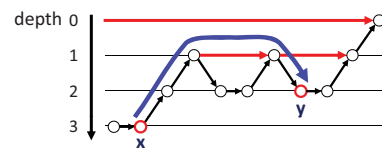
23

Bimonotonicity lemma

- Let x, y be any states in TNFA N_R for **Expand(R)**
- For any states x, y , define $d(x) \leq_1 d(y)$ iff $d(x) - d(y) \leq 1$

Lemma 1. (Bi-monotonicity lemma)

If there is an ϵ -path π from x to y , then **there also exists some bi-monotone ϵ -path** $\pi' = (x_1 = x, \dots, x_n = y)$ from x to y in N_R such that $d(x_1) \geq_1 \dots \geq_1 d(x_k)$ and $d(x_k) \leq_1 \dots \leq_1 d(x_n)$



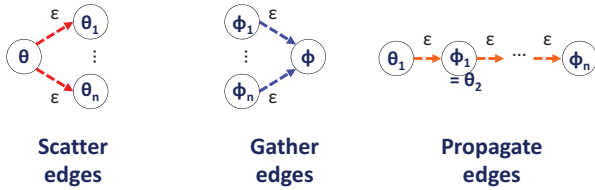
Step 2. There exists a **bimonotone ϵ -path** connecting x and y

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

24

Bit-parallel implementation

- **Scatter** simulates ϵ -moves by **scatter edges**.
- **Gather** simulates ϵ -moves by **gather edges**.
- **Propagate** simulates ϵ -closure by **propagate edges**.
- Operations are separately done for each depth by a specified order.

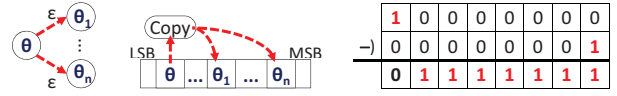


2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

25

Scatter operation

- **Basic idea**: carry propagation of **integer subtraction**
(The origin of using carry propagation is due to [Navarro & Raffinot, '01])



Preprocess: for depth k , we construct the following bitmasks

- $BLK_S[k]$: the state $j = \theta_n + 1$
- $SRC_S[k]$: the source $j = \theta$
- $DST_S[k]$: the destinations $j \in \{\theta_1, \dots, \theta_n\}$

Runtime: for state set D and depth k , we perform

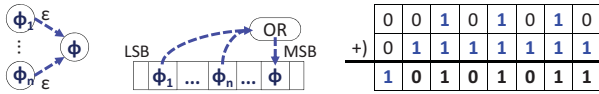
- $Scatter(D, k) \equiv \{ D \leftarrow D \mid ((BLK_S[k] - (D \& SRC_S[k])) \& DST_S[k]); \}$

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

26

Gather operation

- **Basic idea**: carry propagation of **integer addition**
(The origin of using carry propagation is due to [Navarro & Raffinot, '01])



Preprocess: for depth k , we construct the following bitmasks

- $BLK_G[k]$: the states j in interval $[\Phi_1, \Phi-1]$
- $SRC_G[k]$: the sources $j \in \{\Phi_1, \dots, \Phi_n\}$
- $DST_G[k]$: the destination $j = \Phi$

Runtime: for state set D and depth k , we perform

- $Gather(D, k) \equiv \{ D \leftarrow D \mid ((BLK_G[k] + (D \& SRC_G[k])) \& DST_G[k]); \}$

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

27

Propagate operation

- **Basic idea**: **Propagate operation** of Extended SHIFT-AND for the ϵ -closure of a set of ϵ -blocks [Navarro & Raffinot, '01]
■ An ϵ -block is a maximal consecutive sequence of ϵ -edges.



Preprocess: for depth k , we construct the following bitmasks

- $BLK_P[k]$: the states j in ϵ -block $B = \{\theta_1, \Phi_1 = \theta_2, \dots, \Phi_n\}$
- $SRC_P[k]$: the least significant state $j = \min(B) = \theta_1$ in ϵ -block B
- $DST_P[k]$: the most significant state $j = \max(B) = \Phi_n$ in ϵ -block B

Runtime: for state set D and depth k , we perform

- $Propagate(D, k) \equiv \{ A \leftarrow (D \& BLK_P[k]) \mid DST_P[k]; D \leftarrow D \mid (BLK_P[k] \& ((A - SRC_P[k])) \oplus A); \}$

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

28

Putting them together

- Scatter, Gather, and Propagate can be computable in $O(m/w)$ time using $O(m/w)$ space and $O(m)$ preprocessing.
- The following procedure correctly computes the ϵ -closure $EpsClo_N(D)$ in $O(dm/w)$ time using $O(dm/w)$ space and $O(dm)$ preprocessing, where $d = d(R)$.

