JURNAL MEDIA APLIKOM
ISSN : 2086 - 972X
Vol. 1, No. 3, September 2010

APLIKASI PENCOCOKAN STRING DENGAN METODE *FINITE AUTOMATA*

Agus Priyanto
Sekolah Tinggi Ilmu Komputer Yos Sudarso Purwokerto

ABSTRAK

Permasalahan pencocokan string semakin bertambah seiring dengan berjalannya waktu, bahkan permasalahannya berasal dari bidang-bidang yang berbeda. Seperti misalnya dalam pencarian string dalam kosakata bahasa indonesia. Dalam kehidupan sehari-hari kita sering mengalami kesulitan dalam pemeriksaan kosakata yang tepat dalam Bahasa Indonesia.

Pemeriksaan Kosakata adalah sebuah masalah yang kerap kali kita temui, yaitu apabila kita berhubungan dengan suatu text atau interpreter, mulai dari permasalahan sederhana sampai dengan permasalahan yang komplex. Contohnya adalah proses pencarian suatu kata dari sebuah dokumen text. Mengingat betapa pentingnya keberadaan proses pemeriksaan kosakata, maka berbagai penelitian dan pengembangan dilakukan oleh berbagai pihak, dengan menggunakan berbagai metode untuk menemukan algoritma yang seefisien mungkin.

Kata kunci: *string, finite automata, pencocokan string, pola.*

1. PENDAHULUAN

Dalam kehidupan sehari-hari kita sering mengalami kesulitan dalam pemeriksaan kosakata yang tepat dalam Bahasa Indonesia. Pemeriksaan Kosakata adalah sebuah masalah yang kerap kali kita temui, yaitu apabila kita berhubungan dengan suatu *text* atau *interpreter*, mulai dari permasalahan sederhana sampai dengan permasalahan yang komplex. Contohnya adalah proses pencarian suatu kata dari sebuah dokumen *text*.

Mengingat betapa pentingnya keberadaan proses pemeriksaan kosakata, maka berbagai penelitian dan pengembangan dilakukan oleh berbagai pihak, dengan menggunakan berbagai metode untuk menemukan algoritma yang seefisien mungkin. Dan bukanlah tidak mungkin, bahwa saat ini sudah ditemukan algoritma lain yang berbeda, dengan sudut pandang yang berbeda untuk mengatasi masalah Pemeriksaan Kosakata ini.

Dalam Matematika Diskret terdapat satu cabang ilmu yang khusus mempelajari tentang bahasa, yaitu Teori Bahasa Formal. Bahasa yang dibahas pada Teori Bahasa Formal adalah bahasa tulisan. Penelitian ini ditujukan pada Pencarian Kosakata pada bahasa secara tertulis. Keuntungan dari penggunaan Teori Bahasa Formal adalah kita bisa memodelkan Pemeriksaan Kosakata menggunakan Finite State Automata, yaitu suatu mesin abstrak yang dapat mengenali bahasa.

2. TINJAUAN PUSTAKA

a. Teori Bahasa Formal

Teori bahasa membicarakan bahasa formal (*formal language*), terutama untuk kepentingan perancangan kompilator (*compiler*) dan pemroses naskah (*text processor*). Bahasa formal adalah kumpulan *kalimat*. Semua kalimat dalam sebuah bahasa dibangkitkan oleh sebuah tata bahasa (*grammar*) yang sama. Sebuah bahasa formal bisa dibangkitkan oleh dua atau lebih tata bahasa berbeda. Dikatakan bahasa formal karena grammar diciptakan mendahului pembangkitan setiap kalimatnya. Bahasa manusia bersifat sebaliknya, grammar diciptakan untuk meresmikan kata-kata yang digunakan di masyarakat.

b. Teori Automata

Berasal dari bahasa Yunani *automatos*, yang berarti sesuatu yang bekerja secara otomatis (mesin). Dalam tulisan ini akan dipergunakan istilah *automaton* sebagai bentuk tunggal dan *automata* sebagai bentuk jamak. Teori Automata adalah teori tentang mesin abstrak yang :

