ANALISIS PETA PERKEMBANGAN DAN KERAGAMAN FINITE AUTOMATA

Wiwin Sulistyo[1], Reza Pulungan[2]
[1]Teknik Informatika,FTI, UKSW,
Jl. Diponegoro 52-60, Salatiga 50711
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA, UGM
Sekip Utara BLS 21 Yogyakarta 55281
Email: [1]wiwinsulistyo@staff.uksw.edu, [2]pulungan@ugm.ac.id

Abstract:

The theory of automata is adiscipline that has great impacts on the development of computer science, in particular the theory of computation. At this time, the development of the theory of automata is very rapid. In the 1940s up to the 1950s, a simple computational model of the machine called "finite automata" was widely studied. Finite automata are abstract machines that can recognize, accept and generate a sentence in regular languages. Finite automata as one of the models of automata theory have been used in building the components of computer hardware and software.Furthermore, Noam Chomsky mapped languages into 4 levels (Chomsky hierarchy) and regular languages belong to the simplest level. Because of the importance of automata in the development of computational science, automata researches to date continue to grow. This paper tries to survey and assess research and developments in automata theory, especially in the development of finite automata models as well as some variant forms or other automata.

Key words:*Automata, finite state automata, Hirarki Chomsky, context free grammar*

Abstrak:

Teori otomata adalah disiplin ilmu yang berpengaruh besar pada perkembangan Ilmu Komputer khususnya teori komputasi. Pada tahun 1940an sampai dengan 1950an, model mesin komputasi sederhana yang disebut "finite automata" banyak diteliti. Disebutkan finite automata adalah mesin abstrak yang dapat mengenali , menerima dan membangkitkan sebuah kalimat dalam bahasa yang reguler. Pada saat ini, perkembangan teori otomata sangat pesat. Finite automata adalah salah satu model teori otomata yang dikembangkan untuk membangun komponen komputer baik *hardware* maupun *software*.Selanjutnya, Noam Chomsky memetakan tingkatan bahasa menjadi 4 tingkatan (Hirarki Chomsky) dan bahasa yang regular berada pada tingkat yang paling sederhana. Karena pentingnya otomata dalam perkembangan ilmu komputasi, maka sampai saat ini penelitian automata terus berkembang. Oleh sebab itu, makalah ini mencoba melakukan pengkajian perkembangan penelitian teori otomata khususnya pada perkembangan model finite otomata serta beberapa bentuk-bentuk atau varian otomata lainnya

Kata Kunci:*Automata, finite state automata, Hirarki Chomsky, context free grammar*

PENDAHULUAN

Perkembangan teori komputasi sangat ditentukan oleh perkembangan teori otomata. Teori otomata merupakan disiplin ilmu yang mempelajari hal-hal yang berhubungan dengan perangkat komputer yang bersifat abstrak yang sering disebut dengan mesin ("*machines*") [12]. Pada tahun 1930an, Alan Turing telah mengkaji mesin abstraksi yang telah memiliki kemampuan seperti mesin komputer yang saat ini ada. Selain itu, Alan Turing juga telah mendeskripsikan batasan-batasan yang dapat dilakukan atau yang tidak dapat dilakukan oleh mesin komputasi [12].

Selanjutnya bermunculan bermacam-macam definisi terkait dengan teori otomata, hal ini seiring dengan perkembangan penelitian dari teori otomata tersebut. Secara umum otomata merupakan mesin abstrak yang dapat mengenali (*recognize*), menerima (*accept*) dan membangkitkan (*generate*) sebuah kalimat dalam bahasa tertentu [23]. Pada tahun 1940

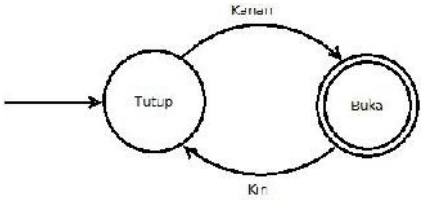
dan 1950, jenis model mesin komputasi sederhana yang disebut dengan "*finite automata*" telah banyak diteliti. Selanjutnya pada akhir tahun 1950, seorang ahli bahasa Noam Chomsky mulai mengkaji ilmu yang berkaitan dengan bahasa yang disebut dengan "*formal grammars*" (tata bahasa formal). Tata bahasa formal ini memiliki hubungan dekat dengan model otomata dan merupakan dasar dari beberapa komponen perangkat lunak yang penting, termasuk pada bagian kompiler [7]. Pada penelitiannya, Noam Chomsky juga mengklasifikasikan tingkatan bahasa menjadi 4 tingkatan yang disebut dengan Hirarki Chomsky. Tingkatan tersebut dibagi dalam beberapa tipe, antara lain tipe 0 yakni *recursively enumerable language* (L_{RE}), tipe 1 yakni *context-sensitive language* (L_{CS}), tipe 2 yakni *Context-Free Language* (L_{CF}) dan tipe 3 yang disebut dengan *Reguler Language* (L_{REG}) [19].

Teori *automata* menjadi bagian yang penting dalam perkembangan disiplin ilmu komputer. *Finite automata* merupakan model yang dikembangkan untuk hal-hal penting yang menyangkut *hardware* dan *software*. Konsep dan model *finite state automata*telah digunakan antara lain pada perangkat lunak untuk perancangan dan pengecekan *digital circuit*, jenis kompiler untuk *lexical analyzer*, perangkat lunak untuk *scanning* bodi teks serta digunakan pada protokol komunikasi untuk sistem keamanan informasi [12]. Oleh sebab itu, hal yang sangat penting untuk mempelajari perkembangan *finite automata* melalui penelitian dan publikasi yang saat ini ada, karena berkaitan erat dengan ilmu komputasi yang memiliki peranan penting dalam perkembangan disiplin ilmu komputer.

FINITE AUTOMATA SECARA UMUM

1. *Finite Automata* Klasik

Pada penggunaannya *Finite State Automata* (FSA) memiliki bagian-bagian dalam modelnya, salah satu contohnya dalam memodelkan kasus kran air seperti pada Gambar 1. Gambar 1 memberikan contoh pemodelan FSA untuk kondisi kran air antara putar "kanan" untuk membuka kran air dan putar "kiri" untuk menutup air. Model FSA terdiri dari beberapa bagian antara lain, lingkaran tunggal menyatakan *state* (kedudukan), lingkaran ganda menyatakan kedudukan akhir (*final state*) dan label pada lingkaran, selanjutnya busur menyatakan transisi (perpindahan kedudukan/*state*), sedangkan label pada busur melambangkan simbol *input*, serta sebuah lingkaran yang didahului dengan busur paling kiri menyatakan *state* awal. Notasi yang penting didalam struktur FSA adalah *Grammars*dan *Regular Expressions* [12]. FSA memiliki jumlah *state* yang terbatas sesuai dengan kondisi yang dibangun, serta memungkinkan adanya perpindahan dari *state* yang satu ke *state* yang lain melalui fungsi transisi (seperti yang terdapat pada Gambar 1). FSA merupakan jenis otomata yang tidakmemiliki tempat penyimpanan, sehingga memiliki kemampuan mengingat yang terbatas [27].

