Turing Machines (4)

Kuliah Teori Bahasa dan Automata
Program Studi Ilmu Komputer
Fasilkom UI

Prepared by:
Suryana Setiawan

Mesin Turing Multitape

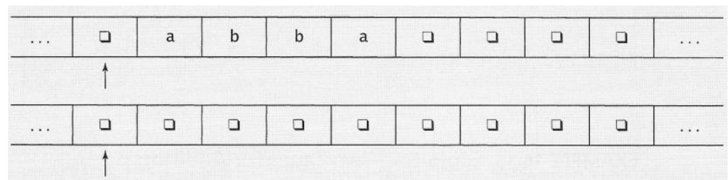
- Mesin Turing dengan k-tape (seperti halnya 1-tape) adalah 6-tuple $M = (K, \Sigma, \Gamma, \delta, s, H)$. Perbedaannya:
  - adanya tape sebanyak k buah,
  - masing-masing memiliki satu head sendiri,
  - konfigurasi mesin terdiri atas
    - status current, isi dari tape-tape tsb, dan posisi head setiap tape
  - Transisi merupakan fungsi

$$((K\text{-}H) \times \Gamma_1 \times \ldots \times \Gamma_k) \rightarrow (K \times \Gamma_1 \times \{\rightarrow, \leftarrow, \uparrow\} \times \ldots \times \Gamma_k \times \{\rightarrow, \leftarrow, \uparrow\})$$

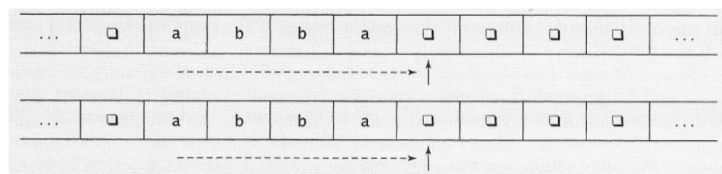
Note: untuk multi-tape terdapat pilihan gerakan head $\uparrow$ (atau S) yang artinya tetap pada posisi tsb.

TM-2T: Duplicator

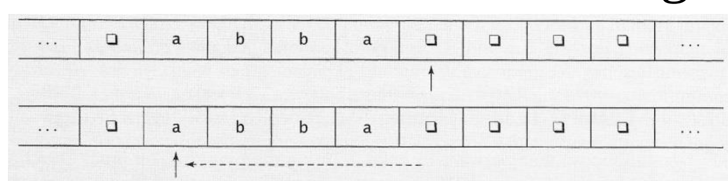
- Sebagai tempat sementara, contoh: operasi duplikasi string.
  - Di awal tape 1 berisi input string dan tape 2 kosong.



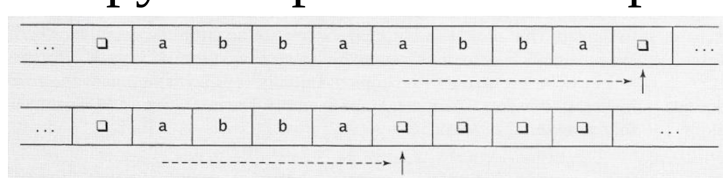
  - Isi tape 1 dicopy ke tape 2, simbol demi simbol.



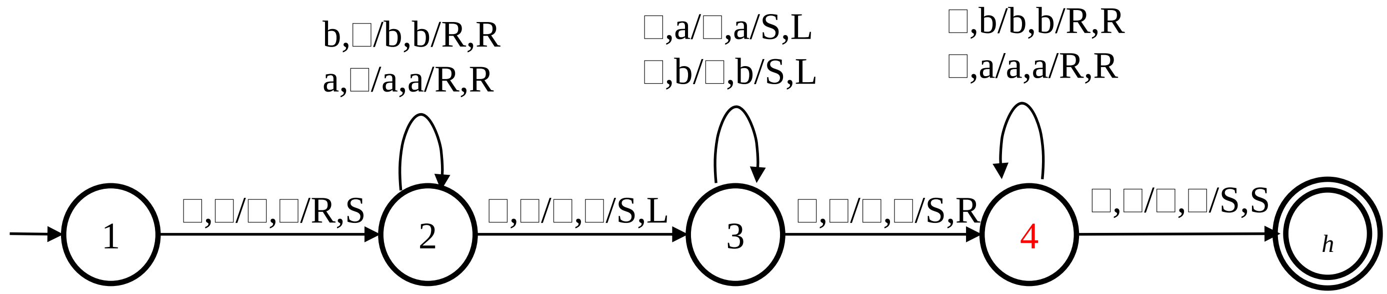
  - Head tape 2 di “rewind” ke awal string.