- Bekerja Sekuensial
- Menerima Input
- Mengeluarkan Output

Pengertian mesin diatas, bukan hanya mesin elektronis/mekanis saja melainkan segala sesuatu (termasuk perangkat lunak) yang memenuhi ketiga ciri di atas. Penggunaan automata pada perangkat lunak terutama pada pembuatan compiler bahasa pemrograman. Secara garis besar ada dua fungsi *automata* dalam hubungannya dengan bahasa, yaitu :

- Fungsi *automata* sebagai pengenal (*RECOGNIZER*) string-string dari suatu bahasa, dalam hal ini bahasa sebagai masukan dari *automata*
- Fungsi *automata* sebagai pembangkit (*GENERATOR*) string-string dari suatu bahasa, dalam hal ini bahasa sebagai keluaran dari *automata*

c. Klasifikasi Bahasa Menurut *Chomsky*

Untuk menyelesaikan suatu masalah, mula-mula harus dikenali dulu dengan baik masalah yang sebenarnya dihadapi. Salah satu caranya adalah dengan mengklasifikasikan masalah tersebut. Dengan demikian dapat dikenali ciri-ciri masalah tersebut, dan dapat lebih difokuskan penyelesaian masalah tersebut pada ciri-ciri yang berhasil dikenali. Pada masalah bahasa, klasifikasi tersebut sudah pernah dilakukan. *Chomsky* membagi bahasa menjadi 4 kelas berdasarkan tata bahasanya. Keempat kelas tersebut adalah :

1. Regular Grammar/Regular Language (RG/RL)

Mesin Pengenal : Finite State Automata (FSA)

Dapat dibagi menjadi dua subkelas :

a. *Left Linear Grammar/Left Linear Language* (LLG/LLL), jika P berbentuk $A \rightarrow Bx|x$

b. *Right Linear Grammar/Right Linear Language* (RLG/RLL), jika P berbentuk $A \rightarrow xB|x$, di mana $A,B \square N$ dan $x \square T^*$

2. Context Free Grammar/Context Free Language (CFG/CFL)

Mesin Pengenal : Push Down Automata (PDA)

Ciri-ciri : bentuk produksi *P* $A \rightarrow \beta$, di mana A | N dan $\beta \square (T \cup N)^*$

3. Context Sensitive Grammar/Context Sensitive Language (CSG/CSL)

Mesin Pengenal : Linear Bounded Automata (LBA)

Ciri-ciri : bentuk produksi *P* $\alpha \rightarrow \beta$, $\alpha,\beta \mid (T \cup N)+$, $|\alpha| \leq |\beta|$

4. Unrestricted Grammar/Unrestricted Language (UG/UL)

Mesin Pengenal : Turing Machine ™

Ciri-ciri : bentuk produksi *P* $\alpha \rightarrow \beta$, $\alpha,\beta \mid (T \cup N)+$

d. FSA (Finite State Automata)

Setiap jenis *automata* mempunyai keunikan yang membuatnya berbeda fungsinya dengan *automata* yang lain. Berikut ini akan dibahas sifat-sifat FSA :

- Pita masukan hanya bisa dibaca, berisi string yang berasal dari suatu abjad.
- Setelah membaca satu simbol pada pita, kepala pita akan maju ke posisi symbol berikutnya.
- Kepala pita tidak bisa mundur.
- Mempunyai sejumlah berhingga status, setiap saat FSA berada pada status tertentu.

Setiap FSA bisa diasosiasikan dengan sebuah diagram transisi, yaitu suatu *graf* berarah sebagai berikut :

- Setiap simpulnya mewakili setiap status pada FSA.

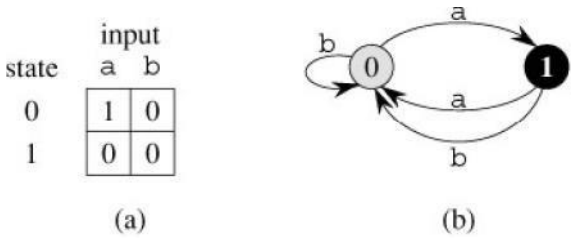
- Jika ada transisi dari status *p* ke status *q* pada input *a*, maka ada busur dari *p* ke *q* berlabel *a*.
- Status awal ditandai dengan kata START, status akhir ditandai dengan 2 lingkaran.