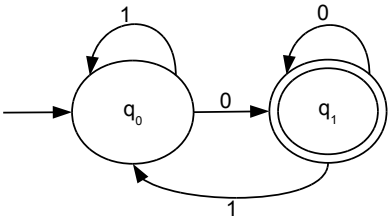


Gambar 1. Pemodelan FSA untuk kran air

Berdasarkan jenisnya, *finite state automata* (fsa) dibagi dalam dua jenis, yakni *Deterministic Finite Automata* (DFA) dan *Non-deterministic Finite Automata* (NFA). *Deterministic Finite Automata* (DFA) memiliki karakteristik dimana untuk setiap *input* yang diterima memiliki tepat satu *state* ke *state* berikutnya. Sehingga secara formal dapat dinyatakan sebagai berikut [12][32]:

Q = Himpunan *state*/kedudukan
Σ = Himpunan input
δ = Fungsi Transisiδ: Q x Σ→ Q
S = *State*/Kedudukan Awal, dimana S∈Q
F = *Final State* (Kedudukan Akhir), dimana F ⊆Q

Sehingga secara formal, ke 5 tupel diatas dapat dituliskan D=(Q, Σ, δ, S, F); sebagai contoh diperlihatkan pada Gambar 2.



Gambar 2. *Deterministic Finite Automata* (DFA)

Berdasarkan notasi diatas, maka secara formal model DFA pada Gambar 2 dapat dinyatakan sebagai berikut:

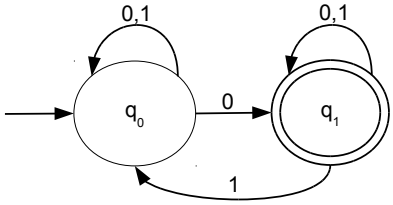
$Q = \{q_0, q_1\}$,$\Sigma = \{0, 1\}$, $A = q_0$, $F = \{q_1\}$.

Selanjutnya, fungsi transisi dapat dinyatakan seperti pada Tabel 1.

Tabel 1. Fungsi transisi *Deterministic Finite Automata* (DFA)

δ	0	1
q_0	q_1	q_0
q_1	q_1	q_0

Sedangkan untuk *Non-deterministik Finite Automata* (*NFA*) secara umum memiliki karakteristik dimana untuk setiap *input* yang diterima dapat memiliki lebih satu *state* ke *state* berikutnya. Sehingga secara lengkap dapat dituliskan sebagai N =(Q, Σ, δ, S, F) [12][32].



Gambar 3. *Non-deterministic Finite Automata* (DFA)

Oleh sebab itu, untuk model NFA seperti pada Gambar 3 dapat didefinisikan sebagai berikut:

$Q = \{q_0, q_1\}$, $\Sigma = \{0, 1\}$, $S = q_0$, $F = \{q_1\}$.

Fungsi transisinya, oleh karena itu, dapat dinyatakan seperti pada Tabel 2.

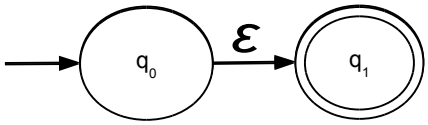
Tabel 2. Fungsi transisi *Non-Deterministic Finite Automata* (NFA)

δ	0	1
q_0	$\{q_0, q_1\}$	q_0
q_1	q_1	$\{q_0, q_1\}$

Sehingga dapat dilihat bahwa antara DFA dan NFA memiliki perbedaan pada fungsi transisinya, dimana DFA untuk setiap *input* yang diterima memiliki tepat satu *state* ke *state* berikutnya, sedangkan NFA memilki karakteristik dimana untuk setiap *input* yang diterima dapat memiliki lebih satu *state* ke *state* berikutnya.

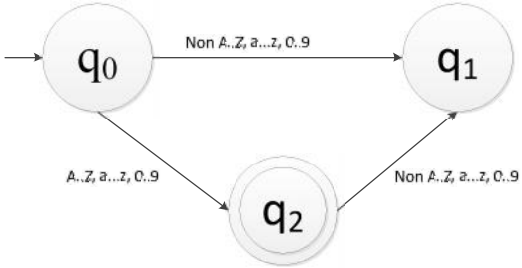
Pada perkembangan berikutnya terdapat ekuivalansi antar DFA, dimana masing-masing otomata dapat menerima bahasa yang sama yakni $L(D_1)$ dan $L(D_2)$, sehingga dengan kondisi bahwa $L(D_1)=L(D_2)$, dapat dikatakan bahwa kedua DFA tersebut ekuivalen. Selain itu, ekuivalensi antara NFA ke DFA, juga dapat terjadi ketika antara NFA dan DFA dapat menerima bahasa yang sama. Hal ini dapat dilakukan dengan memanfaatkan tabel transisi diantara kedua mesin tersebut.

Untuk menyederhanakan model otomata pada DFA dapat dilakukan dengan mereduksi *state* dari suatu *finite state automata* dengan mengurangi jumlah *state*, tapi tanpa mempengaruhi kemampuan awalnya dalam menerima bahasa [27]. Hal tersebut dapat dilakukan dengan konsep *distinguishable* dan *indistinguishable*. Dua buah*state*disebut*distinguishable* jika terdapat $w \in \Sigma^*$ sedemikian hingga, $\delta(q, w) \in F$, sedang $\delta(p, w) \notin F$, Sedangkan dua buah state DFA disebut*indistinguishable* jika $\delta(q, w) \in F$, begitu juga $\delta(p, w) \in F$ [32]. Pada *Non-deterministik Finite Automata* (NFA), terdapat jenis otomata lainnya yang disebut dengan *Non-deterministik Finite Automata* dengan *ε-move*yang memungkinkan terjadinya perpindahan*state* tanpa mengkonsumsi input ("ε") seperti pada gambar 4 [27]. Sehingga secara umum ε-NFA (NFA *ε-move*) sama dengan NFA, kecuali diijinkannya kondisi *ε-move*. Oleh sebab itu, dapat dilakukan ekuivalensi antara ε-NFA ke NFA.



Gambar 4. Non-deterministik Finite Automata (NFA) dengan *ε-move*

Pada perkembangan berikutnya FSA dapat menerima bahasa yang didefinisikan dalam notasi aljabar. Bahasa-bahasa yang dapat diterima oleh automata ini disebut dengan Ekspresi Reguler (ER) atau *Regular Expression* (*RE*). Ekspresi reguler merupakan struktur notasi untuk menggambarkan pola/corak suatu *string* yang sama dari suatu bahasa yang dapat direperesentasikan oleh *finite state automata* [12]. Model bahasa seperti ini biasanya dapat dicontohkan dalam kasus pembatasan penerimaan atau pembacaan *string*, misalkan FSA hanya menerima abjad A...Z, a..z dan angka 0..9, maka model mesin otomatanya dapat dilihat pada Gambar 5.



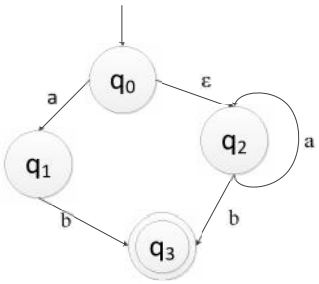
Gambar 5. Model FSA menerima abjad dan angka

Dari Gambar 5 ekspresi regulernya dapat dinyatakan dengan (huruf)* ∪ (angka)*. Dalam hubungannya, antara DFA, NFA dan Ekspresi Reguler dapat dinyatakan seperti yang terdapat pada Gambar 9.

