MODUL 17. BAHASA-BAHASA REKURSIF DAN *RECURSIVELY ENUMERABLE*

Dalam pembahasan sebelumnya kita mendapatkan bahwa terdapat sejumlah bahasa yang mana fungsi karakteristik χ_L dari bahasa tersebut dapat didefinisikan, sementara ada sejumlah lain yang tidak. Pada suatu bahasa dalam Σ maka χ_L dapat didefinisikan jika dan hanya jika setiap string $x \in \Sigma^*$ dapat diputuskan oleh suatu TM apakah ia anggota dari L atau tidak, tanpa pernah TM mengalami *loop forever*.

Kenyataan ini menjelaskan mengapa secara prinsip kitabedakan antara TM yang hanya dapat **menerima** bahasa L dan TM yang data **mengenali** bahasa L.

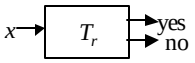
Definisi 10.1. Diketahui $L \subseteq \Sigma^*$ merupakan suatu bahasa dan T adalah suatu TM dengan alfabet masukan Σ.

- T dikatakan menerima (*accept*) L bila $L(T) = L$ dan T dikatakan mengenali (*recognize* atau *decide*) L bila T dapat mengkomputasi suatu fungsi karakteristik χ_L: $\Sigma^* \rightarrow \{0, 1\}$. Dengan kata lain, T mengenali L bila T **halt** untuk setiap string x dalam Σ^*, dan menghasilkan keluaran 1 bila $x \in L$, atau keluaran 0 bila tidak.
- Suatu bahasa L adalah *recursively enumerable* (RE) bila terdapat TM yang menerima L dan disebut rekursif bila terdapat TM yang mengenali L.

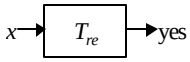
Setiap bahasa rekursif adalah juga RE tetapi tidak semua bahasa RE adalah rekursif. Karena, bila T adalah TM yang mengenali L maka T dapat dimodifikasi menjadi T_1 sehingga akan selalu crash T_1 bila T menghasilkan keluaran 0. Bila T adalah TM yang menerima L, akan ada string-string yang bukan anggota L sehingga T tidak menghasilkan jawaban.

Untuk memahami definisi di atas maka suatu TM yang mengenali bahasa rekursif dan bahasa RE masing-masing digambarkan dalam diagram-diagram sebagai berikut.

TM T_r untuk suatu bahasa rekursif akan menjawab "yes" (recognize) atau "no" setelah memproses string masukan x.



TM T_{re} untuk suatu bahasa RE hanya akan menjawab "yes" setelah memproses string masukan x dan halt; jika mengalami crash ditengah proses tsb atau loop forever maka ia tidak memberikan output.

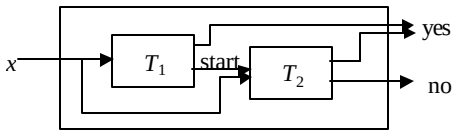


Teorema 10.1. Bila L diterima oleh suatu TM nondeterministik, dan setiap deretan move yang mungkin mengakibatkan T halt atau crash maka L disebut rekursif.

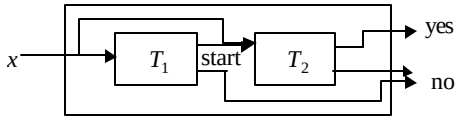
Menurut Teorema 9.2, jika L diterima oleh suatu NTM maka akan ada TM deterministik yang juga dapat menerima L. Jika L diterima tanpat terjadi loop forever oleh NTM tsb maka bukti Teorema 9.2 akan menghasilkan hal yang sama pada TM deterministik 3-tape yang mensimulasikannya.

Teorema 10.2a. Bila L_1 dan L_2 merupakan bahasa-bahasa rekursif pada alfabet masukan Σ, maka $L_1 \cup L_2$ serta $L_1 \cap L_2$ adalah juga rekursif.

Misalkan T_1 dan T_2 masing-masing adalah mesin untuk mengenali L_1 dan L_2. Kita membuat T_0 untuk mengenali $L_1 \cup L_2$ yang bekerja sbb. Jika $x \in L_1$ berarti x akan dikenali oleh T_1 dan halt, maka T_0 juga akan mengenali x dan halt sebagaimana T_1. Jika $x \notin L_1$ berarti T_1 tidak mengenali x dan halt, sementara T_0 langsung meniru bekerjanya T_2. Maka untuk setiap $x \in L_1 \cup L_2$, T_0 akan selalu halt dengan mengenali x, dan untuk setiap $x \notin L_1 \cup L_2$, T_0 akan selalu halt dengan tidak mengenali x.

