Turing Machines

Kuliah Teori Bahasa dan Automata
Program Studi Ilmu Komputer
Fasilkom UI

Prepared by:
Suryana Setiawan

Spesifikasi: 2 sifat penting

- Mesin Turing adalah pembanding setiap model komputasi lain.
  - Power : jangkauan bahasa-bahas yang dapat dikenali.
    - FSM ➔ bahasa-bahasa reguler
    - PDA (FSM dengan *stack*) ➔bahasa-bahasa CF
    - Mesin Turing (FSM dengan *tape*) ➔ ???
- Mampu mendeskripsikan segala komputasi.
  - Setingkat komputer tetapi tidak seterbatas FSM/PDA
- Sederhana, agar penjelasan formal dapat dilakukan.
  - Sesederhana FSM/PDA, tapi tidak seperti sekompleks komputer

Manfaat dan Masalah Stack

- Adanya stack meningkatkan secara signifikan kemampuan suatu FSM, dari hanya mengenali Bahasa Reguler ke bahasa Context Free.
  - Namun, mekanisme LIFO pada stack masih membatasi kemampuan komputasinya.
- Perlu struktur yang menggantikan stack sehingga
  - memungkinkan mengakses isi storage tersebut secara lebih fleksibel.
  - memungkinkan akses suatu data dalam storage tanpa mengganggu data pada posisi lainnya.

Definisi Intuitif: Turing Machines

- Bayangkan suatu FSM yang dilengkapi storage berbentuk tape
  - **Tape** berbentuk **linear**, setiap posisi, atau **square**, terurut dari kiri ke kanan,
  - Setiap posisi dapat menyimpan **satu simbol tape** atau **kosong (□)** atau ***blank***,
  - **Kapasitas** (panjang tape) tak berhingga (tidak berujung baik di kiri maupun di kanan).
- **Read/write** dilakukan melalui sebuah **head**
  - Head bergerak secara sikuensial dari satu posisi ke posisi berikutnya (arah R)/sebelumnya (arah L).
  - Dalam kuliah ditambahkan gerakan S, yaitu tetap berada di tempat (agar sesuai dengan simulator JFlap).

Mekanisme Komputasi

- Dengan adanya tape, input string bisa diasumsikan **sudah langsung ditaruh di dalam tape**, sehingga mesin segera bekerja pada tape.
- Di konfigurasi awal dibuat **konvensi**: head berada di **posisi kosong tepat sebelum simbol terkiri string input** pada tape.
- Komputasi dilakukan menurut **current state**, **current data** tape (pada head), untuk bertransisi ke **next-state**, meng-**update isi tape** (pada head), **arah pemindahan head**.
- Status akhir saat mencapai **halting state** atau **crash** (tidak ada transisi yang bisa dilakukan lagi).

Definisi Formal: Turing Machines

- Suatu Turing Machine M adalah 6-tuple $(K, \Sigma, \Gamma, \delta, s, H)$, dimana:
  - ✓ K: himpunan terbatas **status**.
  - ✓ Σ: **alfabet input** yang tidak berisi □
  - ✓ Γ: **alfabet tape**, termasuk Σ dan □
  - ✓ s: **status mulai**
  - ✓ H: himpunan **status halting**, $H \subseteq K$
  - ✓ δ: **fungsi transisi**, yang memetakan:

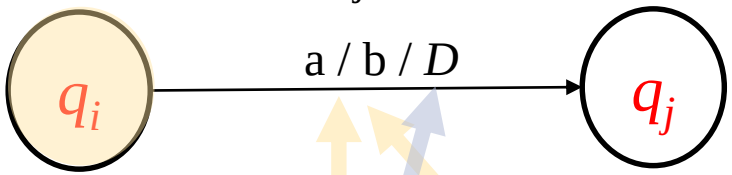
    $(K - H) \times \Gamma \rightarrow K \times \Gamma \times \{R, L, S\}$
- Note: definisi TM deterministik