Ekspresi reguler memiliki perkembangan kearah tata bahasa reguler dan tata bahasa bebas konteks (*context free grammar*). Tata bahasa reguler dapat dibangun atau dikontruksikan untuk aturan-aturan produksi dari suatu tata bahasa reguler. Dimana batasan-batasan aturan produksi pada bahasa reguler dapat dirumuskan $\alpha \rightarrow \beta$, dimana α melambangkan sebuah simbol variabel dan β melambangkan sebuah simbol variabel yang terletak di paling kanan [27]. Dengan demikian terdapat batasan aturan bahasa reguler, bahwa maksimal jumlah simbol variabel yang terletak di posisi paling kanan adalah satu, misal didefinisikan dalam notasi G=(V, T, P, S) [32], dimana:

V:Himpunan simbol variabel/non terminal
T:Himpunan simbol terminal
P:Kumpulan aturan produksi
S: Simbol Awal

Sebagai contoh, diketahui mesin otomata untuk menggambarkan aturan produksi pada bahasa reguler yang ada pada FSA seperti pada Gambar 6.



Gambar 6. Aturan produksi mesin FSA pada bahasa reguler

Berdasarkan Gambar 6, secara formal dapat didefiniskan variabel-varaiabel sebagai berikut:

V={S, A, B, C}, T={a, b}, P={S→aA | B, A→b, B→aB| b}, S= S.

2. Perkembangan *FINITE AUTOMATA*

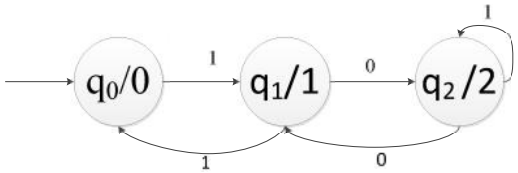
Mencermati kelemahan-kelemahan yang terdapat pada *finite state automata* yang hanya bersifat sebagai "*accepter*", dimana FSA tersebut memiliki keterbatasan pada keputusan yang dihasilkan yakni antara diterima atau ditolak saja. Pada perkembangan berikutnya, terdapat *finite state automata* yang memiliki kemampuan untuk menghasilkan keputusan yang berupa beberapa keluaran. FSA yang memiliki kemampuan seperti ini disebut dengan *finite state automata* "*tranducer*". Terdapat beberapa contoh mesin abstrak yang memiliki kemampuann *tranducer*, diantaranya adalah Mesin Moore dan Mesin Mealy. Pada Mesin Moore, keluaran/*output*akan berasosiasi dengan *state*, sehingga dapat didefinisikan [27]:

$M = (Q, \Sigma, \delta, S, \Delta, \lambda)$ ……(1)

Dimana:

Q = Himpunan State
Σ = himpunan simbol input
δ = fungsi transisi
S = state awal, $S \in Q$
Δ = himpunan output/keluaran
λ = fungsi output untuk setiap "***state"***

Sebagai contoh adalah model mesin Moore untuk penghitungan Modulus (sisa pembagian) suatu bilangan input (dalam biner) dengan bilangan 3, maka dapat digambarkan seperti pada Gambar 7.



Gambar 7. Mesin Moore untuk penghitunganmodulus 3 [33]

Berdasarkan Gambar 7 maka dapat didefinisikan untuk masing-masing variabelnya adalah sebagai berikut:

Q: (q_0,q_1,q_2)
Σ : [0,1]
Δ : [0,1,2]
S : q_0, $\lambda(q_0, 1) = 1$, $\lambda(q_1,0) = 2$, $\lambda(q_1, 1) = 0$, $\lambda(q_2, 0) = 1$, $\lambda(q_2, 1) = 2$

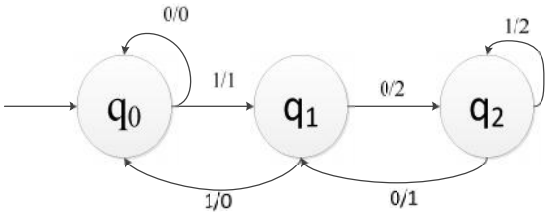
Pada Mesin Mealy, keluaran/*output* berasosiasi dengan transisi. Sehingga pada Mesin Mealy dapat didefiniskan sebagai berikut [27]:

$M = (Q, \Sigma, \delta, S, \Delta, \lambda)$ ……(2)

Dimana:

Q = Himpunan State
Σ = himpunan simbol input
δ = fungsi transisi
S = state awal, $S \in Q$
Δ = himpunan output/keluaran
λ = fungsi output untuk setiap "**transisi"**

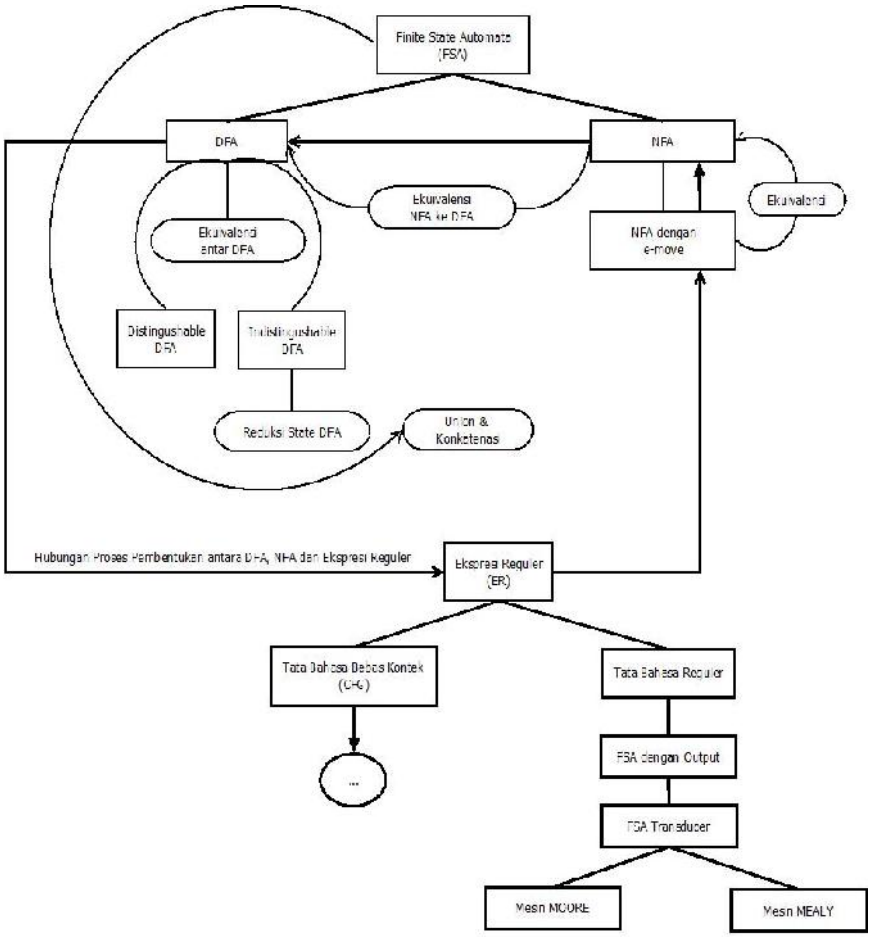
Sebagai contoh, maka pemodelan dari Gambar 7 dapat juga dimodelkan dengan Mesin Mealy menjadi Gambar 8 [27]. Selanjutnya, antara Mesin Moore dan Mesin Mealy dapat ditentukan equvalensi keduanya.