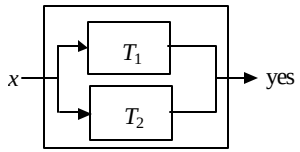


Seperti di atas kecuali T_0 akan halt dan tidak mengenal x jika T_1 juga halt dan tidak mengenal x (artinya jika $x \notin L_1$). Jika T1 halt dengan mengenal x maka T_0 langsung meniru T_2. Maka untuk setiap $x \in L_1 \cap L_2$, T_0 akan selalu halt dengan mengenali x, dan untuk setiap $x \notin L_1 \cap L_2$, T_0 akan selalu halt dengan tidak mengenali x.

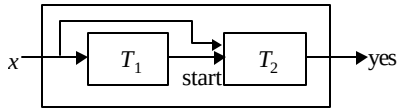


Teorema 10.2b. Bila L_1 dan L_2 merupakan bahasa-bahasa RE pada alfabet masukan Σ, maka $L_1 \cup L_2$ serta $L_1 \cap L_2$ adalah juga RE.

Jika T_1 dan T_2 masing-masing L_1 dan L_2 adalah TM yang menerima L_1 dan L_2. Kita buat T_0 TM 2-tape yang mensimulasikan T_1 dan T_2 pada masing-masing tape. Pada setiap move T_1 dan T_2 maka T_0 melakukan dua move (satu pada tape pertama dan satu pada tape kedua). Apabila salah satu dari T_1 dan T_2 halt maka T_0 juga halt, jika salah satu dari T_1 dan T_2 crash maka T_0 tetap jalan hanya melakukan move seperti pada mesin yang tidak crash. Dengan ternyata tidak ada yang halt, maka T_0 akan loop forever atau crash. Jika T_0 halt untuk x masukan maka $x \in L_1 \cup L_2$. Untuk $x \notin L_1 \cup L_2$, maka T_0 akan crash atau loop forever.

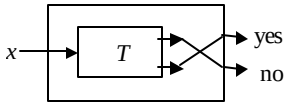


Dengan penjelasan yang mirip, berikut ini untuk menunjukkan ($L_1 \cap L_2$) adalah juga rekursif (RE).



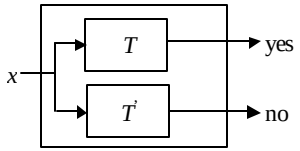
Teorema 10.3. Bila L rekursif maka L' juga rekursif.

Bila TM T mengenali L, kita dapat membuatnya (menjadi T') mengenali L' dengan menukarkan kedua harga output.



Teorema 10.4. Bila L adalah bahasa RE dan L memiliki komplemen yang juga RE, maka L adalah rekursif.

Jika TM yang menerima L tersebut adalah T dan TM yang menerima L' adalah T'. Untuk setiap $x \in L$, T akan halt dan untuk setiap $x \notin L$ maka T' akan halt. Suatu TM T_2 dapat mensimulasikan kedua mesin ini, untuk $x \in L$ (yang menyebabkan T halt) maka T_2 akan menghasilkan keluaran 1 dan untuk $x \notin L$ (menyebabkan T' halt) maka T_2 akan menghasilkan keluaran 0. Dengan adanya T_2 maka L adalah rekursif.



Enumarasi suatu Bahasa

Mengenumerasi suat himpunan adalah mencacah elemen-elemen satu demi satu. Jadi suatu himpunan dikatakan enumerabel adalah apabila terdapat suatu algoritma yang dapat mengenumerasikannya.

Definisi 10.2. T adalah suatu TM k-tape dengan $k \geq 2$ yang mengenumerasi $L \subseteq \Sigma^*$ bila T dalam bekerjanya memenuhi kondisi-kondisi sbb.

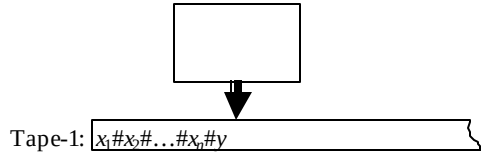
1. Head dari tape-1 tidak akan pernah bergerak ke kiri.
2. Pada setiap saat bekerjanya T, tape-1 memiliki isi

$x_1\#x_2\#\ldots\#x_n\#y$

dengan y suatu prefiks salah satu anggota L.