Jadi fungsi dari diagram transisi adalah untuk menggambarkan cara kerja suatu FSA. Finite Automata digunakan untuk memutuskan apakah sebuah kata atau string adalah kata yang diterima oleh mesin automata tersebut. Sebuah aoutomata memiliki status dan fungsi transisi yang merubah status berdasarkan status saat ini dan karakter yang dibaca saat ini. Ada yang disebut sebagai status akhir dan status ini mungkin lebih dari satu. Bila string itu dijalankan dan berakhir disalah satu status akhir maka string itu diterima atrau termasuk dalam behasa tersebut.

Sebuah finite *automaton* M adalah sebuah 5-tuple (Q,q0,A, ∑, β) dimana

- Q adalah himpunan *state*
- q0 adalah *start state*
- A adalah himpunan *state* yang diterima
- ∑adalah himpunan alfabet masukan
- β adalah fungsi transisi dari M

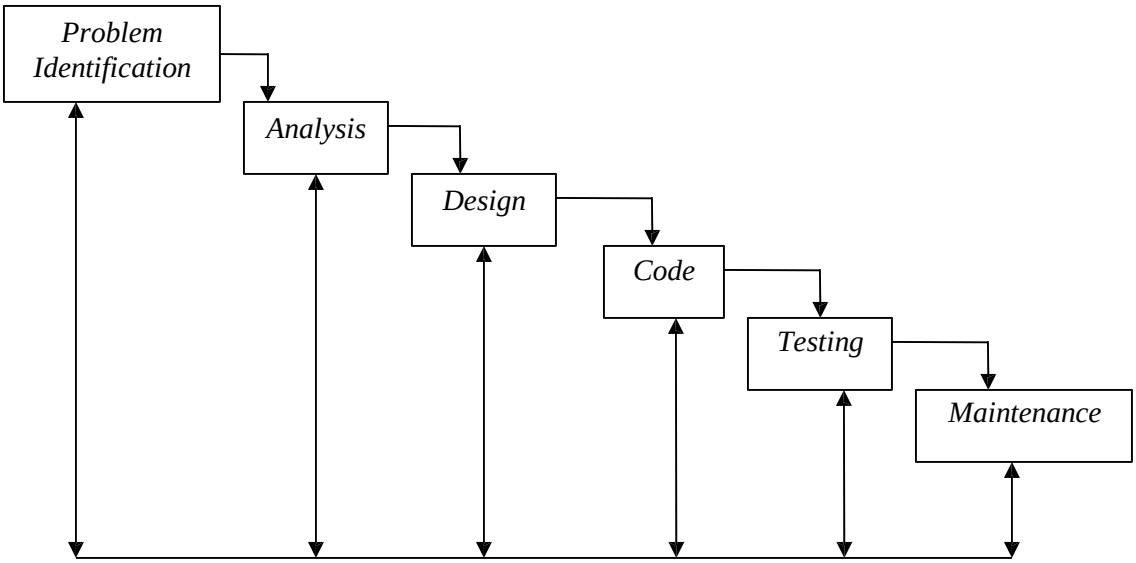
Finite automaton mulai dari *state* q0 dan membaca karakter-karakter dari string masukan secara berurutan. Bila *automaton* pada *state* q dan membaca karakter masukan a, *automaton* tersebut akan berpindah dari *state* q ke *state* β(q,a). Ketika *state* sekarang q adalah anggota A, mesin M bisa menerima string yang dibaca.



Gambar 2.1. *Finite automaton* sederhana dengan himpunan *state* Q = {0,1}, *state* awal q0 = 0, dan input alphabet = {a,b}. (a) representasi tabel dari fungsi transisi β (b) sebuah diagram transisi.

3. JALANNYA PENELITIAN

Membangun sebuah system atau aplikasi adalah menganalisa, mendesain dan memanajemen setiap faktor teknik yang berguna untuk memecahkan suatu masalah. Pengembangan perangkat lunak diasumsikan sebagai sebuah masalah yang terus berputar dan selalu menuntut pada para pengembang sistem agar mempunyai suatu strategi, dengan tujuan tersebut maka membuat sebuah model proses merupakan suatu keharusan agar dalam melakukan pengerjaan suatu perangkat lebih mudah dalam mengerjakan setiap prosesnya. Model proses untuk seorang pengembang sistem dibuat berdasarkan kebutuhan dari suatu produk aplikasi. Gambar 2.10 merupakan salah satu contoh model proses yang digunakan pengembang perangkat lunak yang meliputi proses awal sampai akhir.



