# Penggunaan Induksi Matematika untuk Mengubah Deterministic Finite Automata Menjadi Ekspresi Reguler

Husni Munaya - 13513022[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]husni.munaya@students.itb.ac.id*

***Abstrak*—Ada banyak cara untuk menyelesaikan suatu permasalahan matematika yang berhubungan dengan persoalan pembuktian, salah satunya adalah induksi matematika. Induksi matematika merupakan metode yang digunakan untuk membuktikan kebenaran dari suatu proposisi yang berhubungan dengan bilangan bulat. Pada teori automata, metode ini sering digunakan untuk membuktikan proposisi yang ada. Pada makalah ini, akan dijelaskan bagaimana cara mengubah deterministic finite automata menjadi ekspresi reguler dengan bantuan induksi matematika.**

***Kata kunci*—DFA, ekspresi reguler , induksi matematika**

## I. PENDAHULUAN

*Finite automata* merupakan suatu mesin abstrak dalam bentuk model matematika yang bertugas untuk menerima masukan berupa himpunan simbol, dan kemudian menentukan apakah masukan itu terdapat pada bahasa yang didefinisikan oleh *finite automata* tersebut. Bahasa yang dapat didefinisikan oleh *finite automata* disebut bahasa reguler. *Finite Automata* sangat berguna untuk memodelkan sebuah perangkat keras maupun perangkat lunak. Salah satu aplikasinya pada perangkat lunak adalah *lexical analyzer*, salah satu komponen kompilator yang bertugas untuk mengubah kumpulan karakter menjadi sebuah token.

| Karakter | Token |
|---|---|
| + | Operasi Penjumlahan |
| = | Operasi penugasan |
| / | Operasi pembagian |
| 8 | Bilangan bulat |
| > | Operator lebih dari |
| sum | *identifier* |
| ; | Akhir dari *statement*. |

Tabel 1.1: Lexical anlyzer, bertugas untuk mengubah kumpulan karakter menjadi token.

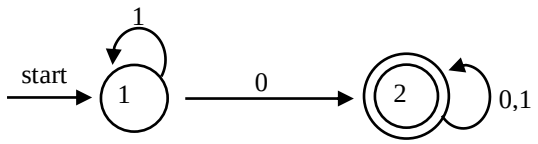
### A. Deterministic Finite Automata

*Deterministic finite automata* merupakan salah satu klasifikasi dari *finite automata*. Dikatakan deterministik karena pada setiap simbol masukan hanya ada satu state yang dapat dituju dari state sebelumnya. *Deterministic finite automata*, yang seterusnya akan kita singkat menjadi DFA, terdiri atas lima komponen $(Q,\sum,\delta,q_0,F)$, yang dalam hal ini:

- $Q$ adalah himpunan state berhingga.
- $\sum$ adalah himpunan simbol masukan berhingga.
- $\delta$ adalah fungsi transisi. Fungsi ini menerima state dan simbol masukan sebagai argumen dan menghasilkan sebuah state.
- $q_0$ adalah state awal, merupakan anggota $Q$.
- $F$ adalah state akhir. F merupakan himpunan bagian dari $Q$.

Mendeskripsikan DFA dengan notasi seperti di atas terkadang menyulitkan. Ada dua cara yang lebih mudah untuk digunakan, yaitu

1. Diagram transisi, yaitu sebuah graf seperti pada gambar 1.1.
2. Tabel transisi, seperti pada tabel 1.2.




Gambar 1.1: DFA (diagram transisi) yang menerima *string* yang mempunyai setidaknya satu simbol 0.

| State | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $q_0$ | $q_1$ |
| $*q_1$ | $q_1$ | $q_1$ |

Tabel 1.2: DFA (tabel transisi) yang menerima *string* yang mempunyai setidaknya satu simbol 0

B. Ekspresi Reguler

Selain mendefinisikan bahasa dengan cara membuat sekumpulan state, seperti yang dilakukan oleh DFA, ada cara lain untuk mendefinisikan suatu bahasa, yaitu ekspresi reguler. Ekspresi reguler merupakan cara aljabar untuk mendeskripsikan suatu bahasa. Contohnya adalah 10* + 01*, yaitu ekspresi yang mendefinisikan bahasa yang terdiri dari simbol 1 yang diikuti oleh sejumlah simbol 0 atau simbol 0 yang diikuti oleh sejumlah simbol 1. Ekspresi reguler sangat berperan penting pada dunia komputasi, ekspresi tersebut digunakan pada beberapa mesin pencari dan perangkat lunak pengolah kata. Ekspresi reguler juga merupakan suatu fitur penting yang terdapat dalam beberapa bahasa pemrograman, seperti Java, Python dan C++. Ekspresi reguler mampu untuk mendefinisikan bahasa yang dideskripsikan oleh *finite automata*, yaitu bahasa reguler. Ekspresi reguler memiliki beberapa operasi, yaitu