3. Untuk setiap $x \in L$, x akhirnya akan menjadi salah satu x_i pada tape-1 ybs.

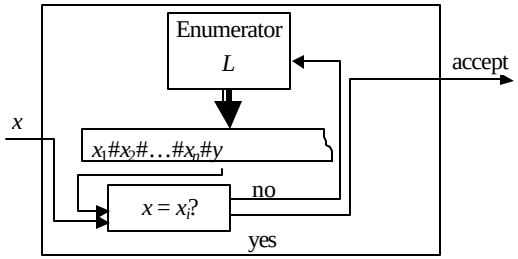
Bila himpunan L berhingga maka T akan halt ketika seluruh anggota L muncul dalam tape-1, atau akan terus bergerak tanpa melakukan penulisan string berikutnya pada tape-1. Bila L tak berhingga tentu T akan terus menuliskan string-string anggota L tanpa akhir.



Kembali pada bahasa Recursively Enumerable (RE) yaitu bahasa yang dapat diterima oleh sejumlah TM. Penamaan bahasa ini demikian berkaitan dengan sifat berikut ini.

Teorema 10.5. Suatu bahasa $L \subseteq \Sigma^*$ adalah RE (jadi dapat diterima oleh sejumlah TM) jika dan hanya jika L dapat dienumerasi oleh sejumlah TM.

Jika L dapat dienumerasi oleh TM T maka suatu TM T_1 dapat dibuat mensimulasikan T dalam mengenumerasi L. Selain itu T_1 memiliki satu tape tambahan untuk menyimpan string masukan x. Pada saat T (dan juga T_1) selesai menuliskan x_i dan simbol #, segera T_1 mundur ke # sebelumnya untuk membandingkan string x_i (string yang terakhir dituliskan pada tape) dengan string masukan x pada tape yang satu lagi. Jika sama maka T_1 halt.



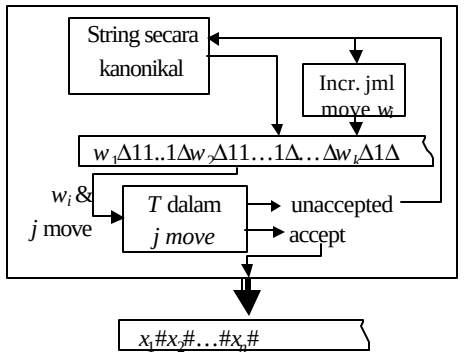
Untuk suatu $x \notin L$ maka T_1 tidak akan pernah menghasilkan output. Dengan demikian T_1 adalah suatu TM yang menerima L dan L adalah RE.

Sebaliknya, jika L adalah RE maka ada TM T yang dapat menerimanya. Kita buat suatu TM T_1 dengan 3-tape untuk dapat mengenumerasi L. Tape-1 untuk penulisan output enumerasi. Tape-2 digunakan T_1 untuk men-generate setiap string $w_i \in \Sigma^*$ dalam urutan kanonikal, dan tape-3 digunakan dalam simulasi T sebagai tape T. Pengurutan kanonis dari string-string dalam Σ^* adalah mengurutkan dari yang paling pendek ke yang paling panjang, dan dari antara yang panjangnya sama, diurutkan seprti mengurutkan kata dalam kamus.Contoh, pengurutan kanonis pada $\{0, 1\}^*$.

Λ, 0, 1, 00, 01, 10, 11, 000, 001, … , 111, 0000, …

Evaluasi suatu string $w_i \notin L$ secara tuntas akan membawa T_1 ke dalam infinite loop yang berakibat string-string w_j ($j > i$) berikutnya tidak pernah dievaluasi pada hal mungkin ada yang merupakan $\in L$. Untuk menghindari itu maka simulasi T pada w_i harus dilakukan dengan pembatasan jumlah move yang dilakukan dan T_1 melakukannya dalam sejumlah pass; pada pass ke-i T_1 melakukan:

- generate string w_i dan tuliskan pada tape-2 setelah w_1, w_1, … w_{i-1}
- simulasi dilakukan pada masing-masing string pada tape-2 dalam sejumlah move yang lebih banyak satu move dari pass sebelumnya.