Fungsi Transisi δ

$$\delta: (K–H) \times \Gamma \rightarrow K \times \Gamma \times \{R, L, S\}$$

Non-halting states | Tape-symbol | Next states | Tape-symbol | Head movement

- Suatu transisi $((q_i, a), (q_j, b, D)) \in \delta$ menyatakan



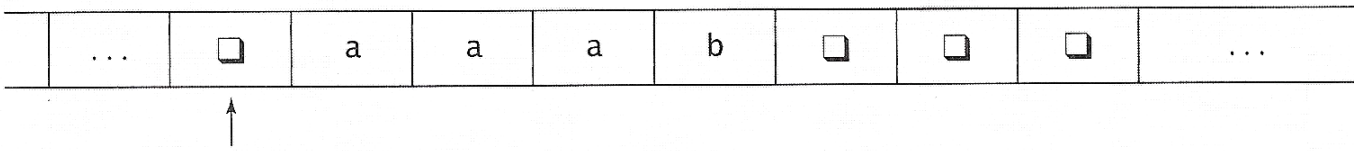
  - dari **current state q_i** bertransisi ke **next state q_j**,
  - saat head **membaca a** pada posisi head, mengubah isi tape (pada posis head) **menjadi b**,
  - **bergerak ke arah *D***, yaitu satu posisi ke kiri (L) atau ke kanan (R) atau tetap (S).

Pengertian "simbol □"

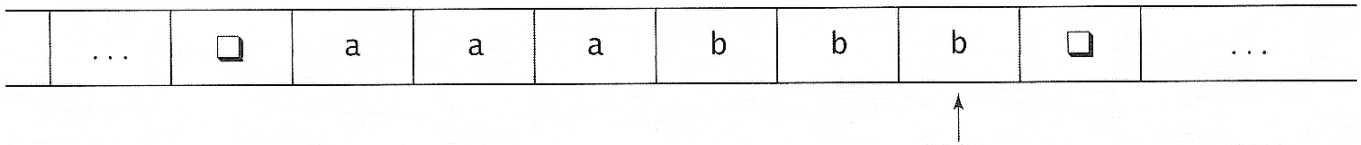
- Setiap posisi tape "berisi satu simbol tape" $x \in \Gamma$ ($x \neq \square$), atau "kosong".
  - Jika disebut kosong, maka kenyataan yang sebenarnya, posisi tape tersebut berisi simbol "□" (blank).
  - Jika disebutkan tape hanya berisi string α, maka kenyataan yang sebenarnya, yang lainnya berisi "□".
- **Active tape**: isi tape dari non-blank terkiri hingga non-blank terkanan, di tambah beberapa blank di luar itu yang terkait beroperasinya mesin tsb.
  - Selama komputasi, bagian tape yang kosong yang lain tidak perlu diperhatikan.

Contoh (Deskripsi Masalah)

- TM *M* yang dapat memproses input string dari $\{a^i b^j : 0 \leq j \leq i\}$ dengan menambahkan sejumlah b di belakang string agar menjadi $a^i b^i$.
- Saat mulai, isi tape sebagai berikut:



- Saat halt isi tape menjadi:



- Tanda panah menunjukkan posisi head
- Untuk contoh ini string input <u>diasumsikan selalu benar</u> $\{a^i b^j : 0 \leq j \leq i\}$

Contoh (Operasi pada *M*)

- Pindahkan head ke kanan satu posisi, jika simbol di bawah head adalah □, maka halt
- Dalam loop:
  - Tandai (sebenarnya ganti) setiap a dengan $.
  - Scan ke kanan menemukan b atau □
    - Jika b, tandai (sebenarnya ganti) dengan #, kemudian siap balik ke kiri.
    - Jika □, berarti b habis, tetapi masih ada a tersisa, maka tuliskan # dan siap balik ke kiri.
    - Balik ke kiri untuk menemukan a atau □,
      - jika a kembali ke awal loop,
      - jika □, semua a sudah ditangani, maka halt.
- Lakukan pass terakhir untuk mengganti $ ke a, dan # ke b.