1. Union bahasa *L* dan *M*, dilambangkan oleh $L \cup M$. Contohnya, Jika L = {0,01,001} dan M = {10,11}, Maka $L \cup M$= {0,01,10,11,001}.

2. Konkatenasi dari bahasa *L* dan *M*, dilambangkan oleh $L \cdot M$, atau dapat juga ditulis *LM*. Contohnya, Jika L = {1,11} dan M = {10,01}, maka LM = {110,101,1110,1101}.

3. Klosur dari bahasa *L*, dilambangkan dengan *L**, merepresentasikan bahasa dari himpunan *string* yang dapat dibentuk dengan mengambil anggota himpunan yang ada pada L, dan menyambungnya. Contohnya, jika L = {0,1}, maka L* adalah semua *string* yang mempunyai simbol 0 dan 1. Jika L={0,11}, maka L* adalah semua *string* yang mempunyai simbol 1 berpasangan. Misalna 11,011,0110.

Dari ketiga operator tersebut, operator klosur memiliki tingkat prioritas yang paling utama, prioritas selanjutnya adalah operator konkatenasi, dan yang terakhir operator union.

## II. DASAR TEORI

### *2.1 Definisi Induksi Matematika*

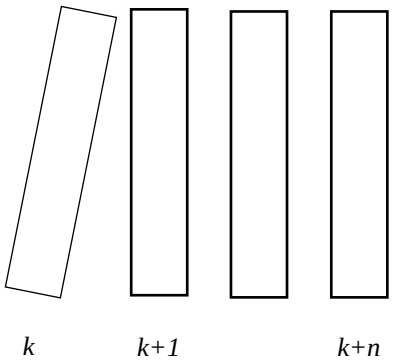
Induksi matematika merupakan metode pembuktian yang digunakan untuk membuktikan proposisi yang berhubungan dengan bilangan bulat. Dengan menggunakan induksi matematika, kita dapat mengurangi langkah-langkah pembuktian bahwa semua bilangan bulat termasuk ke dalam suatu himpunan kebenaran dengan hanya sejumlah langkah terbatas.

### *2.2 Pembuktian dengan Induksi Matematika*

Induksi matematika dapat digunakan untuk membuktikan pernyataan mengenai bilangan bulat n. Misal P(n) adalah suatu pernyataan mengenai bilangan bulat. Kita ingin membuktikan bahwa pernyataan P(n) berlaku untuk semua bilangan bulat positif n. Terdapat dua langkah untuk menyelesaikan permasalahan di atas, yaitu:

1. Basis
   Pada tahap ini, kita akan membuktikan bahwa P(n) bernilai benar untuk n=*i*, yang pada hal ini *i* adalah basis. Untuk basis, biasanya digunakan *i*=0 atau *i*=1, tergantung pada permasalahan yang dihadapi.
2. Langkah Induksi
   Langkah induksi menunjukan jika P(n) bernilai benar, maka P(n+1) juga benar untuk setiap $n \geq i$, *i* adalah basis.

Induksi matematika mempunyai efek yang sama dengan domino sebagaimana diilustrasikan pada gambar 2.1. Jika batu ke- *k* pada domino rubuh, maka pasti batu ke- *k*+1 akan ikut rubuh hingga akhirnya semua domino rubuh. Begitu pula dengan induksi matematika, jika P(n) benar, maka P(n+1) juga benar.




Gambar 2.1 : Jika domino ke- *k* rubuh, maka domino ke- *k*+1 dan selanjutnya akan ikut rubuh.

Sebagai contoh, kita akan membuktikan proposisi berikut dengan metode induksi matematika.