Pada pass-1, T_1 mengenerate w_1 dan mensimulasi satu move T pada w_1. Pada pass-2, string w_2 digenerate dan dituliskan setelah w_1, kemudian melakukan simulasi dua move pada w_1 dan satu move pada w_2.
…
Pada pass ke-i T_1 mengenerate w_i dan dituliskan setelah w_{i-1}, kemudian melakukan simulasi i move pada w_1, i-1 move pada w_2, … , dan 1 move pada w_i.

Untuk mencatat jumlah move maka setiap string pada tape-2 akan diikuti suatu suatu string 1^k yang menunjukan pada pass berikutnya akan dilakukan simulasi dalam k move pada string ybs. Setiap bilangan ini akan bertambah satu saat suatu pass selesai dilalui. Blank digunakan sebagai separator antara bilangan dan string.

Contoh untuk $\Sigma = \{a, b\}$ dengan pengurutan kanonikalnya adalah Λ, a, b, aa, ab, ba, bb, … Dan tape-2 akan berisi sbb dari pass pertama dan seterusnya.
Pada pass ke 1: $\Delta\Delta 1$
pada pass ke 2: $\Delta\Delta 11\Delta a\Delta 1$
pada pass ke 3: $\Delta\Delta 111\Delta a\Delta 11\Delta b\Delta 1$
pada pass ke 4: $\Delta\Delta 1111\Delta a\Delta 111\Delta b\Delta 11\Delta ab\Delta 1$
…
pada pass ke i: $\Delta\Delta \underbrace{11\ldots1}_{i}\Delta a\Delta \underbrace{11\ldots1}_{i-1}\Delta b\Delta \underbrace{11\ldots1}_{i-2}\Delta\ldots\Delta x_i\Delta 1$

$\Delta\Delta 11\ldots111\Delta a\Delta 11\ldots1\Delta b\Delta 11\ldots1\Delta\ldots\Delta w_i\Delta 1\Delta$

(w_i adalah string ke i dalam pengurutan kanonikal $\{a, b\}$ yang baru digenerate pada pass ke i)

Simulasi T dalam jumlah move tertentu di atas dilakukan pada tape-3. Apabila suatu simulasi string w_i menghasilkan halt pada T maka T_1 menuliskan x ke tape-1 diikuti simbol #. Untuk tidak mengulangi pemeriksaan w_i yang telah dituliskan di tape-1 maka banyak pilihan dapat dilakukan, misalnya dengan menghilangkan string (serta bilangan move-nya) dari tape-2.

Teorema 10.6. L adalah rekursif jika dan hanya jika terdapat suatu TM yang mengenumerasikannya dalam pengurutan kanonikal.

Jika T mesin yang dapat mengenumerasi L secara kanonikal maka dapat dibuat T_1 yang membaca masukan x lalu dengan mensimulasi T ia men-generate secara kanonikal string-string $x_1, x_2, \ldots$ anggota L dan membandingkan setiap string yang digenerate dengan x. Saat $x = x_i$ maka T_1 berhenti dengan output 1. Sementara karena urutannya kanonikal maka pada saat x seharusnya berada di antara x_{i-1} dan x_i ($x \neq x_{i-1}$ dan $x \neq x_i$) maka T_1 berhenti dengan output 0.

Kebalikannya, jika L bahasa rekursif dan T mesin yang mengenalinya, maka mesin T_1 3-tape dibentuk yang dapat men-generate string demi string $w_1, w_2, \ldots$ anggota Σ^* dalam urutan kanonis. Setiap w_i digenerate, mesin T_1 mengevaluasinya dengan mensimulasikan T pada tape-3, jika T menjawab 1 maka T_1 menuliskan w_i dan # pada tape-1 jika T menjawab 0 maka T_1 melanjutkan mengenerate string w_{i+1}, dan seterusnya. Di sini teknik simulasi dalam jumlah move terbatas pada pembuktian Teorema 10.5 tidak diperlukan karena T akan selalu menghasilkan jawaban.

Adakah Bahasa yang Bukan RE?

Harusnya ada karena kalau tidak ada maka bahasa RE yang nonrekursif tidak pernah ada juga. Mengapa? Karena setiap bahasa RE akan memiliki komplemen yang juga RE dan ini juga berarti bahasa-bahasa RE tersebut adalah bahasa rekursif. Sesuai dengan Teorema 10.4.