Contoh (Mesin *M*)

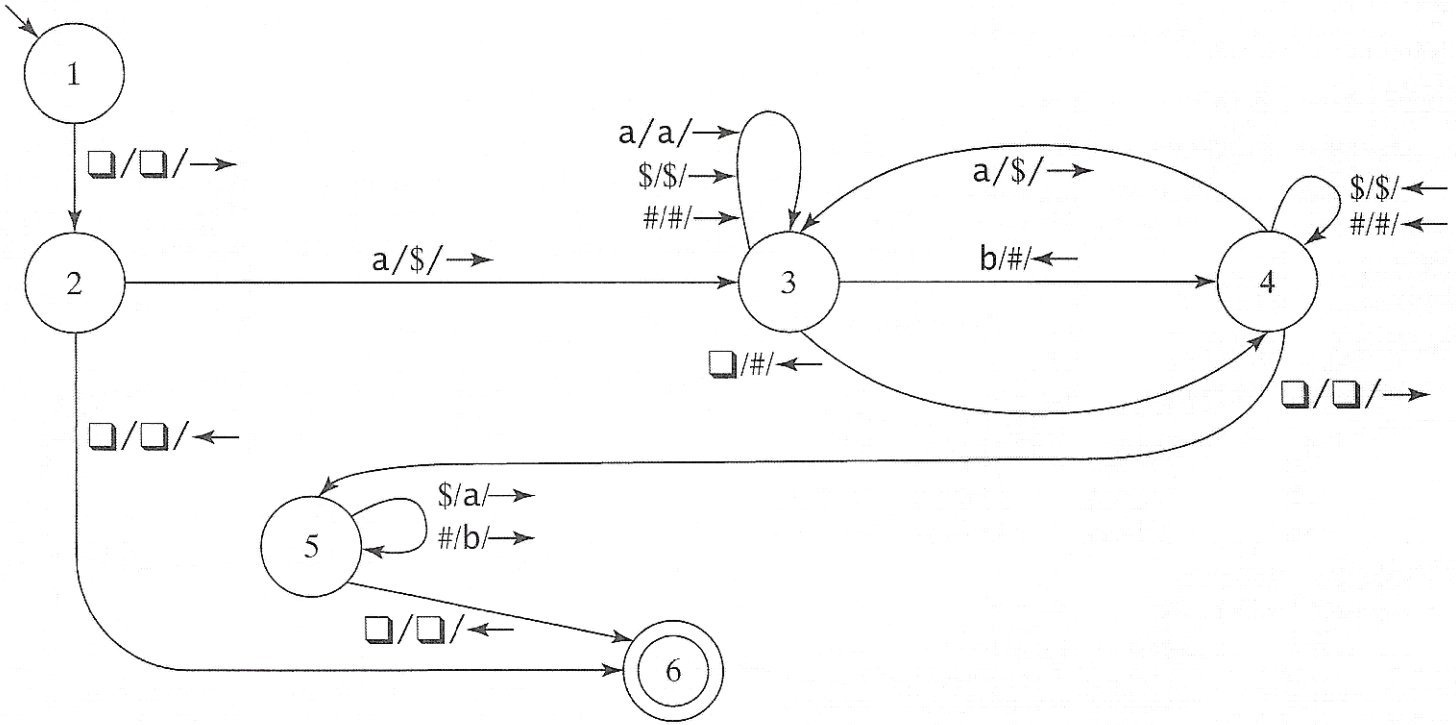
- M = ({1,2,3,4,5,6}, {a,b}, {a,b, □, $, #}, δ, 1, {6}), dengan δ = {

  ((1, □), (2, □, →)), ((1, a), (2, q, →)), ((1, b), (2, q, →)), ((1, $), (2, $, →)), ((1, #), (2, #, →)), ((2, □), (6, $, →)), ((2, a), (3, $, →)), ((2, b), (3, $, →)), ((2, $), (3, $, →)), ((2, #), (3, $, →)), ((3, □), (4, #, ←)), ((3, a), (3, a, →)), ((3, b), (4, #, ←)), ((3, $), (3, $, →)), ((3, #), (3, #, →)), ((4, □), (5, □, →)), ((4, a), (3, $, →)), ((4, $), (4, $, ←)), ((4, #), (4, #, →)), ((5, □), (6, □, →)), ((4, a), (3, $, →)), ((5, $), (5, a, ←)), ((5, #), (5, b, →))

  }

- Note: 6 merupakan halting state maka transisi untuk 6 tidak perlu didefinisikan.