*Buktikan bahwa jumlah n bilangan ganjil positif pertama adalah* $n^2$

*Langkah penyelesaian*

1. Basis: Untuk n=1, jumlah satu bilangan ganjil positif pertama adalah $1^2 = 1$. Hal tersebut benar karena jumlah satu bilangan ganjil positf pertama adalah 1.
2. Langkah induksi: Asumsikan pernyataan P(n) benar, yaitu:

   $$1+3+5+...+(2n-1)=n^2$$

   Kita juga harus memperlihatkan bahwa P(n+1) benar, yaitu

   $$1+3+5+...+(2n-1)+(2n+1)=(n+1)^2$$

   Hal tersebut dapat ditunjukan sebagai berikut:

$$\begin{aligned} 1+3+5+...+(2n-1)+(2n+1) &= [1+3+5+...+(2n-1)] \\ &\quad + (2n+1) \\ &= n^2 + 2n + 1 \\ &= (n+1)^2 \end{aligned}$$

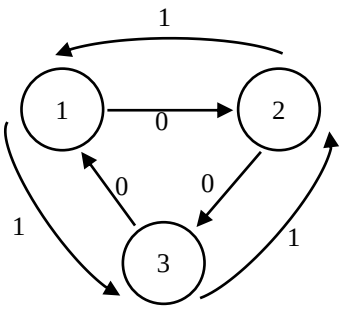
Karena langkah basis dan langkah induksi telah diperlihatkan benar, maka pernyataan tersebut benar, yaitu jumlah n bilangan ganjil positif pertama adalah $n^2$.

## III. MENGUBAH DFA MENJADI EKSPRESI REGULER

DFA dan ekspresi reguler mendeskripsikan bahasa yang sama, yaitu bahasa reguler. Oleh karena itu, kita dapat mengatakan bahwa jika L = L(A) untuk suatu DFA A, maka terdapat ekspresi reguler sehingga L = L(R). Berdasarkan teorema tersebut, kita akan mencoba untuk mengubah DFA menjadi ekspresi reguler.

### *A. K-Paths*

Misalkan A mempunyai state dengan nomor {1,2,…,n} untuk sebuah bilangan bulat positif n. Dengan mengubah nama state dari DFA tersebut, pembuktian dengan induksi akan lebih mudah karena kita dapat menganggap state tersebut sebagai n bilangan bulat positf pertama. Kemudian kita akan membuat sebuah ekspresi reguler yang menggambarkan lintasan yang dapat ditempuh oleh DFA A. Lintasan yang akan kita buat ini dinamakan dengan k-*paths*. K-*paths* merupakan sebuah lintasan yang dapat dibentuk pada graf suatu DFA A yang *intermediate* statenya tidak melebihi *k*.




Gambar 3.1: DFA A.

Untuk memahami lebih dalam tentang k-paths, mari kita lihat gambar 3.1. DFA A terdiri atas 3 state, oleh karena itu, kita dapat membuat *k-paths* dengan k = 0 hingga k = 3.

0-*paths* dari state 2 ke state 3: R **= 0.**
1-*paths* dari state 2 ke state 3: R **= 0+11.**
2-*paths* dari state 2 ke state 3: R **= (10)*+0+1(01)*1**

Kita akan menggunakan *k-paths* untuk mengubah DFA menjadi ekpresi reguler dengan induksi matematika

### *B. Induksi Dengan K-Paths*

Kita akan menggunakan notasi R*ij*$^k$ sebagai nama dari ekspresi reguler yang bahasanya adalah himpunan *string* w, yang pada hal ini w adalah *k-paths* dari state i menuju state j. Berikut adalah definisi induktif untuk membangun ekspresi reguler tersebut:

*Basis : k = 0*, yaitu Rij$^0$. Karena state dimulai dari angka 1, lintasan yang harus diambil tidak boleh memiliki *intermediate* state sama sekali. Lintasan yang memenuhi kondisi seperti di atas adalah:

1. Sisi dari simpul (state) 1 ke simpul j.
2. Lintasan dengan panjang 0 yang hanya terdiri dari simpul *i*.

Oleh karena itu, kita dapat menyimpulkan bahwa pada DFA A dan simbol masukan $a$ terdapat beberapa kemungkinan sebagai berikut:

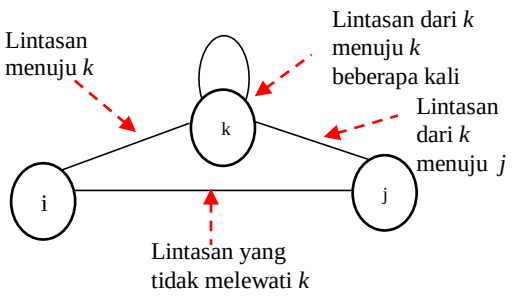
- Jika tidak ada fungsi transisi untuk simbol $a$, maka Rij$^0$ = $\emptyset$.
- Jika ada fungsi transisi untuk simbol $a$, maka Rij$^0$ = $a$.
- Jika ada fungsi transisi untuk simbol $a_1, a_1, \ldots, a_k$ maka Rij$^0$ = $a_1 + a_2 + \cdots + a_k$.