Gambar 8. Mesin Mealy untuk penghitungan modulus 3 [27]

Secara umum, perkembangan *finite state automata* (FSA)telah mengalami proses yang sangat panjang. Secara garis besar perkembangan *finite state automata* (FSA) dapat digambarkan seperti pada Gambar 9.

Pada Gambar 9 terdapat cabang yakni tata bahasa bebas kontek atau sering disebut dengan *Context Free Grammar* (CFG) yang secara garis besar dapat dilihat pada Gambar 13. Tata bahasa reguler memiliki aturan-aturan produksi yang dapat dirumuskan dengan$\alpha \rightarrow \beta$, dimana terdapat batasan pada tata bahasa reguler, bahwa maksimal jumlah simbol variabel yang terletak di posisi paling kanan adalah satu. Sedangkan, tata bahasa reguler *Context-Free Language* (Bahasa Bebas Kontek) merupakan perluasan dari kelas tata bahasa regular, dimana *context-free grammar* (CFG) merupakan bahasa yang bersifat natural/baku dan bersifat rekursif [12]. Oleh sebab itu, bahasa bebas kontek memiliki batasan bahwa pada ruas kiri haruslah memiliki simbol-simbol variabel tepat satu [19] [27], contohnya K→e, H→XeZ, T→Xyz.



Gambar 9. Peta perkembangan FSA

Grammar merupakan notasi yang digunakan untuk mendefinisikan bahasa (baik yangnatural maupun pemrograman) secara rekursif, sebagai contoh*Palindromes*. *Palindrome* adalah string-string yang dibaca sama daridepan dan dari belakang, sehingga dapat dituliskan [12]:

$L_{PAL} = \{w \quad \{0,1\}^* \mid w^R = w\}$,

Grammar:

$P \rightarrow \varepsilon$

$P \rightarrow 0$
$P \rightarrow 1$
BASIS: ε, 0, 1 adalah *palindrome*,
$P \rightarrow 0P0$
$P \rightarrow 1P1$
INDUKSI: *P palindrome* $P0,1P1$ saja.

Secara implisit dapat dikatakan bahwa semua *palindrome* dapat diperoleh denganmengaplikasikan aturan-aturan di atas secara rekursif, tetapihanya *palindrome* yang dapat diperoleh denganaturan-aturan tersebut. Terdapat empat komponen didalam CFG yang dapat dinyatakan dengan G=(V,T,P,S), dimana[12]:

- *T*: adalah himpunan simbol-simbol terminal ($T = \Sigma$),
- *V*: adalah himpunan simbol-simbol non-terminal (variabel)yang merepresentasikan himpunan-himpunan string yangdidefinisikan secara rekursif,
- *S*: simbol *start* ($S \quad V$). *S* adalah variabel khusus yangmembangkitkan keseluruhan bahasa yang diinginkan,
- *P*: himpunan aturan-aturan produksi yang berisidefinisi-definisi pembangkitan yang rekursif.

V, T dan P adalah himpunan yang terhingga, sebagai contoh:

Grammar *Geq* = (V , T ,P,S) , di mana:
$T = \{0, 1\}$,
$V = \{S,A,B\}$,
$P = \{S \rightarrow \varepsilon| 0A \mid 1B, A \rightarrow 1S \mid 0AA, B \rightarrow 0S \mid 1BB\}$

Context-Free Grammar hanya diperbolehkan memilikiaturan-aturan produksi untuk substitusi berbentuk[12]:

$A \rightarrow \alpha_1\alpha_2 \cdots \alpha k$, di mana,
LHS: $A \quad V$, dan
RHS: $\alpha_i \quad V \quad T$ untuk semua i .

Didalam CFG terdapat derivasi (penurunan) yang digunakan untuk menggambarkan perolehan suatu string dengan cara menurunkan simbol-simbol variabel menjadi simbol-simbol terminal, misal untuk CFG G, $\alpha, \beta, \gamma \quad (V \quad T)^*$, $A \quad V$, dapat tuliskan menjadi $\alpha A\beta \quad \alpha\gamma\beta$ (derivasi langsung), jika aturan produksi $A \rightarrow \gamma$ adalah di G. Derivasi tersebutberarti $\alpha\gamma\beta$ dapat diturunkan dari $\alpha A\beta$. Sehingga, andaikan barisan derivasi: $\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha r$, dapat dituliskan menjadi $\alpha_1 \overset{*}{\Rightarrow} \alpha_r$,yang berarti α_r dapat diturunkan dariα_1 dalam 0 atau lebih langkah derivasi. Selanjutnya berdasarkan arahnya, derivasi memiliki dua jenis yaitu derivasi*leftmost* dan derivasi *rightmost*[12].Derivasi *leftmost* dapat dicontohkan sebagai berikut:

$$E \underset{lm}{\Rightarrow} E + E \underset{lm}{\Rightarrow} I + E \underset{lm}{\Rightarrow} x + E \underset{lm}{\Rightarrow} x + E * E \underset{lm}{\Rightarrow} x + I * E \underset{lm}{\Rightarrow} \ldots$$

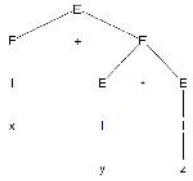
Sedangkan Derivasi *Rightmost* dapat dicontohkan sebagai berikut:

$$E \underset{rm}{\Rightarrow} E + E \underset{rm}{\Rightarrow} E + E * E \underset{rm}{\Rightarrow} E + E * I \underset{rm}{\Rightarrow} \ldots$$

Selanjutnya didalam CFG juga terdapat Pohon Derivasi (*parse tree*) yang digunakan untuk menggambarkan perolehan suatu string dengan cara menurunkan simbol-simbol variabel menjadi simbol-simbol terminal. Secara umum untuk merepresentasikan [12]:

- $A \overset{*}{\Rightarrow} \alpha$:Akar dilabeli A,
- Daun dengan urutan dari kiri ke kanan menghasilkan α,

Nodeinternal dilabeli dengan variabel dan anak-anak darivariabel ini membentuk aturan produksi yang digunakan, sebagai contoh sebuah pohonderivasi dari $E \overset{*}{\Rightarrow} x + y * z$,seperti pada gambar 10 berikut ini:



Gambar 10. Contoh Pohon derivasi

Teorema: Untuk suatu CFG $G = (V , T ,P,S)$ dan string$w \quad T^*$,maka pernyataan-pernyataan berikut adalah *equivalent*, dimana [12]*:*

(a) $w \quad L(G)$ *(atau* $S \overset{*}{\Rightarrow} w$*),*
(b) $S \overset{*}{\Rightarrow} lm\ w$,
(c) $S \overset{*}{\Rightarrow} rm\ w$,
(d) *Terdapat sebuah pohon-S dengan yield w.*

Selain itu, juga terdapat kondisi ambiguitas pada pohon penurunan, yang disebabkan karena adanya lebih dari satu pohon penurunan yang berbeda untuk mendapatkan suatu untai yang sama [27]. Ini bukanlah hanya masalah sintaksis, karena memang diperoleh dua interpretasi semantik yang berbeda, misalkan:

Pohon 1: $x + (y * z)$ dan Pohon 2: $(x + y) * z$

Sehingga, sebuah CFG G dikatakan *ambiguous*, jika untuk suatu w L(G) terdapatdua *parse tree* yang berbeda [12]. Kondisi*compilerparse tree* menentukan interpretasi yang dilakukan, oleh karena itu ambiguitas tidak boleh ada. Sebuah CFL disebut *inherently ambiguous* jika semua*grammarnyaambiguous*[12].