Gambar 2.4 *Waterfall Model*

a. Identifikasi Masalah (*Problem Identification*)
Pengidentifikasian masalah merupakan dasar dalam menentukan langkah berikutnya yang akan diambil, dalam langkah ini dilakukan dengan pengumpulan data dari pengguna, sehingga permasalahan yang mungkin dihadapi pada tahap berikutnya akan diperkecil dengan memperoleh data yang lengkap.

b. Analisa Kebutuhan (*Analysis*)
Analisa data dilakukan untuk dibuat rancangan perangkat lunak agar sesuai dengan kebutuhan, sehingga perangkat lunak yang dibuat bisa sesuai dengan apa yang diharapkan.

c. Perancangan (*Design*)
Perancangan perangkat lunak merupakan pekerjaan bertahap yang sangat difokuskan pada elemen dari program yaitu diantaranya struktur data, arsitektur perangkat lunak, detail prosedur dan karakteristik dari antar muka (*Interface*), sehingga perangkat lunak yang dibuat bisa sesuai dengan permintaan pengguna.

d. Pengkodean (*Coding*)
Desain yang dibuat kemudian diterjemahkan dengan bahasa pemrograman yang akan digunakan. Saat ini banyak pilihan bahasa pemrograman dan setiap bahasa pemrograman tersebut memiliki kelebihan maupun kekurangan masing-masing, dan mempunyai spesifikasi penggunaan yang berbeda-beda pula.

e. Pengujian program (*Testing)*
Setelah proses pengkodean selesai dilaksanakan kemudian dilanjutkan dengan pengujian program. Langkah ini berfungsi

untuk menguji logika internal dari perangkat lunak, untuk memastikan setiap perintah telah benar atau setidaknya meminimalisir kesalahan sistem.

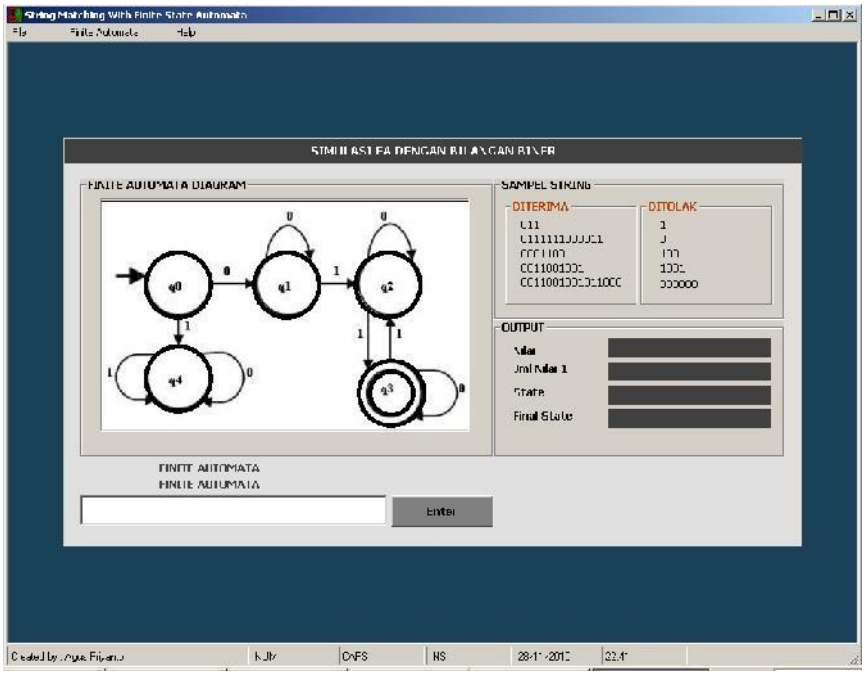
f. Peningkatan kualitas (*Maintenance*)

Langkah ini merupakan langkah terakhir dalam proses pembuatan perangkat lunak, hal ini dilakukan agar perangkat lunak yang dibuat menjadi lebih baik dari hasil sebelumnya baik dari tampilannya maupun kecepatan aksesnya.

