A derivation of the Knuth-Morris-Pratt pattern matching program
Cai Chengdian, [No Value]

Published: 01/01/1985

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 27. Sep. 2017
A Derivation of the Knuth-Morris-Pratt Pattern Matching Program

by

Cai Chengdian

EUT Report 85-WSK-02
ISSN 0167-9708
Coden: TEUEDE
A DERIVATION OF THE KNUTH-MORRIS-PRATT
PATTERN MATCHING PROGRAM

by

Cai Chengdian

Department of Computer Science
Jinan University
Guangzhou, Guangdong
China

Department of Mathematics and Computing Science
Eindhoven University of Technology
5600 MB Eindhoven
The Netherlands

Abstract

The purpose of this note is to produce a formal derivation of the Knuth-Morris-Pratt pattern matching program.

AMS Classification: 68 B 10, 68 C 05, 68 E 99.
Introduction

Pattern matching is one of the fundamental operations on strings. Several algorithms that solve this problem have been developed. Among these, Knuth-Morris-Pratt's one ([1]) is well-known and has the advantage of time complexity linear in the length of the text and storage requirements linear in the length of the pattern. But it requires some complicated processing on the pattern that is difficult to understand and this has limited the extent to which the program is used ([2]). In [1] the program is introduced by a play-by-play description for an example, and the formal correctness of the program is not given in detail. In [3] a formal treatment for this program is given, but the processing on the pattern is not contained in detail in the text either, and is somewhat different from [1]. In this note we shall apply formal techniques that are given by [3] to produce a proof and a program for the pattern matching problem. The processing on the pattern is essentially the same as [1].

Notational interlude and a property

The expression

\[(\text{MIN } i : D(i) : E(i))\]

denotes the minimum value of \(E(i)\) for all \(i\) satisfying \(D(i)\).

For \(r \leq s\), \(x(r, s)\) denotes the sequence \(x(r), x(r + 1), \ldots, x(s - 1)\) of \(s - r\) elements.

For two sequences \(x(s, s + k)\) and \(y(r, r + k)\),

\[x(s, s + k) = y(r, r + k)\]
denotes

\[(\forall i: 0 \leq i < k: x(s + i) = y(r + i))\]

and

\[x(s, s + k) \neq y(r, r + k)\]

denotes

\[(\exists i: 0 \leq i < k: x(s + i) \neq y(r + i))\]

From the above, we have for \(k > 0\) the following property:

\[0 < q \leq k \land x(s, s + k) = y(r, r + k)\]

\[\implies x(s + q, s + k) = y(r + q, r + k)\]

The notations for proof structures and programs have been adapted from [3], [4] and [5].

The development of the pattern matching program

For two integer sequences \(p(0, M)\) and \(t(0, N)\) \((M > 1\) and \(N \gg M)\), a program is requested to determine whether \(p\) occurs as a continuous subsequence of \(t\) and, furthermore, if so, to find the position of the first occurrence of \(p\) in \(t\).

Formally, our program has to establish

\[R1 \lor R0\]

where

\[R1: (\forall u:\neg\text{match}(u))\]

\[R0: k = (\text{MIN } u: \text{match}(u): u)\]

and

\[\text{match}(u) = (0 \leq u \leq N - M) \land (t(u, u + M) = p(0, M))\]
To this end, we introduce the invariant

\[ P_0 \land P_1, \]

where

\[ P_0: t(k,k+j) = p(0,j) \land 0 \leq j \leq M \land 0 \leq k \leq N - j, \]

and

\[ P_1: (\forall u: 0 \leq u < k: \overline{\text{match}(u)}). \]

\( P_0 \) and \( P_1 \) derive their importance from

\[ P_0 \land P_1 \land j = M \Rightarrow R_0, \]

and

\[ P_1 \land k > N - M \Rightarrow R_1. \]

\( P_0 \land P_1 \) is trivially established by "\( k, j := 0, 0 \)" and we can sketch our program as follows:

\[
\begin{array}{l}
[k, j: \text{int} \\
; p(i: 0 \leq i < M), t(i: 0 \leq i < N): \text{array of int} \\
; k, j := 0, 0 \{P_0 \land P_1\} \\
; \text{do } j \neq M \land k \leq N - M \rightarrow \\
\quad \text{"increase the bound function } 2k + j \text{ of } j \text{ and } k \text{ under} \\
\quad \text{invariance of } P_0 \land P_1" \\
\quad \text{od } \{P_0 \land P_1 \land (j = M \lor k > N - M), \text{ hence } R_1 \text{ cor } R_0\}
\end{array}
\]

Now we investigate increases of \( j \) or \( k \) that cause an increase of the bound function.