Namun kalau memang ada ini memberikan implikasi komputasional yang sangat penting. Menurut thesis Church-Turing kalau ada suatu bahasa yang tidak dapat diterima oleh suatu TM maka ia tidak dapat dipecahkan juga oleh algoritma mana pun. Jadi, jika anda menghadapi suatu masalah komputasional maka bisa jadi masalah tersebut tidak akan dapat terpecahkan secara teoritis karena termasuk non-RE.

Suatu pembahasan teoritis dapat membuktikan bahwa tidak semua bahasa adalah RE namun tidak di bahas disini. Yang akan di bahas berikut ini adalah suatu contoh bahasa yang bukan RE.

Jika $e(T)$ pengkodean dalam TM universal pada TM T maka dalam semua kemungkinan string dalam Σ^* terdapat partisi:
SA = $\{w \in \{0, 1\}^* \mid w = e(T)$ untuk semua TM T dan T menerima $w\}$
NSA1 = $\{w \in \{0, 1\}^* \mid w = e(T)$ untuk semua TM T dan T tidak menerima $w\}$
NSA2 = $\{w \in \{0, 1\}^* \mid w \neq e(T)$ untuk TM T mana pun$\}$

SA (*self accepting*) dan NSA (*non self-accepting*) yang mana NSA = NSA1 + NSA2, adalah komplementer. Jika SA adalah RE, NSA ternyata bukan RE!

Hal ini dapat dibuktikan dengan kontradiksi. Jika NSA adalah RE maka terdapat TM T_0 dengan $L(T_0)$ = NSA. T_0 dalam hal ini menerima masukan string w jika dan hanya jika $w \in$ NSA. Kita evaluasi dengan $w_0 = e(T_0)$ apakah T_0 menerima w_0? Menurut definisi T_0 tidak menerima w_0 yang berarti juga bahwa w_0 anggota dari SA. Namun kenyataan bahwa $w_0 = e(T_0)$ dan T_0 tidak menerima w_0 menunjukkan bahwa w_0 adalah anggota dari NSA1. Jadi kontradiktif dan asumsi bahwa $L(T_0)$ adalah RE adalah salah.

Unrestricted Grammar

Jika bahasa *context free* dapat dispesifikasikan dengan CFG bagaimana dengan bahasa-bahasa yang diterima oleh TM? Secara umum bahasa-bahasa tersebut dapat dispesifikasikan oleh Unrestricted Grammar.

Definisi 11.1. **Unrestricted Grammar** (UG) atau **Grammar Berstruktur Frasa** adalah 4-tuple $G = (V, \Sigma, S, P)$, dengan V dan Σ adalah himpunan-himpunan disjoin masing-masing dari variabel dan terminal. S anggota dari V sebagai start symbol, dan P adalah himpunan produksi dalam bentuk

$$a \rightarrow b$$

dengan a , $b \in (V \cup \Sigma)^*$ dan a berisi minimal satu variabel.

Contoh: Untuk $L = \{a^i b^i c^i \mid i \geq 1\}$ suatu RG dapat dibuat dengan produksi sbb.

$S \rightarrow FS_1$ $\quad$ $FA \rightarrow a$
$S_1 \rightarrow ABCS_1 \mid ABC$ $\quad$ $aA \rightarrow aa$
$BA \rightarrow AB$ $\quad$ $aB \rightarrow ab$
$CA \rightarrow AC$ $\quad$ $bB \rightarrow bb$
$CB \rightarrow BC$ $\quad$ $bC \rightarrow bc$
$\quad$ $\quad$ $cC \rightarrow cc$

Masukan string *aabbcc* dapat diterima sdengan penurunan sbb.

$S \Rightarrow FS_1 \Rightarrow FABCS_1 \Rightarrow FABCABC \Rightarrow FABACBC$
$\Rightarrow FAABCBC \Rightarrow FAABBCC \Rightarrow aABBCC \Rightarrow aaBBCC$
$\Rightarrow aabBCC \Rightarrow aabbCC \Rightarrow aabbcC \Rightarrow aabbcc$

Teorema 11.1. Bila $G = (V, \Sigma, S, P)$ adalah UG maka terdapat mesin Turing $T = (Q, \Sigma, \Gamma, q_0, d)$ dengan $L(T) = L(G)$.

Teorema 11.2. Jika $L \subseteq \Sigma^*$ adalah bahasa RE, maka terdapat UG yang dapat menghasilan L.

Grammar dan Bahasa Context Sensitive

Suatu kelas bahasa di antara CFG dan RE adalah Bahasa Context Sensitive (CSL) yang di spesifikasikan oleh Grammar Context Sensitive (CSG).