4. HASIL DAN PEMBAHASAN

a. Menu Utama

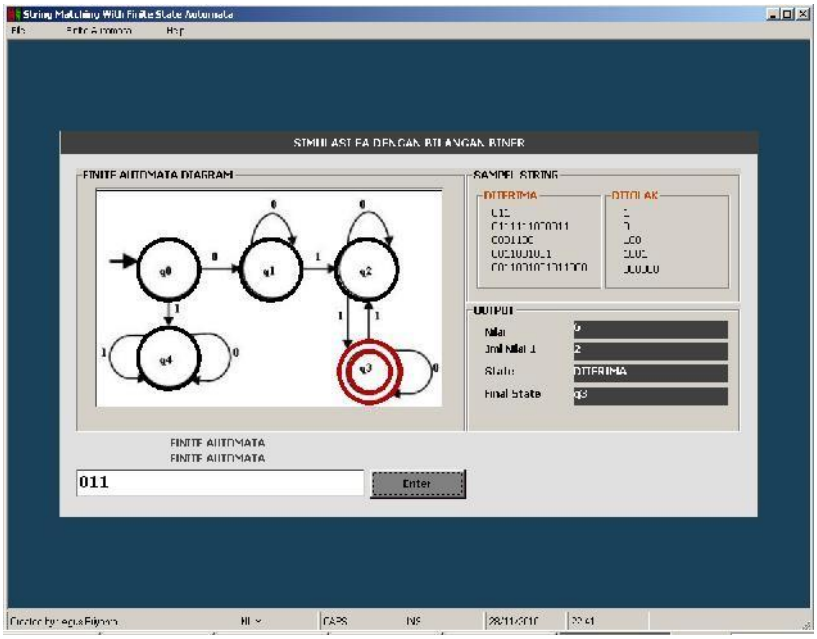
Setelah proses perancangan dilakukan maka dihasilkan sebuah aplikasi sistem pemeriksaan kosakata yang siap untuk digunakan. *Form* utama merupakan halaman awal aplikasi pemeriksaan kosakata yang terdiri atas berbagai macam menu yang digunakan untuk membuka *form* lain yang berkaitan. Selain tombol menu, dalam halaman utama juga terdiri atas nama aplikasi, nama pembuat, waktu dan tanggal. Tampilan halaman utama aplikasi Pemeriksaan Kosakata menggunakan Finite State Automata diperlihatkan pada gambar dibawah ini.



Gambar 4.1 *Form Menu Utama*

b. Simulasi FA

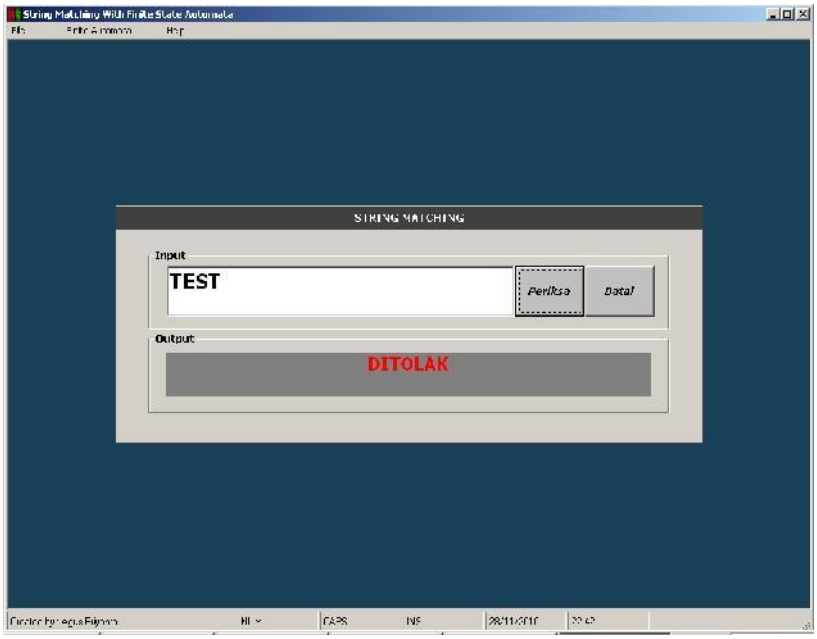
Form Simulasi FA ini digunakan untuk melakukan simulasi bilangan biner yang akan diperiksa menggunakan konsep *Finite State Automata*. Jika bilangan biner yang diperiksa ditolak atau diterima oleh mesin maka aplikasi akan memunculkan output yang diterima atau ditolak oleh mesin.