Untuk kasus *i* = *j*, lintasan yang dimungkinkan adalah lintasan dengan panjang 0 dan lintasan yang menuju dirinya sendiri. Lintasan dengan panjang 0 disimbolkan oleh $\epsilon$

*Langkah Induksi*: Asumsikan bahwa ada lintasan *k-paths* dari state *i* ke state *j*. Untuk hal ini, ada dua kasus yang dapat dipertimbangkan:

1. Lintasan tersebut tidak melalui state k. Pada kasus ini, ekspresi reguler yang mendeskripsikan lintasan tersebut adalah $R_{ij}^{k-1}$.
2. Lintasan tersebut melalui state k setidaknya satu kali. Kita dapat membagi lintasan tersebut menjadi beberapa bagian, seperti yang digambarkan pada gambar 3.2.
   - Bagian pertama adalah lintasan dari state *i* yang langsung menuju pada state *k*. Lintasan ini dideskripsikan oleh $R_{ik}^{k-1}$
   - Yang kedua adalah bagian yang menggambarkan lintasan dari state *k*, menuju dirinya sendiri sebanyak nol atau lebih. Lintasan ini dideskripsikan oleh $(R_{kk}^{k-1})^*$
   - Bagian ketiga merupakan lintasan dari state *k* yang menuju state *j*. Lintasan ini dideskripsikan oleh $R_{kj}^{k-1}$

   Ketiga bagian tersebut sudah mencakup semua lintasan yang dimungkinkan dari state *i* menuju state *j*.




Gambar 3.2: Semua kemungkinan lintasan yang dapat ditempuh dari state *i* ke state *j*.

Secara singkat, untuk pergi dari state *i* ke state *j*, kita secara langsung dapat menuju ke state *j* atau kita dapat pergi ke state yang lebih besar dari *j*, misalkan *k*. Pada state *k* kita mungkin saja menuju kembali ke state *k* beberapa kali, lalu setelah itu kita menuju state *j*. Hal tersebut dapat dideskripsikan oleh ekspresi reguler berikut:

$$R_{ij}^{\,k} = R_{ij}^{\,k-1} + R_{ik}^{\,k-1}\left(R_{kk}^{\,k-1}\right)^* R_{kj}^{\,k-1}$$

Sekarang, kita akan coba untuk mengubah DFA yang ada pada gambar 1.1 menjadi ekspresi reguler. DFA ini menerima input yang setidaknya memiliki satu 0. Mengapa? Karena untuk mencapai state akhir, DFA harus menerima masukan simbol 0 pada saat berada di state awal. Tabel di bawah adalah basis untuk membentuk ekspresi reguler.

| | |
|---|---|
| $R_{11}^{(0)}$ | $\epsilon + 1$ |
| $R_{12}^{\,k}$ | 0 |
| $R_{21}^{\,k}$ | Ø |
| $R_{22}^{\,k}$ | $\epsilon + 0 + 1$ |

Tabel 3.1: Langkah basis, yaitu $R_{ij}^{(0)}$

Selanjutnya, kita akan melakukan langkah induksi. Ekspresi yang akan dibagun pada tahap ini adalah

$$R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)}(R_{11}^{(0)})^* R_{1j}^{(0)}$$

| | Substitusi secara langsung | Penyederh-anaan |
|---|---|---|
| $R_{11}^{(1)}$ | $\epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1)$ | 1* |
| $R_{12}^{(1)}$ | $0 + (\epsilon + 1)(\epsilon + 1)^*0$ | 1*0 |
| $R_{21}^{(1)}$ | $\emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1)$ | Ø |
| $R_{22}^{(1)}$ | $(\epsilon + 0 + 1) + \emptyset(\epsilon + 1)0$ | $\epsilon + 0 + 1$ |