Procedure $EpsClo_N(D)$: a state set of TNFA N_R

1. for $k \leftarrow d(R), \dots, 1$ do
2. $D \leftarrow Propagate(D, k);$
3. $D \leftarrow Gather(D, k-1);$
4. end for
5. $D \leftarrow Propagate(D, 0);$
6. for $k \leftarrow d(R), \dots, 1$ do
7. $D \leftarrow Scatter(D, k-1);$
8. $D \leftarrow Propagate(D, k);$
9. end for
10. return D ;

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

29

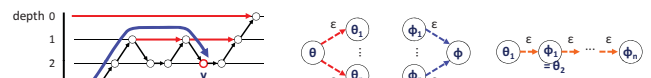
Main result for the class EXNET

- Combining $Move_N$ and $EpsClo_N$ with our algorithm **Extended² SHIFT-AND**, we have:

Theorem 1. Our bit-parallel algorithm for **EXNET** solves the regular expression matching problem for **EXNET** in

- $O(ndm/w)$ time
- $O(dm/w)$ space
- $O(dm)$ preprocessing

m : size of R , n : size of T , d : depth of T , w : word length



2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

30

Extension to the class REG

- For an extension of our algorithm for REG by the barrel shifter technique in VLSI design, we have:

Theorem 2. Our modified algorithm for REG solves the regular expression matching problem for REG of general regular expressions in

- $O(nd\log(m)w)$ time
- $O(d\log(m)w)$ space
- $O(d\log(m)m)$ preprocessing

m : size of R , n : size of T , d : depth of T , w : word length

- Note:
- If there are at most constant number of back edges with mutually distinct lengths, then we can replace the $O(\log m)$ term with $O(1)$
 - If the $O(1)$ -bit-reversal operation is available, then we can also replace the $O(\log m)$ term with $O(1)$

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

31

Conclusion

- Fast bit-parallel matching algorithm for extended network expressions that runs in
 - $O(ndm/w)$ time
 - $O(dm/w)$ space
 - $O(dm)$ preprocessing
- Extension for regular expressions
- We implemented our algorithm on FPGA, and achieves the high throughput of 0.5 Gbps
- Future works:
 - Tree and XML matching

m : size of R , n : size of T , d : depth of R , w : word length

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

32

Thank you

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

33

21st International Workshop on Combinatorial Algorithms (IWOCAL'10)

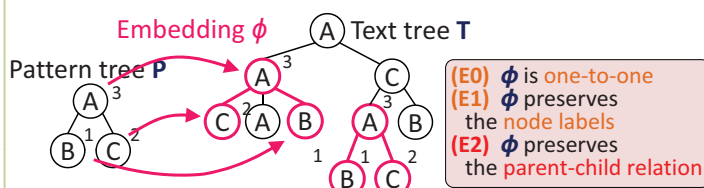
Faster Bit-Parallel Algorithms for Unordered Pseudo-Tree Matching and Tree Homeomorphism

Yusaku Kaneta and Hiroki Arimura

Graduate School of Information Sci. and Tech.
Hokkaido University, Japan

Background: Tree matching problem

- Problem of finding an embedding ϕ from a pattern tree P to a text tree T
- Fundamental problem in computer science [Kilpelainen & Mannila, '94]
- It has many applications
- We consider unordered tree matching and its variants (for labeled, rooted tree)



2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

38

Background: Many-to-one matching

- In original theoretical studies: Tree matching with one-to-one mapping has been mainly studied so far
- In recent practical studies: Tree matching with many-to-one mapping attracts much attention
- Goal:** To develop efficient algorithms for two tree matching problems with many-to-one mappings
 - Unordered pseudo-tree matching problem (UPTM)
 - ⇔ XPath queries with child axis only
 - Unordered tree homeomorphism problem (UTH)
 - ⇔ XPath queries with descendant axis only

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

39

Definition

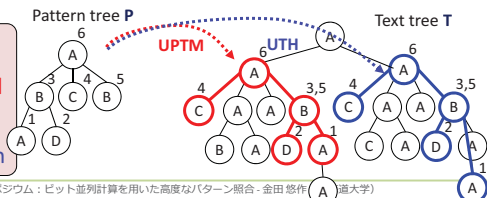
Unordered pseudo-tree matching problem (UPTM)

- A pattern tree P matches a text tree T if there is a many-to-one mapping $\phi: V(P) \rightarrow V(T)$ from P into T satisfying the conditions (E1) and (E2)
- An occurrence of P in T is the image of the root of P
- The problem is to find all occurrences of P in T

Unordered tree homeomorphism problem (UTH)

- is defined similarly, where many-to-one mapping satisfying (E1) and (E3) is used.