Selanjutnya didalam tata bahasa bebas kontek terdapat penyederhanaan yang bertujuan untuk melakukan pembatasan sehingga tidak

menghasilkan pohon penurunan yang memiliki kerumitan yang tidak perlu atau menghilangkan *contruct* yang dapat melambatkan proses *parsing grammar* dalam menemukan suatu bentuk normalnya. Hal ini bisa dilakukan dengan cara (gambar 13) sebagai berikut[12]:

1. Menghilangkan produksi ε, dalam bentuk α→ε, *X* *V adalah nullable jika* $X \overset{*}{\Rightarrow} \varepsilon$,
2. Menghilangkan produksi unit, unit produksi bentuk *A → B* melambatkan *parser*
3. Menghilangkan produksi *useless*,
   Contoh [23] : S→Aa|B, A→ab|D, B→b|E, C→bb, E→aEa. Produksi yang *useless*: A→D, C→bb, E→aEa, B→E. Penyederhanaannya menjadi: S→Aa|B, A→ab, B→b.

Selanjutnya, terdapat bentuk normal chomsky (*chomsky Normal Form (CNF)*), dimana ruas kanannya memiliki tepat sebuah terminal atau dua variabel. Sehingga, CFG G adalah dalam CNF jika semua aturan produksinyaberbentuk [12]:

(a) *A → a,*
(b) *A → XY ,di mana A,X,Y V dan a T,*
Contoh: S→AB, A→b, B→AC|d

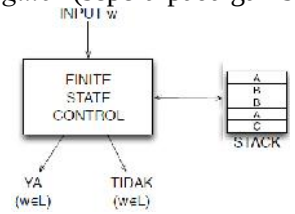
Didalam tata bahasa bebas kontek (CFG) selain CNF juga terdapat Bentuk Normal Greibach (*Greibach Normal Form* (GNF)) yang memiliki aturan produksi: A→aα, dimana a: terminal (a∈T), α: rangkaian simbol-simbol variabel (V^*) [27][32]. Suatu CFG dapat dikatakan dalam bentuk GNF jika ruas kanannya diawali dengan satu simbol terminal, sebagai contoh:

S→b|bBC
B→bC
C→cS

Bentuk Normal Greibach dapat terbentuk bila sudah dalam bentuk CNF, tidak bersifat rekursif kiri serta tidak menghasilkan ε [32]. Terdapat dua cara pembentukan GNF, yakni dengan teknik subtitusi dan perkalian matriks [27]. Contoh pembentukan GNF dengan teknik subtitusi dari bentuk CNF sebagai berikut [23]:

S→CA, A→a|d, B→b, C→DD, D→AB, ditentukan urutan simbol variabel adalah S<A<B<C<D.
Dari kondisi diatas terdapat yang tidak memenuhi urutan yaitu D→AB (dimana D>A tidak sesuai dengn urutan yang telah ditentukan).
Selanjutnya dilakukan subtitusi menjadi D→aB|dB. Setelah itu dilakukan subtitusi mundur, sehingga menjadi:
C→DD⇒C→aBD | DBD
S→CA⇒S→aBDA | dBDA
Sehingga hasil akhir dalam bentuk GNF:
S→aBDA | dBDA
A→a | d
B→b
C→aBD | dBD
D→aB | dB

Selanjutnya, sejak Tahun 1960 CFG memiliki peranan utama dalam perkembangan teknologi kompilasi. Sehingga, untuk semua *Context-Free Grammer*(tipe 2) adalah mungkin untuk membangun suatu *Push Down Automata* (PDA) yang akan mengenalinya. PDA merupakan *finite state automata* yang memiliki memori (berupa *stack*) yang tak terbatas agar bisa menerima bahasa *non-reguler* (seperti pada gambar 11).



Gambar 11. Struktur Model *Push Down Automata* (PDA) [12]

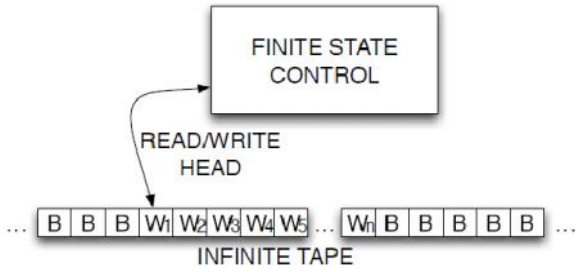
Sehingga, misal terdapat PDA, *M*= (*Q*,Σ,Γ,δ,q_0,Z_0,*F*), dimana:
Q: himpunan *state—p, q, r , s,…,*
Σ: alfabet *input—a, b, c,…,*
Γ: alfabet *stack—A,B,C,…,*
q_0: *Startstate* (*q*0 *Q*),
Z_0: simbol *stack* awal (Z_0 Γ),
F : himpunan *state* Final (*F* *Q*)

Dalam proses transisi, pertama-tama PDA pop top dari *stack* untukmenentukan *X*, membaca *input* untuk menentukan *a* (kecualijika ini adalah transisi ε), kemudian dengan mengetahui *q*, *a*,*X*,PDA memilih secara non-deterministik salah satukemungkingan (p_i, α_i), dari proses tersebut dapat dituliskan sebagai berikut[12]:

δ(*q*,*a*,*X*) = {(p_1,α_1), (p_2,α_2),…}, di mana:
q, *pi* *Q*,*a* Σ atau *a* = ε,*X* Γ,α_i Γ^*

Selanjutnya, pada PDA dibagi menjadi dua jenis yaitu *Deterministik Push Down Automata* (DPDA) dan *Non-deterministik Push Down Automata* (NPDA). Dimana masing-masing terbagi dalam 2 jenis yakni *Null Stack* dan *Final Stack* (gambar 14). Dan diantara DPDA dan NPDA termasuk jenis didalamnya yakni *Null Stack* dan *Final Stack* dapat saling berekuivalensi. Pada perkembangan berikutnya ternyata masih didapati kelemahan yang terdapat *push down automata* (PDA) yakni keterbatasan pola akses

yang terdapat pada *stack* yang digunakan oleh PDA tersebut. Dimana pengambilan data pada tumpukan dibawah akan menyebabkan hilangnya data yang berada diatasnya, yaitu pada proses "pop". Oleh sebab itu, munculah mesin turing yang memiliki kemampuan menyimpan informasi yang tidak terbatas pada suatu pita, dimana pada pita tersebut secara terdapat sel-sel yang digunakan untuk menyimpan informasi yang dilakukan secara *array*, sehingga terdapat *head* yang menandakan posisi pengaksesan pada pita tersebut. Dengan proses transisi tersebut, maka membuat mesin turing lebih fleksibel dalam pola aksesnya. Oleh karena itu mesin Turing dapat memodelkan segalasesuatu yang dapat dikomputasikan. Bentuk umum sebuah mesin turing seperti pada gambar 12.



