Conservative String Covering of Indeterminate Strings

P. Antoniou[1], M. Crochemore[1], C. S. Iliopoulos[1], I. Jayasekera[1] and G. M. Landau[2]

[1] Department of Computer Science, King's College London
[2] Department of Computer Science, University of Haifa

August 30, 2008

- Introduction
- Finding constrained pattern p in indeterminate string x
- Computing λ-conservative covers of indeterminate strings
- Computing λ-conservative seeds of indeterminate strings

Definition

For a strings $x = uwv$:

- $|x|$ is the **length** of x
- ϵ is the **empty** string
- $x[i]$ is the i**-th symbol** of x
- w is a **substring** of x and x is a **superstring** of w
- $u(v)$ is a **prefix** (**suffix**) of x
- $x[i \dots j]$ denotes the **substring** of x starting at position i and ending at j

Definition

For strings $x = x[1 \ldots n]$ and $y = y[1 \ldots m]$:

- xy denotes the **concatenation** of strings x and y.
- x^k denotes the concatenation of k copies of x.
- If $x[n - i + 1 \ldots n] = y[1 \ldots i]$ for some $i \geq 1$, the string $x[1 \ldots n]y[i + 1 \ldots m]$ is a **superposition** of x and y. We also say that x overlaps y.

Definition

Indeterminate Strings and Conservative Indeterminate Strings

- An **indeterminate string** is a sequence $T = T[1]T[2]\ldots T[n]$, where $T[i] \subseteq \Sigma$ for each i and Σ is the alphabet.
- If at any position in an indeterminate string, $|T[i]| = 1$, we call this a **solid** symbol. However, when $|T[i]| > 1$, we call this a **non-solid** symbol.
- A **conservative indeterminate string** is an indeterminate string where its number of non-solid symbols if bounded by a constant k.

Definition

Covers and Conservative Covers

- A substring w of x is called a cover os x, if x can be constructed by concatenating or overlapping copies of w. We also say that w covers x.
- For example, if $x = ababaaba$, then aba and x are covers of x.
- A **conservative cover** is a cover with less indeterminate symbols than a given constant c.
- Conservative covers avoid results of covers of length one ($T[1] = \Sigma$).

As a building step we explain the constrained pattern matching problem in indeterminate strings, which can be defined as follows:

Definition

INPUT: A pattern, p, of length m, with at most κ non-solid symbols, where κ is constant and a text, t, of length n.

QUERY: Find all occurrences of pattern, p, in text, t.

Example

We consider a pattern, $p = A[CG]TA[AG]$ and text, $t = GA[CG][CT]AG[AT]A[AG][CT][AT]AG$. It can be seen from the figure below that p occurs in t starting at positions 2, 5, 8 and 9.

i	0	1	2	3	4	5	6	7	8	9	10	11	12
t	G	A	[CG]	[CT]	A	G	[AT]	A	[AG]	[CT]	[AT]	A	G
		A	[CG]	T	A	[AG]							
					A	[CG]	T	A	[AG]				
								A	[CG]	T	A	[AG]	
									A	[CG]	T	A	[AG]

The algorithm works in two steps:

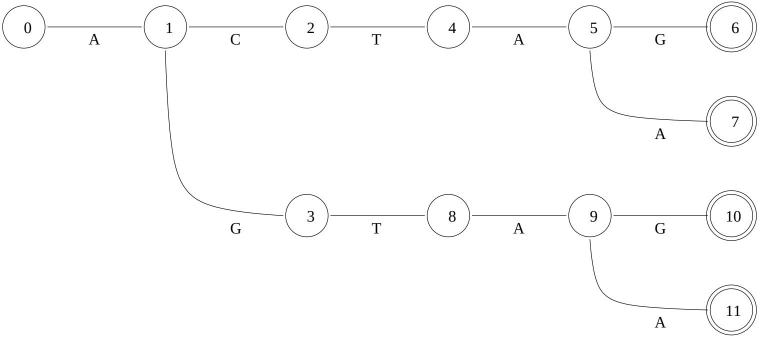
STEP 1:

- Let the pattern p be $p = P_1 P_2 \dots P_m$. We build the Aho-Corasick automaton for the dictionary of the prefixes of the pattern

$$D = \{\pi_1 \pi_2 \dots \pi_m, \forall \pi_i \in P_i, 1 \leq i \leq m\}$$

- Note that $|D| = \prod_{i=1}^{m} |P_i| < 2^{\kappa}$ as there are at most κ non-solid symbols.