ϕ preserves:
(E1) the node labels
(E2) the parent-child relation
(E3) the ancestor-descendant relation



2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

Related work

Many studies for tree matching with one-to-one mappings

- [Kilpelainen, Mannila, SIAM J'95]:
The unordered tree matching and inclusion problems
- Corresponds to the subgraph isomorphism problem

Few studies for tree matching with many-to-one mappings

[Yamamoto, Takenouchi, WADS'09] UPTM problem

- $O(nr \cdot \text{leaves}(P) \cdot \text{depth}(P)/w) = O(nm^3/w)$ time
- $O(n \cdot \text{leaves}(P) \cdot \text{depth}(P)/w) = O(nm^2/w)$ space

[Gotz, Koch, Martens, DBPL'07] UTH problem

- $O(nm \cdot \text{depth}(P)) = O(nm^2)$ time
- $O(\text{depth}(T) \cdot \text{branch}(T)) = O(n^2)$ space

m : the size of P , n : the size of T , h : the height of T , w : the word length, and r : the maximum number of the same label on paths in P

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

41

Our results

- New decomposition formula for unordered pseudo-tree matching problem (UPTM)
- Bit-parallel algorithm for UPTM that runs in
 - $O(nm \log(w)/w)$ time
 - $O(hm/w + m \log(w)/w)$ space
 - $O(m \log(w))$ preprocessing time
- Key: Fast bit-parallel computation of Tree aggregation in $O(\log m)$ time
 - Improves a naïve implementation in $O(m)$ time
- Modified algorithm for UTH with the same complexity

m : the size of P , n : the size of T , h : the height of T , w : the word length

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

42

Summary

Algorithm for UPTM	Time	Space (in words)
BP-MatchUPTM (this work)	$O(nm \log(w)/w)$	$O(hm/w + m \log(w)/w)$
[Yamamoto, Takenouchi, WADS'09]	$O(nm^3/w)$	$O(nm^2/w)$

- Our algorithm improves the algorithm by [YT'09] (by $O(m^2/\log(w))$)

Algorithm for UTH	Time	Space (in words)
BP-MatchUTH (this work)	$O(nm \log(w)/w)$	$O(hm/w + m \log(w)/w)$
[Gotz, Koch, Martens, DBPL'07]	$O(nm^2)$	$O(hn)$

- Our algorithm improves the algorithm by [Gotz et al.'07]
- This is the first bit-parallel algorithm for UTH (by $O(mw/\log(w))$)

m : the size of P , n : the size of T , h : the height of T , w : the word length

[Yamamoto, Takenouchi, WADS'09] H. Yamamoto and D. Takenouchi, Bit-parallel tree pattern matching algorithms for unordered labeled trees, In Proc. WADS'09, 554-565, 2009.

[Gotz, Koch, Martens, DBPL'07] M. Gotz, C. Koch, and W. Martens, Efficient algorithms for tree homeomorphism problem, In Proc. DBPL'07, 17-31, 2007.

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

43

Algorithm for the UPTM problem

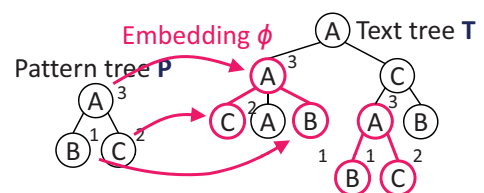
2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

44

Our algorithm MatchUPTM

Consists of two components:

- New decomposition formula for bottom-up computataion
- Bit-parallel implementation of five set operations: Constant, Union, Member, LabelMatch_p, and TreeAggr_p
 - Especially, $O(\log m)$ time bit-parallel implementation of TreeAggr operation



2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

45

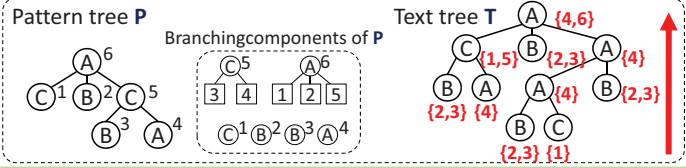
Decomposition formula for UPTM

- The **embedding set** $\text{Emb}^{P,T}(v)$ of text node $v \in V(T)$
 - is the set of pattern node $x \in V(P)$ such that $P(x)$, the subtree of P rooted at x , occurs in T at node v