Contoh (Diagram mesin M)



Konfigurasi

- Setiap saat komputasi **konfigurasi** mesin dinyatakan sebagai (q, $\alpha\underline{x}\beta$).
  - *Current state q.*
  - Isi *active tape* string $\alpha\underline{x}\beta$:
    - $\underline{x}$ → simbol tape pada posisi *head.*
    - α → string isi tape di kiri (*prefix*) posisi *head.*
    - β → string isi tape di kanan (*suffix*) posisi *head.*
- **Konfigurasi awal**: (s, $\underline{\square}\gamma$), start state s, dan head berada pada kotak kosong tepat disamping kiri string.
- **Halting configuration**: jika q merupakan status *halting*.

Yields dan Komputasi

- **Yield in one step**: untuk mesin TM *M*, relasi antara konfigurasi C_1 dan C_2 dimana C_2 dicapai dari C_1 setelah satu kali transisi,
  ditulis $C_1 \vdash_M C_2$.
- **Yields**: untuk mesin TM *M*, reflexive, transitive closure dari $\vdash_M$, ditulis $\vdash^*_M$
- **Path**: dari mesin TM *M*, adalah suatu sikuens $C_0, C_1, C_2, \ldots$ dengan C_0 adalah **konfigurasi awal**, apabila terjadi $C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots.$
- **Komputasi**: dari mesin TM *M*, adalah suatu path $C_0, C_1, \ldots, C_n$, untuk $n \geq 0$, dengan Cn adalah **halting configuration**.
  - Disebut komputasi halt dalam n langkah, $C_1 \vdash_M^n C_n$

Contoh

- Untuk contoh menambahkan b sebelumnya, jika input string aaab , TM ybs menghasilkan komputasi:
- (1, □aaab□□) ├ (2, □aaab□□) ├ (3, □$aab□□)
  ├ (3, □$aab□□) ├ (3, □$aab□□) ├ (4, □$a#□□)
  ├ (3, □a#□□) ├ (3, □a#□□) ├ (4, □a##□)
  ├ (4, □a##□) ├ (4, □a##□) ├ (3, □$$$##□)
  ├ (3, □$$$##□) ├ (3, □$$$##□) ├ (3, □$$$##□)
  ├ (4, □$$$###) ├ (4, □$$$###) ├ (4, □$$$###)
  ├ (4, □$$$###) ├ (4, □$$$###) ├ (4, □$$$###)
  ├ (4, □$$$###) ├ (5, □$$$###) ├ (5, □a$$###)
  ├ (5, □aa$###) ├ (5, □aaa###) ├ (5, □aaab##)
  ├ (5, □aaabb#) ├ (6, □aaabbb□)