Definisi 11.3. Suatu Grammar *context-sensitive* (CSG) adalah merupakan unrestricted grammar dengna setiap produksi memiliki bentuk

$$a \rightarrow b \quad \text{dengan } |a| \leq |b|$$

Suatu bahasa *context-sensitive* (CSL) adalah bahasa yang dapat dihasilkan dari penurunan CSG.

Contoh:
Pada contoh sebelumnya bahasa $L = \{a^i b^i c^i \mid i \geq 1\}$ dispesifikasikan oleh UG yang bukan CSG. Namun, ternyata L adalah CSL karena suatu CSG dapat ditemukan dengan memodifikasi variabel F dan variabel A pertama, sbb.

$S \rightarrow A_1BCS_1 \mid A_1BC$ $\quad$ $A_1 \rightarrow a$

$S_1 \rightarrow ABCS_1 \mid ABC$ $\quad aA \rightarrow aa$
$BA \rightarrow AB$ $\quad aB \rightarrow ab$
$CA \rightarrow AC$ $\quad bB \rightarrow bb$
$CB \rightarrow BC$ $\quad bC \rightarrow bc$
$\quad cC \rightarrow cc$

Sebagaimana dengan bahasa-bahasa, mesin yang dapat menerima CSL adalah LBA (Linear Bounded Automata).

Definisi 11.4. Suatu LBA adalah 5-tuple $M = (Q, \Sigma, \Gamma, q_0, d)$ yaitu suatu NTM dengan keterbatasan berikut ini. Terdapat dua ekstra simbol tape $\langle$ dan $\rangle$ yang bukan elemen Γ. NTM M mulai dari konfigurasi $(q_0, \langle x \rangle)$ dengan $\langle$ pada kotak-0 tape dan $\rangle$ pada kotak berikutnya setelah simbol terakhir x. Dalam beroperasinya M tidak boleh mengganti simbol $\langle$ atau $\rangle$ pada kotaknya, serta tidak boleh bergerak ke sebelah kiti kotak yang berisi $\langle$ dan ke sebelah kanan kotak yang berisi $\rangle$.

Jadi dengan kata lain simbol-simbol $\langle$ dan $\rangle$ memastikan bahwa string dalam tape tidak akan pernah panjangnya lebih dari $|x|$.

Teorema 11.4. Bila $L \subseteq \Sigma^*$ merupakan CSL, maka terdapat suatu LBA yang menerima L.

Teorema 11.5. Bila terdapat suatu LBA $M = (Q, \Sigma, \Gamma, q_0, d)$ yang menerima bahasa $L \subseteq \Sigma^*$ maka terdapat suatu CSG yang menghasilkan $L - \{\Lambda\}$.

Teorema 11.6. Setiap CSL adalah bahasa rekursif.

Teorema 11.7. Terdapat bahasa rekursif L pada $\Sigma = \{a, b\}$ sehingga $L - \{\Lambda\}$ yang bukan context-sensitive.

Hirarki Chomsky

Chomsky pada tahun 1956-1959 memperkenalkan empat type bahasa dalam hirarki bahasa: bahasa type 3, bahasa type 2, bahasa type 2, dan bahasa type 0.

Berikut ini tabel hirarki beserta karakteristiknya yang dinyatakan dengan kelas grammar, mesin abstrak, dan kodel komputasinya.

Type	**Bahasa/Grammar**	**Bentuk produksi dalam Grammar**	**Mesin Penerima**
3	Regular (Finite State Grammar)	$A \rightarrow aB, A \rightarrow a$ $(A, B \in V, a \in \Sigma)$	Finite Automaton
2	Context-free	$A \rightarrow a$ $(A \in V, a \in (V \cup \Sigma)^*$	Pushdown Automaton
1	Context-sensitive	$a \rightarrow b$ $(a, b \in (V \cup \Sigma)^*, \lvert a \rvert \leq \lvert b \rvert$ dan a min. 1 variabel	Linear-bounded Automaton
0	Recursively Enumerable (unrestricted)	$a \rightarrow b$ $(a, b \in (V \cup \Sigma)^*,$ dan a min. 1 variabel	Turing Machine

Catatan: bahasa rekursif adalah himpunan bagian dari bahasa RE serta di atas RE masih terdapat bahasa-bahasa non RE yaitu contohnya NSA.