Lemma 1 (decomposition formula): For any $x \in V(P)$, $v \in V(T)$, $x \in \text{Emb}^{P,T}(v)$

- $\Leftrightarrow$ (i) **Label matching:** $\text{label}_P(x) = \text{label}_T(v)$ and
(ii) **Tree aggregation:** $\text{children}(x) \subseteq \bigcup_{1 \leq j \leq \alpha(v)} \text{Emb}^{P,T}(v[j])$

- From Lemma1, we can develop a **bottom-up algorithm** for **UPTM** in $O(nm)$ time and $O(hm)$ space, where h is the height of T



2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

46

Bit-parallel implementation

- To obtain further speed-up, we use **bit-parallelism**
 - Encoding an embedding set $\text{Emb}(v) \subseteq \{1, \dots, m\}$ for each node v by a **bitmask** $X \in \{0, 1\}^m$ of length m .
 - By implementing the **five set operations** by using **Bit-wise Boolean operations** $\&$, $|$, $\sim$ and **integer addition** $+$ [BGY'92]
- Key: Bit-parallel implementation of **TreeAggr_P**

Operation	Original impl.	Bit-parallel impl.
Constant(S)	$O(m)$ time	$O(m/w)$ time
Union(R, S)	$O(m)$ time	$O(m/w)$ time
Member(R, x)	$O(m)$ time	$O(m/w)$ time
LabelMatch _P (R, α)	$O(m)$ time	$O(m/w)$ time (From [BGY'92])
TreeAggr _P (R, S)	$O(m)$ time	$O(m \log(w)/w)$ time (This work)

[BGY'92] R. Baeza-Yates and G. H. Gonnet, CACM, 35(10), 74-82, 1992.

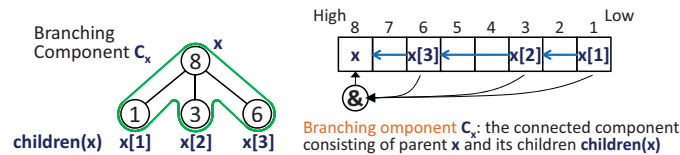
m : the size of P , n : the size of T , w : the word length

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

47

Bit-parallel tree aggregation

- Computes the parent value as the logical AND of the children values
- Preprocess:** Build the following bitmasks
 - DST:** the position of parent x
 - SRC:** the positions of children $\text{children}(x)$
 - SEED:** the lowest position of component C_x
 - INT:** the interval of C_x except for x and $\text{children}(x)$
- Runtime:** Simulate tree aggregation by bit-operations



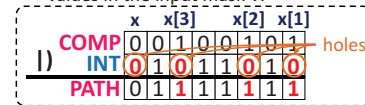
2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

48

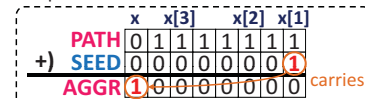
Bit-parallel tree aggregation

- Basic idea: Using the carry propagation by integer addition**

- Line2:** Compute the **PATH** mask. We fill the "holes" at the children positions in **INT** with the children values in the input mask **Y**.

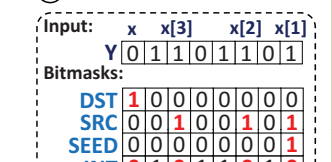
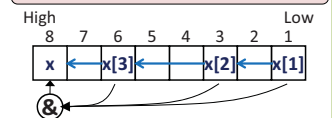


- Line3:** Compute the **AGGR** mask. If all the "holes" in **PATH** are filled then the parent value is set



Runtime:

- COMP** $\leftarrow Y \& \text{SRC}$;
- PATH** $\leftarrow \text{COMP} | \text{INT}$;
- AGGR** $\leftarrow \text{PATH} + \text{SEED}$;
- RESULT** $\leftarrow \text{AGGR} \& \text{DST}$;



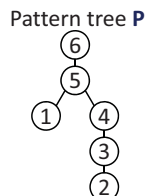
2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

49

Separator tree-based decomposition

- By using the **separator tree-based decomposition technique**, we can implement **Tree Aggregation** in $O(\log(m))$ time using $O(m \log m)$ preprocessing time

Lemma (Jordan, 1869). Let S be a binary tree. Then, there exists a node in S such that $|S(v)| \leq (2/3)|S|$ and $|S(v')| \leq (2/3)|S|$, where $S(v)$ is the subtree of S rooted at v and $S(v')$ is the tree obtained by pruning $S(v)$ from S .