Gambar 12. Model Mesin Turing [12]

Secara formal sebuah Mesin Turing dapat didefinisikan sebagai berikut[12]:

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$……(3)

di mana,

Q: himpunan terhingga state-state,
Σ: himpunan terhingga alphabet input,
Γ: himpunan terhingga alphabet tape,
δ: fungsi transisi,
q_0 Q: state initial (awal),
B Γ: simbol kosong (blank) di tape,
F Q: himpunan state-state Final.

Dimana,

- $\Sigma \subseteq \Gamma$: karena input string w disediakan di tape,
- $B \in (\Gamma - \Sigma)$: sisa *tape* pada awalnya kosong,
- Di awal berada di state $q_0$tape berisi w dikelilingi olehsimbol-simbol kosong

Secara umum fungsi transisinya dapat dinyatakan, $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$, dimana $\delta(q,X) = (p,Y,L)$ berarti jika sedang di state q danhead sedang membaca simbol X, maka pindah ke state p,ganti X dengan Y di *head tape*, dan pindahkan *head tape* 1 *cell*ke kiri[12].

Sehingga secara umum, dengan memperhatikan perkembangan *finite state automata* (FSA) pada cabang bahasa bebas kontek seperti yang telah dipaparkan sebelumnya, maka secara garis besar dapat digambarkan seperti pada gambar 13.

PERKEMBANGAN DAN BENTUK-BENTUK LAIN AUTOMATA

Pada *Evolutionary Automata* (EA) (Mark Burgin dan Eugene Eberbach) menunjukkan evolusi automata menjadi beberapa kelas yang berbeda, antara lain: *Evolutionary Finite Automata* (EFA), *Evolutionary Pushdown Automata* (EPDA), *Evolutionary linear Bounded Automata* (ELBA), *Evolutionary Turing Macines* (ETM) serta *Evolutionary Inductive Turing Macines* (EITM).

Pada tahun 1981 Anne Condon beserta timnya melakukan penelitian yang membahas terkait bahasa yang diterima oleh *non-deterministic log n -tapeautomata* yang memindai masukan mereka hanya sekali dan berhubungan dengan daya komputasi untuk *log* dua arah *n -tape automata* [31]. Selanjutnya pada tahun yang sama, diperkenalkan juga *Finite State Program Linear* (FSPL) untuk memodelkan aplikasi pengolahan data sederhana. Dimana metode aljabar linear digunakan untuk mambangun prosedur yang dapat meminimalkan *register* [31].

Pada perkembangan berikutnya sekitar tahun 1994, dilakukan penelitian terkait masalah dalam pengendalian *automata* pada *string* tak terbatas telah didefinisikan dan dianilisis. Fokus penelitiannya adalah melakukan pengembangan karakterisasi *fixpoint* dari "*controllability subset*" dan *deterministic Rabin outomaton*. Paper ini menyajikan solusi langsung dan efisien untuk masalah dasar dalam teori sistem kejadian diskrit yang memiliki aplikasi untuk mengontrol sintesis program dan memutuskannya secara logis [26]. Pada tahun 1995, Ming Li dan kawan-kawan menyajikanpendekatan baru untuk teori bahasa formal dengan menggunakan kompleksitas Kolmogorov. Hasil utama yang disajikan di sini adalah alternative untuk pumping lemma(s), karakterisasi baru untuk bahasa regular, dan metode baru untuk memisahkan deterministik bahasa bebas konteksd an *non-deterministic* bahasa bebas konteks. Pendekatan ini juga sukses di*high end* dari hirarkiChomsky karena orang dapat mengukur *nonrecursiveness* dalam hal kompleksitas Kolmogorov [18]. Selanjutnya, disusul pada tahun 1997 juga terdapat publikasi hasil penelitian yang melakukan Generalisasi dari konsep "*deterministic*" kepada "l-r-

deterministic" pada *descending tree automata* (disebut *root-to-frontier*) [22]. Kembali pada tahun 1998, Anne Condon dan kawan-kawan mengenalkan ukuran baru untuk mengukur kompleksitas bahasa yang disebut dengan *tiling complexity* dan telah diterapkan pada *finite state aotomata* baik *probabilistic* dan *non-deterministic state* (npfa's) [31].

Untuk memperluas keterbatasan penerimaan yang terdapat *pada finite state automata*, maka pada tahun 2000 Jurgen Dassow beserta rekannya memperkenalkan metode untuk mengatasi keterbatasan tersebut dengan menambahakan elemen yang berasal dari proses karakterisasi baru bahasa bebas konteks [6]. Selanjutnya pada tahun yang sama, Frederique Bassin dalam publikasinya yang berjudul "*Finite State Version Of The Kraft–Mcmillan Theorem**" mampu menghasilkan *finite state* berdasarkan teorema Kraft-Mcmillan (*Finite State* versi Kraft-Mcmillan). Dibuktikan dengan menggunakan konstruksi baru yang disebut konstruksi multiset, yang merupakan versi dengan penggandaan konstruksi pada bagian yang sudah dikenal ("*well-known" subset*) dariteori*automata* [1]. Begitu juga pada tahun 2000, Andreas Birkendorf mampu menghasilkan Efisensi *Learning Deterministic Finite Automata*(DFA) dengan memanfaatkan *Smallest Counterexamples** [2].

Selanjutnya pada tahun 2002, Manuel Delgado melakukan penelitian untuk mencari variasi keterhubungan antara Teori group kombinatorial, teorisemigroup, dan teoribahasa formal [8]. Tidak ketinggalan, pada tahun 2003 Abdelaziz Fellah dan Carma Harding, melakukan penelitian dan menghasilkan *System Timed Alternating Finite Automata* (TAFA) dengan menkombinasikan antara *alternating finite automata* (AFA) dengan *Timed finite automata* (TFA) [11]. Kemudian pada tahun yang sama, munculah desain FSM-Hume dalam kerangka *Finite State* [21]. Selain itu, pada tahun 2004 juga dilakukan penlitian untuk membandingkan secara detil, karakteristik antara *Finite Automata* dengan *Turing Machine* [25]. Sehingga secara lebih detil dapat diketahui kelemahan dan kelebihan masing-masing mesin. Selanjutnya pada tahun 2005, seorang peneliti otomata yaitu Jon M. Corson, mencoba berkreasi dalam mengatasi keterbatasan dalam *finitite automata* dengan menghasilkan *Extended Finite automata* untuk menyelesaian keterbatasan bahasa yang diterima pada Monoid K [5]. Vincent D. Blondel pada tahun yang sama, menghasilkan penelitian yang berkaitan dengan bahasa yang diterima oleh *Quantum Finite Automata* "*empty*" atau "*nonempty*" [3].

Pada tahun 2006 diperkenalkan konsep ekspresi reguler untuk *alphabets* tanpa batas (*Infinite Alphabets*) dan menunjukkan bahwa bahasa dapat ditentukan oleh ekspresi reguler alfabet tak terbatas jika dan hanya jika diterima oleh *unifikasi finite-state* berbasis otomaton [16]. Perkembangan berikutnya terkait pengembangkan teori Kleene–Schutzenberger yang digunakan untuk pembobotan pada Buchi Automata untuk bahasa yang sifatnya "*infinite word*" (kata-kata tanpa batas) [9]. Selanjutnya pada tahun 2007, Tomasz Jurdzinski dalam penelitiannya mencoba untuk menghubungkan *Shrinking Restarting Otomata kearah Finite Change Automata* dan membuahkan hasil [15]. Tidak berhenti disitu, Rajni Jindal dan Shraddha Singhai (2009) membangun arsitektur modular untuk evolusi *Finite State Automata* sehingga menghasilkan *Modular Finite State Automata* [14].