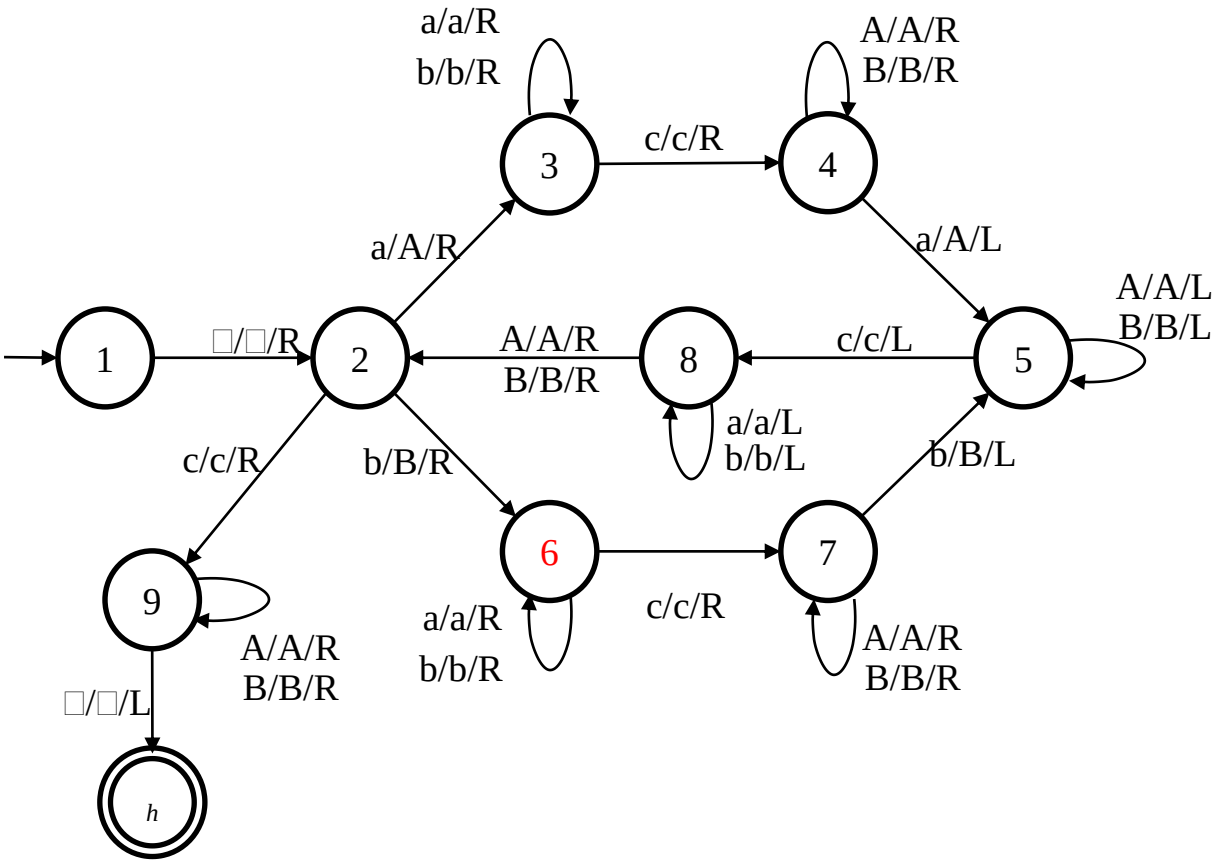
Fungsi Mesin Turing

- Reconizer: untuk menerima atau menolak.
- Sebagai pengkomputasi fungsi.
  - Mesin harus selalu halt karena output pada tape akan digunakan oleh mesin turing lain.
- Sebagai recognizer suatu bahasa.
  - Mesin harus **halt** untuk menerima string input dan **crash** untuk menolak string input.
  - Masin harus selalu halt, dengan **halt-yes** atau **halt-no**, bila output yes/no ini akan dilanjutkan oleh mesin turing lain.

Contoh: Recognizer WcW

- Recognizing WcW = {*wcw* : *w* ∈ {a,b}*}, accepting dengan halt, rejecting dengan crash.
- **Ide algoritma**: dalam iterasi
  - Dalam pencabangan, membaca suatu simbol di ruas kiri, lalu memeriksa di ruas kanan juga harus simbol yang sama.
  - Ruas kanan diketahui setelah melalui c saat scanning
  - Simbol yang sudah diperiksa diubah ke simbol lain agar tidak diperiksa dua kali
    - a menjadi A, b menjadi B
    - perpindahan head ke kiri sampai ketemu A atau B.
  - Iterasi berakhir jika terjadi mismatching atau seluruh simbol string sudah diperiksa (berubah menjadi simbol lain)
- Kasus yang perlu diperhatikan, jika panjang kedua ruas berbeda.