Because \( j \) does not occur in \( P_1 \), for an increase of \( j \) we need to consider \( P_0 \) only. For the maintenance of \( P_0 \), apparently, at each iteration an increase
of j by one is adequate. And we derive

\[ P_{j+1}^{0} = t(k,k+j+1) = p(0,j+1) \land 0 \leq j + 1 \leq M \land 0 \leq k \leq N - (j + 1) \]

\[ = P_{0} \land t(k+j) = p(j) \land j \neq M \land k \leq N - M. \]

If B holds, where

\[ B: t(k+j) \neq p(j), \]

we know that if k were kept constant, an increase of j would destroy \( P_{0} \), thus we increase k. Because of

\[ B \Rightarrow \neg \text{match}(k) \] (1)

an increase of k by 1 maintains \( P_{1} \) under B, and it trivially maintains \( P_{0} \) for \( j = 0 \).

As a consequence, "increase the bound function 2* \( k + j \) of j and k under invariance of \( P_{0} \land P_{1} \)" can be refined as the following alternative construct:

\[
\begin{align*}
\text{if } & t(k+j) = p(j) \Rightarrow j := j + 1 \{P_{0} \land P_{1}\} \\
& j = 0 \land t(k+j) \neq p(j) \Rightarrow k := k + 1 \{P_{0} \land P_{1}\} \\
& j \neq 0 \land t(k+j) \neq p(j) \Rightarrow ? \\
\text{fi}.
\end{align*}
\]

(2)

But for \( j \neq 0 \land B \), \( P_{0} \) possibly allows a further increase of k in view of what follows.

For \( 0 < i \leq j \),

\[ t(k,k+j) = p(0,j) \]

\[ \Rightarrow \{(0)\} \]

\[ t(k+i,k+j) = p(i,j). \] (3)
For $j \neq 0$, the equation
\[
i: p(i,j) = p(0,j-i) \land 0 < i \leq j
\] (4)
has at least one solution, e.g. $i = j$.
Thanks to (3), we have for each solution $i$ of (4)
\[
\text{P0 } \Rightarrow \text{ t(k+i,k+j) } = \text{p}(0,j-i) \land 0 \leq j-i < j \land 0 \leq k+i \leq N-(j-i).
\]
Therefore, P0 is maintained by
\[
"k,j := k+i,j-i"
\] (5)
where $i$ is any solution of (4).
On the other hand, if $i$ satisfies
\[
p(i,j) \neq p(0,j-i) \land 0 < i \leq j
\]
we have, on account of (3),
\[
P0 \Rightarrow \text{ t(k+i,k+j) } \neq \text{p}(0,j-i)
\]
\[
\text{hence } \quad P0 \Rightarrow \neg \text{match}(k+i).
\] (6)
For $j \neq 0$, let
\[
f(j) = (\text{MIN } i: p(i,j) = p(0,j-i) \land 0 < i \leq j; i).
\]
We conclude that
\[
"k,j := k+f(j),j-f(j)"
\] (7)
maintains $\text{P0 } \land \text{P1}$ under B:
(i) thanks to (5), (7) maintains P0, because $f(j)$ is a solution of (4);
(ii) in view of (1) and (6)

\[(A \ i: \ 0 \leq i < f(j): \ P0 \land B \Rightarrow \neg \text{match}(k+1))\]

holds, so that (7) maintains P1 under B.

Consequently, the question mark in the alternative construct (2) can be replaced by (7), and we have developed our program.

The repetition in the program terminates, since \(2^k + j\) is bounded from above by \(2^N + M\) and increases by at least 1 at each iteration.

The computation of \(f(1,M)\)

The only problem left is to obtain an \(f(1,M)\) satisfying

\[Rf: (A \ v: \ 0 < v < M: f(v) = (\min i: D(i,v): i))\]

where

\[D(i,v): p(i,v) = p(0,v-i) \land 0 < i \leq v.\]

In the standard fashion we derive from \(Rf\) an invariant \(Pf\) by replacing the constant upper bound \(M-1\) by a suitably bounded variable \(k\):

\[Pf: (A \ v: \ 0 < v \leq k: f(v) = (\min i: D(i,v): i)) \land \]

\[1 \leq k \leq M-1.\]

"\(k := 1; f:(k) = 1\)" establishes \(Pf\), and we can sketch the computation of \(f(1,M)\) as follows:
The repetition in the sketch trivially terminates.