- Aho-Corasick automaton for $p = A[CG]TA[AG]$:



i	0	1	2	3	4	5	6	7	8	9	10	11
f(i)	0	0	0	0	0	1	3	1	0	1	3	1

STEP 2:

- Assume that we have processed $T[1 \ldots i]$.
- We will now perform iteration $i + 1$.
- For each symbol τ occurring at $T[i + 1]$, we try to extend each prefix in P by that symbol τ, or we follow its failure link provided by the Aho-Corasick automaton.

i	0	1	2	3	4	5	6	
t	G	A	[CG]	[CT]	A	G	[AT]	...
P	0	{1}	{2,3}	{4,8}	{5,9}	{6, 10}	{8}	...

- Note that $|P|$ is bounded by the maximum number of possible prefixes, which in turn is bounded by the size of the automaton, therefore this is constant. Thus, this method is linear.

The λ-conservative cover problem is defined as follows:

Definition

INPUT: A conservative indeterminate text, t, of length n, a constant κ (which is the maximum number of non-solid symbols allowed in a cover) and an integer λ (which is the length of the cover).

QUERY: Is there a conservative cover of, c, of t, of length λ?

We now present a two step algorithm to this problem.

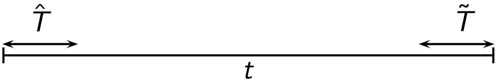
STEP 1:

- We consider the prefix, $\hat{T}$, of t of length λ,

$$\hat{T} = T_1 \ldots T_\lambda$$

and the suffix, $\tilde{T}$ of t of length λ,

$$\tilde{T} = T_{n-\lambda+1}, \ldots T_n$$



- The cover, c, covers the beginning and the end of T. Thus $\hat{T}$ and $\tilde{T}$ provide the set of potential candidates.

- We build the Aho-Corasick automaton for the dictionary

$$D = \{t_1 \dots t_\lambda \mid \forall\ t_i \in T_i \cap T_{i+n-\lambda},\ 1 \leq i \leq \lambda\}$$

STEP 2:

- For each $d \in D$ we find all of its occurrences in T, parsing the text T through the Aho-Corasick Automaton built in STEP 1.
- If a word d occurs at position i then we set a flag $L(i) = true$.
- If the distance $|i - j|$ of any two consecutive flags is less than λ, then we have a cover

$$C_1 C_2 \dots C_\lambda, \text{where}$$

$$C_i = \{d_i, \text{ is the } i-th \text{ letter of every word in } D, \ 1 \leq i \leq \lambda\}$$

- The overall complexity of the above two steps is linear.

The λ-conservative seed problem is defined as follows:

Definition

INPUT: An indeterminate text t, of length n, a constant κ (which is the maximum number of non-solid symbols allowed in a seed) and an integer λ (which is the length of the seed).

QUERY: Is there a conservative seed, s, of t, of length λ?

Again, we present a two step algorithm to solve this problem.

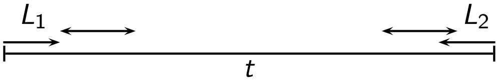
STEP 1:

- The first occurrence of the seed can be in any of the positions $\{1 \dots \lambda\}$. Thus we consider the following strings of length λ :

$$L_1 = \{T[1..\lambda], T[2..\lambda+1], \dots T[\lambda..2\lambda]\}$$

and all the suffixes of string t of length λ :

$$L_2 = \{T[n-\lambda..n], T[n-\lambda-1..n-1] \dots T[n-2\lambda]\}$$



- We build the Aho-Corasick automaton for the dictionary

  $D = \{t_{i_1} \ldots t_{i_\lambda} \mid \forall t_{i_j}, \mathrm{where\ t_{i_j}\ is\ the\ j-th\ symbol\ of\ T \in L_1 \cup L_2}\}.$

STEP 2:

- For each $d \in D$ we find all of its occurrences in T, parsing the text T through the Aho-Corasick Automaton built in STEP 1.
- If a word d occurs at position i then we set a flag $L_d(i) = true$.
- If the distance $|i - j|$ of any two consecutive flags in L_d is less than λ, then d is a candidate for a seed.
- Let i_1 and i_2 be the first and last occurrences of d in T. We check if $T[1, i_1]$ is a suffix of d and if $T[i_2, n]$ is a prefix of d, if that is the case then d is a suffix.
- The overall complexity is $O(\lambda n)$.

THANK YOU!