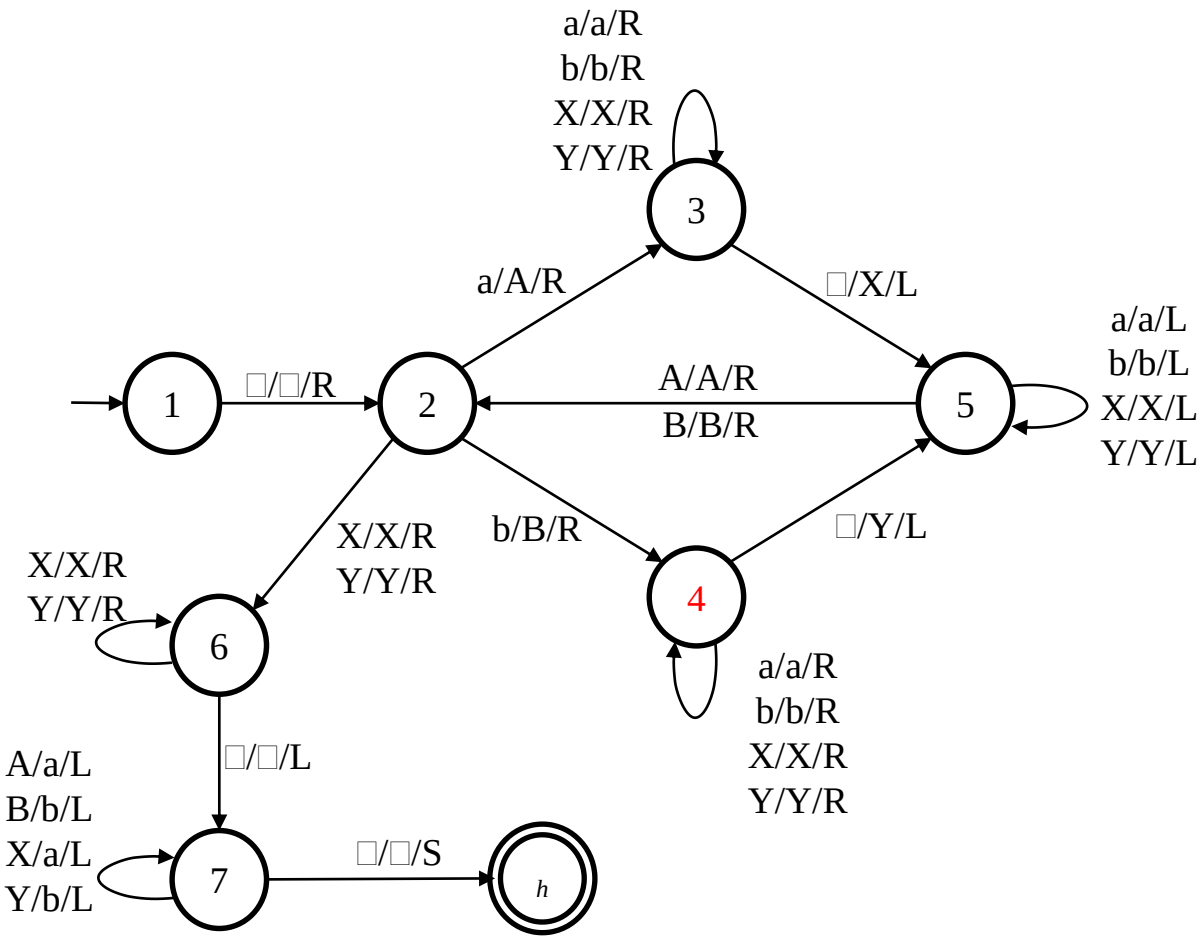
Contoh: Recognizer WcW



Contoh: Replikator String *w*

- input w dan dan output *ww*, $w \in \{a,b\}^*$.
- Konfigurasi awal (1,$\underline{\sqcup}w$) dan konfigurasi halt (h,$\underline{\sqcup}ww$).
- **Ide algoritma**: dalam iterasi
  - Jika c adalah a atau b, baca dan tandai tandai
    - a menjadi A, atau b menjadi B
  - lalu ke kanan mencari blank menuliskan symbol c',
    - Jika c=a maka c'=X
    - Jika c=b maka c'=Y
  - lalu ke kiri hinga ketemu symbol A atau B, ke kanan satu posisi
  - Jika sudah tidak ada lagi a atau b, ke kanan mencari blank
  - Lalu mundur ke kiri dalam iterasi ubah
    - setiap A menjadi a, B menjadi b, X, menjadi a, Y menjadi b,
  - hingga mencapai blank, lalu halt.

Contoh: Replikator String *w*



Halting, Crash atau Forever Loop

- Mesin **berhenti** jika:
  - Mencapai **status halt** (status khusus untuk menyatakan mesin berhenti).
  - Terpaksa berhenti (**crash**) karena transisi untuk konfigurasi saat ini (status dan isi tape pada head) tidak terdefinisi.
- Mesin bisa **tidak pernah berhenti (**karena *forever-loop*)
  - Tidak dapat mencapai halt maupun crash.
  - Karena **kesalahan logika** dalam pendefinisian transisi atau karena ***nature of its related problem*** sendiri (tidak ada mesin ekivalen yang bisa halt/crash).