Gambar 4.2 *Form* Simulasi FA

c. Pencocokan String

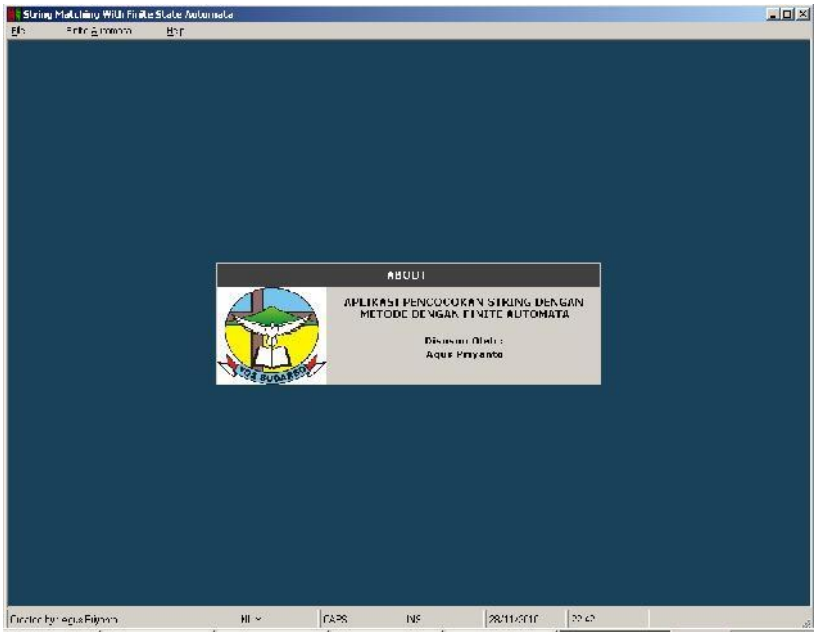
Form String Matching digunakan untuk melakukan pengecekan kata, *user* hanya memasukan kata dalam Bahasa Indonesia kemudian kata tersebut akan diperiksa dengan menggunakan mesin *automata*.



Gambar 4.3 *Form String Matching*

d. About

Form About digunakan untuk menampilkan biodata dari pembuat aplikasi ini.



Gambar 4.4 *Form About*

5. KESIMPULAN

Kesimpulan yang diperoleh setelah melakukan perancangan, pembuatan dan pengujian Aplikasi Pencocokan String Menggunakan Prinsip Finite State Automata yaitu:

a. String yang dimasukan kedalam Aplikasi dapat diterima oleh mesin automata dengan menggunakan Prinsip Finite State Automata. Setiap inputan string tersebut dapat berakhir pada Final State.
b. Algoritma pencarian *string* merupakan salah satu komponen penting dalam pemrosesan *string* dan juga digunakan sebagai dasar dari berbagai aplikasi perangkat lunak khususnya sebagai bentuk dasar transfer data. Algoritma pencarian *string* bekerja dengan menghasilkan posisi tempat terjadinya pengulangan suatu pola *string* dalam *string* yang lebih panjang.

6. DAFTAR PUSTAKA

[1] Kusumo, S.A., 2000. *Microsoft Visual Basic 6.0*. PT. Elex Media Komputindo, Jakarta.
[2] Jogiyanto, H. M., 2005. *Analisis dan Desain.* Andi Offset, Yogyakarta.
[3] English Wikipedia 2009 http://en.wikipedia.org/wiki/Stri ng_searching_algorithm Tanggal akses : 28 Maret 2010 pukul : 22.00
[4] String Matching with Finite Automata http://www.math.uic.edu/~leon/ cs-mcs401- s08/handouts/finite- automata.pdf Tanggal akses : 28 Maret 2010 pukul : 22.30
[5] Munir, Rinaldi, "Diktat Kuliah IF2251 Strategi Algoritmik", 2006, Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung,186.