Tabel 3.2: Langkah induksi tahap 1, yaitu $R_{ij}^{(1)}$

Pada tahap ini, kita melakukan penyederhanaan terhadap ekspresi reguler. Penyederhanaan dilakukan untuk memudahkan pekerjaan kita, jika tidak, ekspresi yang dihasilkan akan sangat panjang dan sulit untuk dibaca serta merepotkan. Penyederhanaan pada $R_{21}^{(1)}$ dan $R_{22}^{(1)}$bergantung pada aturan operasi ekspresi reguler dengan Ø. Untuk setiap ekspresi reguler R:

1. $\emptyset R = R\emptyset = \emptyset$. Ø merupakan *annihilator* untuk konkatenasi. Hasil konkatenasi dengan Ø selalu menghasilkan dirinya sendiri.
2. $\emptyset + R = R + \emptyset = R$. Ø merupakan identitas untuk union. Hasil union dengan Ø tidak mengubah apapun.

Sekarang, kita akan membangun ekspresi $R_{ij}^{(2)}$, dengan k = 2, didapat

$$R_{ij}^{(2)} = R_{ij}^{(2)} + R_{i2}^{(2)}(R_{22}^{(2)})^* R_{2j}^{(2)}$$

| | Substitusi secara langsung | Penyederh-anaan |
|---|---|---|
| $R_{11}^{(2)}$ | $1^* + 1^*0 + (\epsilon + 0 + 1)^*\emptyset$ | 1* |
| $R_{12}^{(2)}$ | $1^*0 + 1^*0 + (\epsilon + 0 + 1)^*(\epsilon + 0 + 1)$ | $1^*0(0 + 1)^*$ |
| $R_{21}^{(2)}$ | $\emptyset + (\epsilon + 0 + 1)^*(\epsilon + 0 + 1)\emptyset$ | Ø |
| $R_{22}^{(2)}$ | $(\epsilon + 0 + 1) + (\epsilon + 0 + 1)(\epsilon + 0 + 1)^*(\epsilon + 0 + 1)$ | $(0 + 1)^*$ |

Tabel 3.3: Langkah induksi tahap 2, yaitu $R_{ij}^{(2)}$

Pada tahap ini, kita sudah membuat semua ekspresi yang menggambarkan semua lintasan yang mungkin. Ekspresi regular tersebut ekivalen dengan DFA pada gambar 1.1. Pada kasus ini, 1 adalah state awal dan 2 adalah satu satunya state akhir. Oleh karena itu, untuk mendefinisikan bahasa yang sama dengan DFA tersebut, kita hanya memerlukan lintasan dari state awal ke state akhir, yaitu $R_{12}^{(2)}$. Ekspresi tersebut adalah $1^*0(0 + 1)^*$. Secara intuisi, sangat mudah untuk menginterpretasikan ekspresi tersebut. Bahasanya terdiri atas *string* yang dimulai dengan beberapa simbol 1 diikuti dengan simbol 0, kemudian diikuti oleh beberapa gabungan simbol 0 dan 1. Dengan kata lain, bahasa tersebut terdiri atas *string* yang setidaknya mempunyai satu simbol 0.

## IV. KESIMPULAN

Induksi matematika dapat menyelesaikan berbagai macam persoalan matematika. Salah satu penerapannya adalah pada teori automata. Dari pembahasan di atas, dapat disimpulkan bahwa DFA dapat diubah menjadi ekspresi reguler dengan mencari solusi dari

$$R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)}(R_{11}^{(0)})^* R_{1j}^{(0)}$$

## V. UCAPAN TERIMA KASIH

Saya mengucapkan terimakasih kepada Tuhan Yang Maha Esa karena atas rahmat-Nya makalah ini dapat diselesaikan. Tidak lupa saya mengucapkan terimakasih kepada Pak Rinaldi Munir dan Bu Harlili atas ilmu dan bimbingannya dalam kuliah IF2120 Matematika Diskrit. Saya juga ingin mengucapkan terimakasih kepada teman-teman atas dukungan yang diberikan pada penulisan makalah ini.

.

## REFERENSI

[1] Munir, Rinaldi. 2009 "Diktat Kuliah IF 2091 Struktur Diskrit". Bandung:Program Studi Teknik Informatika STEI ITB.

[2] J.E. Hopcroft, R. Motwani, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2001.

[3] http://www.cs.man.ac.uk/~pjj/farrell/comp3.html. Diakses pada 10/12/2014 17.44 WIB.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 10 December 2014

Husni Munaya -13513022