Abuzer Yakaryılmaz dan A.C. Cem Say mencoba mengupas kelebihan model baru *two-way finite automaton* yaitu *probabilistic* dan*quantum finite automata* (2PFAs and 2QFAs) melalui penelitiannya yang berjudul "*Succinctness Of Two-Way Probabilistic And Quantum Finite Automata"* [28]. Selanjutnya, pada tahun 2011 terdapat penelitian yang membahas terkait ambiguitas NFA dalam menerima input dalam proses komunikasi dengan memanfaatkan klasifikasi yang ada pada NFAyaitu UNA *(unambiguous nondeterministic automata*), FNA (*finitely ambiguous NFA*), PNA (*polynomially ambiguous NFA*) and ENA (*exponentially ambiguous NFA*) [13]. Pada tahun yang sama, Gerry Eisman dan Bala Ravikumar juga memperkenalkan beberapa model *Finite Automata* dalam kemampuannya untuk dapat mengenali *non-regular language* [10].

Di tahun 2012, Mark Burgindan Eugene Eberbach, membahasa evolusi atau perkembangan konsep automata dalam kerangka *Evolutionary Computation* (EA) dalam suatu perkembangan komputasi [4]. Selanjutnya terdapat penelitian yang memilikifokus padageneralisasi*Quantum Finite Automata* (QFAs) dimana input beroperasi dalam satu arah atau modus *realtime* dan menyajikan beberapa hasil baru mengenai keunggulannya

dibandingkan finite automata klasik [29]. Pada penerapannya konsep *Finite State Automata* (FSA) juga dapat digunakan untuk membangun simulasi pada mesin pembuat minuman kopi otomatis [20].

Berkaitan dengan ekspresi reguler, terdapat penelitian yang menyelidik ipengacakan ekspresi regular dan konversinya menjadi *non-deterministic finite automata*. Hasil dari penelitian ini adalah untuk membangun sebuah algoritma baru untuk membangun*ε-free non-deterministic finite automata* dari pengacakan ekspresi reguler [17]. Selanjutnya, terdapat penelitian terapan otomata yang menyelidiki struktur komputasi yang terjadi pada interaksi sosial yang terdistribusi, contohnya komunita s*opensource* Wikipedia. Secara statistik dicari pola perilaku kooperatif yang terjadi dan melakukan pemilihan model untuk menentukan apakah aspek sistem tersebut dapat dijelaskan melalui *finite-state model* dan *collective state model*, ternyata *collective statemodel* lebih sederhana dalam prosesnya [7].

KESIMPULAN

Dengan memperhatikan perkembangan teori otomata berdasarkan penelitian-penelitian yang telah dipaparkan sebelumnya, maka pada prinsipnya teori otomata tidak pernah berhenti berevolusi. Sehingga dapat dikatakan bahwa teori automata menjadi bagian yang penting dalam perkembangan disiplin ilmu komputer. Perkembangan *finite automata* terlihat bahwa sebagian besar difokuskan pada peningkatan kemampuan finite automata dalam menerima *input* serta pengembangan batasan aturan produksinya, sehingga mampu mengahasilkan keluaran yang lebih dinamis. Selain itu, terdapat penelitian untuk menghasilkan efisiensi komputasi dari suatu proses finite automata.

REFERENSI

[1] Bassino F., Beal M.P., And Perrin D., 2000, A Finite State Version Of The Kraft–Mcmillan Theorem∗, Siam J. Comput., Vol. 30, No. 4, Pp. 1211–1230, 2000.

[2] Birkendorf A., Boker A, And Simon H.U., 2000, Learning Deterministic Finite Automata From Smallest Counterexamples ∗, Siam J. Discrete Math., Vol. 13, No. 4, Pp. 465–491, 2000.

[3] Blondel V. D., Emmanuel Jeandel, Pascal Koiran And Natacha Portier, 2005, Decidable And Undecidable Problems About Quantum Automata∗, Siam J. Comput., Vol. 34, No. 6, Pp. 1464–1473, 2005).

[4] Burgin M.,Eberbach E., 2012,*Evolutionary Automata: Expressiveness And Convergence Of Evolutionary Computation*,The Computer Journal, Vol. 55 No. 9, 2012.

[5] Corson J.M., 2005, Extended Finite Automata And Word Problems, International Journal Of Algebra And Computation, Vol. 15, No. 3 (2005) 455–466, World Scientific Publishing Company).

[6] Dassow J., Mitrana V., 2000, Finite Automata Over Free Groups, International Journal Of Algebra And Computation Vol. 10, No. 6 (2000) 725-737).

[7] Dedeo S., 2013, Collective Phenomena And Non-Finite State Computation In A Human Social System, Plos One 8(10): E75818. Doi:10.1371/Journal.Pone.0075818, 2013. 12

[8] Delgado M., Margolis S., Steinberg B., 2002, Combinatorial Group Theory, Inverse Monoids, Automata, And Global Semigroup Theory, International Journal Of Algebra And Computation, Vol. 12, Nos. 1 & 2 (2002) 179-211.

[9] Droste M., Uschmann U.P., 2007, On Weighted Buchi Automata With Order-Complete Weights, International Journal Of Algebra And Computation, Vol. 17, No. 2 (2007) 235–260, World Scientific Publishing Company.

[10] Eisman G., Ravikumar B., 2011, On Approximating Non-Regular Languages By Regular Languages, Fundamenta Informaticae 110 (2011) 125–142, Doi 10.3233/Fi-2011-532, Ios Press.

[11] Fellah A. And Harding C., 2003, Language Equations For Timed Alternating Finite Automata, International Journal Of Computer Mathematics, Vol. 80, No. 9, September 2003, Pp. 1075–1091.

[12] Hopcroft J.E., Motwani R., Ullman J.D., 2001, *Introduction to Automata Theory, Language, and Computation*, copyright Addison-Wesley 2001, 2nd Edition , ISBN 0-201-44124-1, 2001.

[13] Hromkovic J. And Schnitger G., 2011, Ambiguity And Communication, Theory Comput Syst (2011) 48: 517–534, Doi 10.1007/S00224-010-9277-4.

[14] Jindal R. And Singhai S., 2009, Finite State Automata Evolution Using Modular Architecture, Journal Of Algorithms & Computational Technology Vol. 4 No. 4, 2009.

[15] Jurdzinski J., Otto F., 2007, Shringking Restarting Otomata*, International Journal Of Foundations Of Computer Science Vo.18, No.2 (2007) 361-385, World Scientific Publishing Company.

[16] Kaminski M., Tan T., 2006, Regular Expressions For Languages Over Infinite Alphabets, Fundamenta Informaticae 69 (2006) 301–318, Ios Press.

[17] Kumar A. And Verma A.K., 2013, A Novel Algorithm For The Conversion Of Shuffle Regular Expressions Into Non-Deterministic Finite Automata, Maejo Inernatioal Journal Of Science And Technology 2013, 7(03), 396-407, Issn 1905-7873, 2013.