Now we refine "compute $f(k+1)". From the definition of $f(k+1)$, i.e.

$$f(k+1) = \text{MIN } i : D(i,k+1) : i$$

and

$$(i \leq k \land D(i,k+1)) \Rightarrow D(i,k)$$

we conclude

$$D(f(k+1),k) \lor f(k+1) = k + 1 . \quad (8)$$

$f(k)$ is the minimum solution of the equation

$$i : D(i,k) , \quad (9)$$

hence,

$$f(k) \leq f(k+1) \leq k + 1 . \quad (10)$$

In order to compute $f(k+1)$, it consequently suffices to search the solutions of (9), starting with $f(k)$, in increasing order in the light of the linear search theorem ([3]).
We observe that the largest solution of (9) is \( k \). Thus, the search has the form

\[
\begin{align*}
  j & := f(k) \\
  \text{do } j \neq k \land \neg D(j,k+1) \rightarrow \\
  & j := F(j) \text{ od ,}
\end{align*}
\]

where \( F(j) \) is the minimum solution larger than \( j \) of (9).

The search terminates, since \( j < F(j) \leq k \). And

\[ Q: D(j,k) \land j < f(k+1) \]

is an invariant of the search. The reason is:

(i) \( Q \) is established by "\( j := f(k) \)"; on account of (10) and \( D(f(k),k) \);

(ii) \( D(F(j),k) \) holds according to the definition of \( F(j) \);

(iii) \( j \leq f(k+1) \land \neg D(j,k+1) \Rightarrow \{(8) \text{ and the definition of } F(j)\} \\
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad F(j) \leq f(k+1). \)

Considering \( D(j,k) \land p(k) = p(k-j) \Rightarrow D(j,k+1) \), we obtain the following program to compute \( f(k+1) \):

\[
\begin{align*}
  & | [ j: \text{int} \\
  & \quad ; j := f(k) \{Q\} \\
  & \quad ; \text{do } j \neq k \land p(k) \neq p(k-j) \rightarrow \\
  & \quad \quad j := F(j) \text{ od } \{Q \land (j = k \lor p(k) = p(k-j))\} \\
  & \quad ; \text{if } p(k) = p(k-j) \Rightarrow f:(k+1) = j \{j \leq f(k+1) \land D(j,k+1), \\
  & \quad \quad \text{ hence } f(k+1) = j\} \\
  & \quad [ p(k) \neq p(k-j) \Rightarrow f:(k+1) = k + 1 \{k \leq f(k+1) \land \neg D(k,k+1), \\
  & \quad \quad \text{ hence } f(k+1) = k + 1\} \\
  & \quad \text{fi} \\
  & ] | . \quad (11)
\end{align*}
\]
Finally, by using
\[ p(j,k) = p(0,k-j) \land 0 < i \leq k-j \]
\[ \Rightarrow \{0\} \]
\[ p(j+i,k) = p(i,k-j) , \] (12)

\( F(j) \) can be refined as follows:

\[
F(j) = \{\text{the definition of } F(j)\} \\
= (\text{MIN } i: p(i,k) = p(0,k-i) \land j < i \leq k: i) \\
= j + (\text{MIN } i: p(j+i,k) = p(0,k-j-i) \land 0 < i \leq k-j: i) \\
= \{j \text{ is a solution of (9), hence (12)}\} \\
= j + (\text{MIN } i: D(i,k-j): i) \\
= \{\text{the definition of } f(k-j)\} \\
= j + f(k-j) .
\]

Thanks to \( 0 < j \leq k \land j \neq k \) and \( Pf, f(k-j) \) is defined. Therefore, in (11) "\( j := F(j) \)" can be replaced by "\( j := j + f(k-j) \)", and we have completed the computation of \( f(1,M) \).

**Acknowledgement**

I am greatly indebted to A.J.M. van Gasteren for her instructive comments and several improvements. I would like to thank Edsger W. Dijkstra, W.H.J. Feijen, J.G. Wiltink and the members of the Tuesday Afternoon Club for many valuable suggestions.
References