Naïve decomposition:

Bit-position	1	2	3	4	5	6
Node	1	2	3	4	5	6
Level 1					5	6
Level 2	1			4	5	
Level 3			3	4		
Level 4		2	3			

$O(m)$

Separator tree-based decomposition:

Bit-position	1	2	3	4	5	6
Node	2	3	4	1	5	6
Level 1	1			4	5	
Level 2	2	3				
Level 3			3	4	5	6

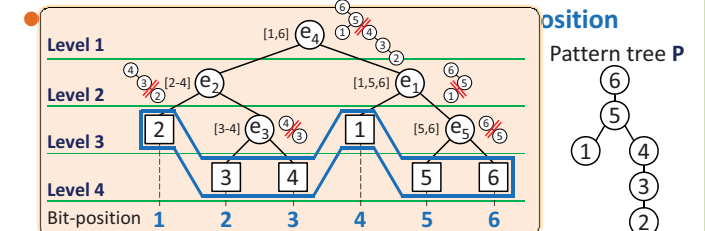
$O(\log m)$

Bit-assignment also differs from naïve decomposition.

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

50

Separator tree-based decomposition



Naïve decomposition:

Bit-position	1	2	3	4	5	6
Node	1	2	3	4	5	6
Level 1					5	6
Level 2	1			4	5	
Level 3			3	4		
Level 4		2	3			

$O(m)$

Separator tree-based decomposition:

Bit-position	1	2	3	4	5	6
Node	2	3	4	1	5	6
Level 1	1			4	5	
Level 2	2	3				
Level 3			3	4	5	6

$O(\log m)$

Bit-assignment also differs from naïve decomposition.

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

51

Main result for the UPTM problem

- By applying the module decomposition techniques of [Myers '92] and [Bille '06], we have:

Theorem 1. (complexity of the UPTM problem)

The algorithm **BP-MatchUPTM** solves the unordered pseudo-tree matching problem in

- $O(nm \log(w)/w)$ time, using
- $O(hm/w + m \log(w)/w)$ space and
- $O(m \log(w))$ preprocessing time

m : the size of P , n : the size of T , h : the height of T , w : the word length

Note: This improves the time complexity $O(nm^3/w)$ of the previous bit-parallel algorithm by [Yamamoto & Takenouchi, WADS'09] with a factor of $O(m^2/\log(w))$

[Bille'06] P. Bille, New algorithms for regular expression matching, In *Proc. ICALP'06*, 643-654, 2006.

[Myers'92] E. W. Myers, A four-russian algorithm for regular expression pattern matching, *JACM*, 39(2), 430-448, 1992.

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

52

Main result for the UTH problem

- Modified Bit-parallel algorithm **BP-MatchUTH**:
 - Based on a similar decomposition formula
 - The code is same as VisitUPTM except line 9

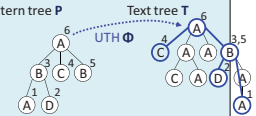
Theorem 2. (complexity of the UTH problem)

The algorithm **BP-MatchUTH** solves the unordered tree homeomorphism problem in

- $O(nm \log(w)/w)$ time
- $O(hm/w + m \log(w)/w)$ space
- $O(m \log(w))$ preprocessing time

m : the size of P , n : the size of T , h : the height of T , w : the word length

Note: This seems the first bit-parallel algorithm for UTH problem as far as we know, and It slightly improves the time complexity $O(nm^2)$ of the algorithm by [Gotz, Koch, Martens, DBPL'07] with a factor of $O(mw/\log(w))$



2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

53

Conclusion

- Tree matching with many-to-one mapping
 - UPTM**: unordered pseudo-tree matching
 - UTH**: unordered tree homeomorphism
- Bit-parallel algorithms for **UPTM** and **UTH** that run in
 - $O(nm \log(w)/w)$ time
 - $O(hm/w + m \log(w)/w)$ space
 - $O(m \log(w))$ preprocessing
- Future works
 - Extension of this technique for tree matching and inclusion with one-to-one mappings (seems difficult)
 - Applications to practical subclasses of XPath and XQuery languages

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

54

Thank you

2010/11/29 第2回 離散構造処理系シンポジウム：ビット並列計算を用いた高度なパターン照合 - 金田 悠作 (北海道大学)

55