[18] Li M. And Vitanyi P., 1995, A New Approach To Formal Language Theory By Kolmogorov Complexity*,Siam J. Comput., Vol. 24, No. 2, Pp. 398-410, April 1995.

[19] Linz P., 2001, *An Introduction to Formal Languages and Automata*, 3th edition, Copyright 2001 by Jones and Bartlett Publishers, Inc, ISBN 0-7637-142, 2001.

[20] Melly E.P, Wamiliana, Kurniawan D., 2012, Penerapan Konsep Finite State Automata (Fsa)Pada Mesin Pembuat Minuman Kopi Otomatis, Jurnal Komputasi, Desember 2012, Vol 1, No. 1.

[21] Michaelson G., HammondK, Serot J., 2003, Fsm-Hume Is Finite State, In proceeding of: Revised Selected Papers from the Fourth Symposium on Trends in Functional Programming, TFP 2003, Edinburgh, United Kingdom, 11-12 September 2003.

[22] Nivat M. And PodelskiA., 1997, Minimal Ascending And Descending Tree Automata, Siam J. Comput, Vol. 26, No. 1, Pp. 39-58, February 1997.

[23] Nugroho A.S., 2013, Teori Bahasa & Otomata, Cetakan Pertama, Graha Ilmu, 2013, ISBN 978-602-262011-2, 2013.

[24] Pnueli A. And SlutzkiG., 1981, Automatic Programming Of Finite State Linear Programs*, Siam J. Comput., Vol. 10, No. 3, August 1981.

[25] Tantau T., 2004, Comparing Verboseness For Finite Automata And Turing Machines, Doi: 10.1007/S00224-003-1108-4, Theory Comput. Systems37, 95–109 (2004)).

[26] Thistle J. G. And WonhamW. M., 1994, Control Of Infinite Behavior Of Finite Automata*, Siam J. Control And Optimization, Vol. 32, No. 4, Pp. 1075-1097, July 1994.

[27] Utdirartatmo F., 2005, Teori Bahasa dan Otomata, Edisi Kedua 2005, Graha Ilmu, ISBN 979-3289-67-8, 2005.

[28] Yakaryılmaz A. And Say A.C.C, 2010, Succinctness Of Two-Way Probabilistic And Quantum Finite Automata, (Discrete Mathematics And Theoretical Computer Science Dmtcs Vol. 12:4, 2010, 19–40.

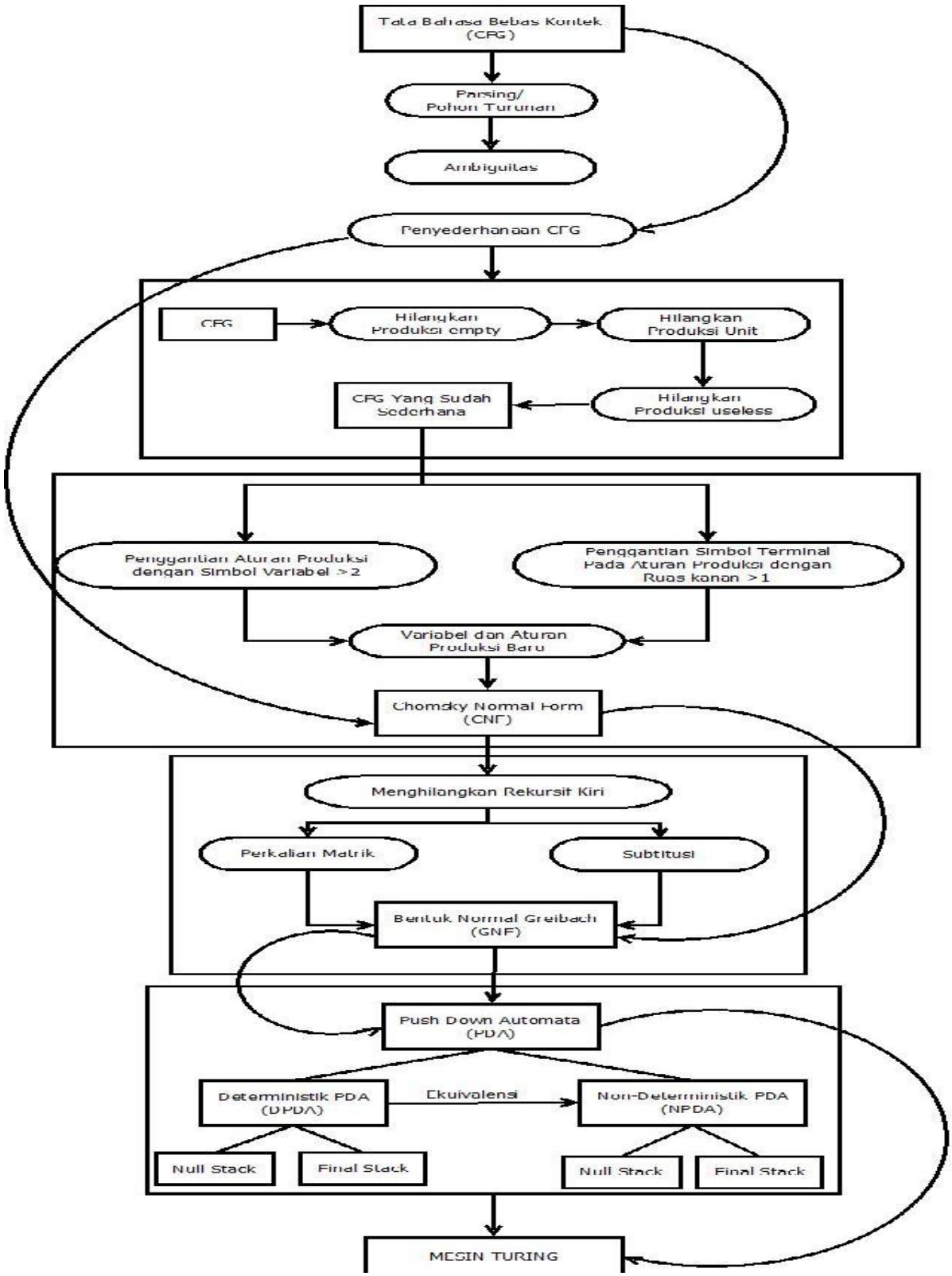
[29] Yakaryılmaz A., 2012, Superiority Of One-Way And Realtime Quantum Machines∗,∗∗, Rairo-Theor. Inf. Appl. 46 (2012) 615–641, Doi:10.1051/Ita/2012018.

[30] Chomsky N., 1959, *On Certain Formal Properties of Grammars**, Information and Control 2,137-167, 1959.

[31] Condon A., Hellerstein L., Pottle S., And Wigderson A., 1981, Languages Simultaneously Complete For One-Way And Two-Way Log-Tape Automata*, Siam J. Comput., Vol. 27, No. 3, Pp. 739-762, Mei 1981.

[32] Nugroho A.S., 2013, Teori Bahasa & Otomata, Graha Ilmu, SBN 978-602-262-011-2.

[33] Natarajan A.M., Tamilarasi A., Balasubramani P., 2003, Theory of Computation, New Age International (P) Limited, ISBN 81-224-1473-7, 2003.



Gambar 13. Peta perkembangan bahasa bebas konteks