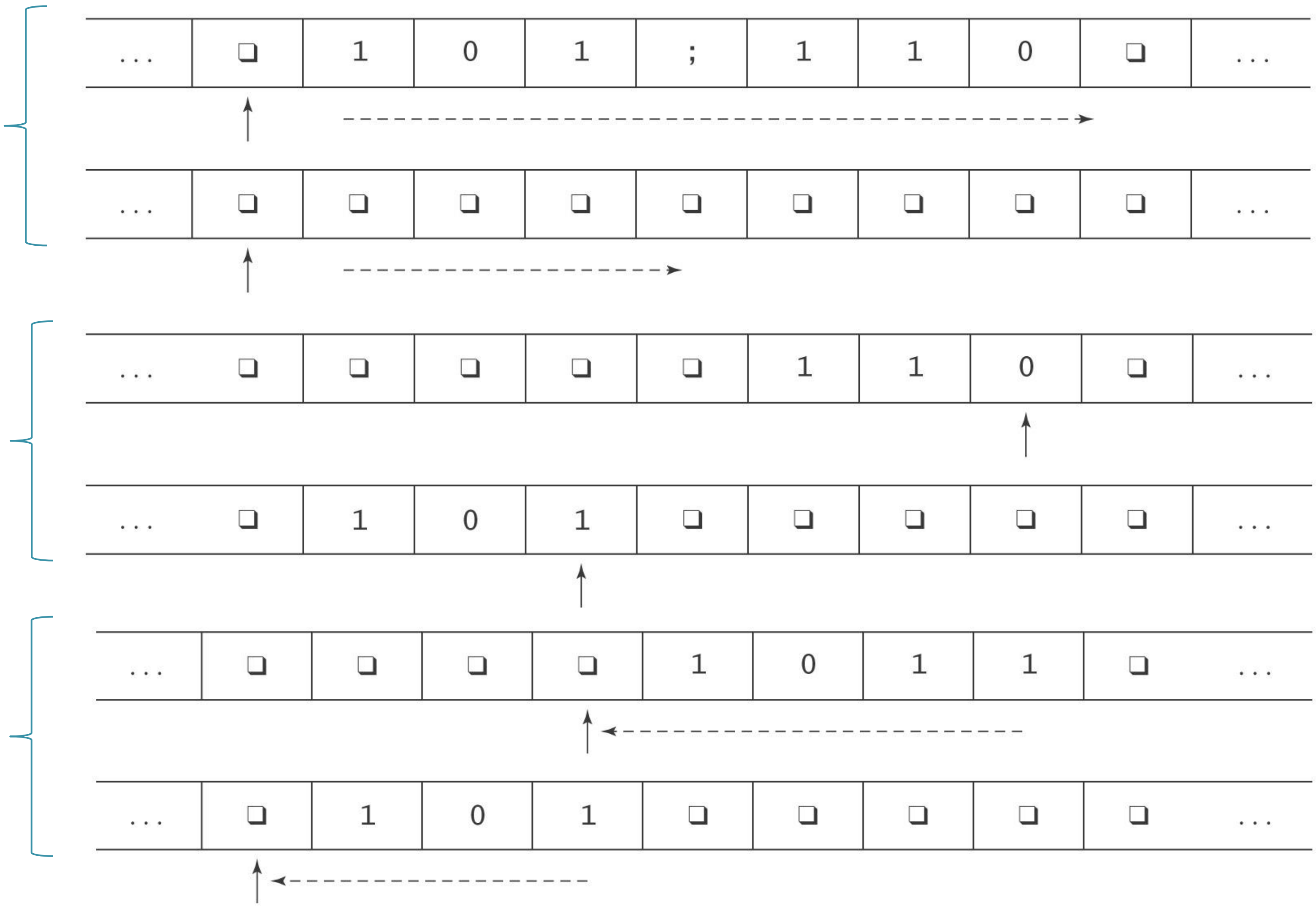
  - Isi tape 2 dicopy ke tape no 1 mulai posisi current.



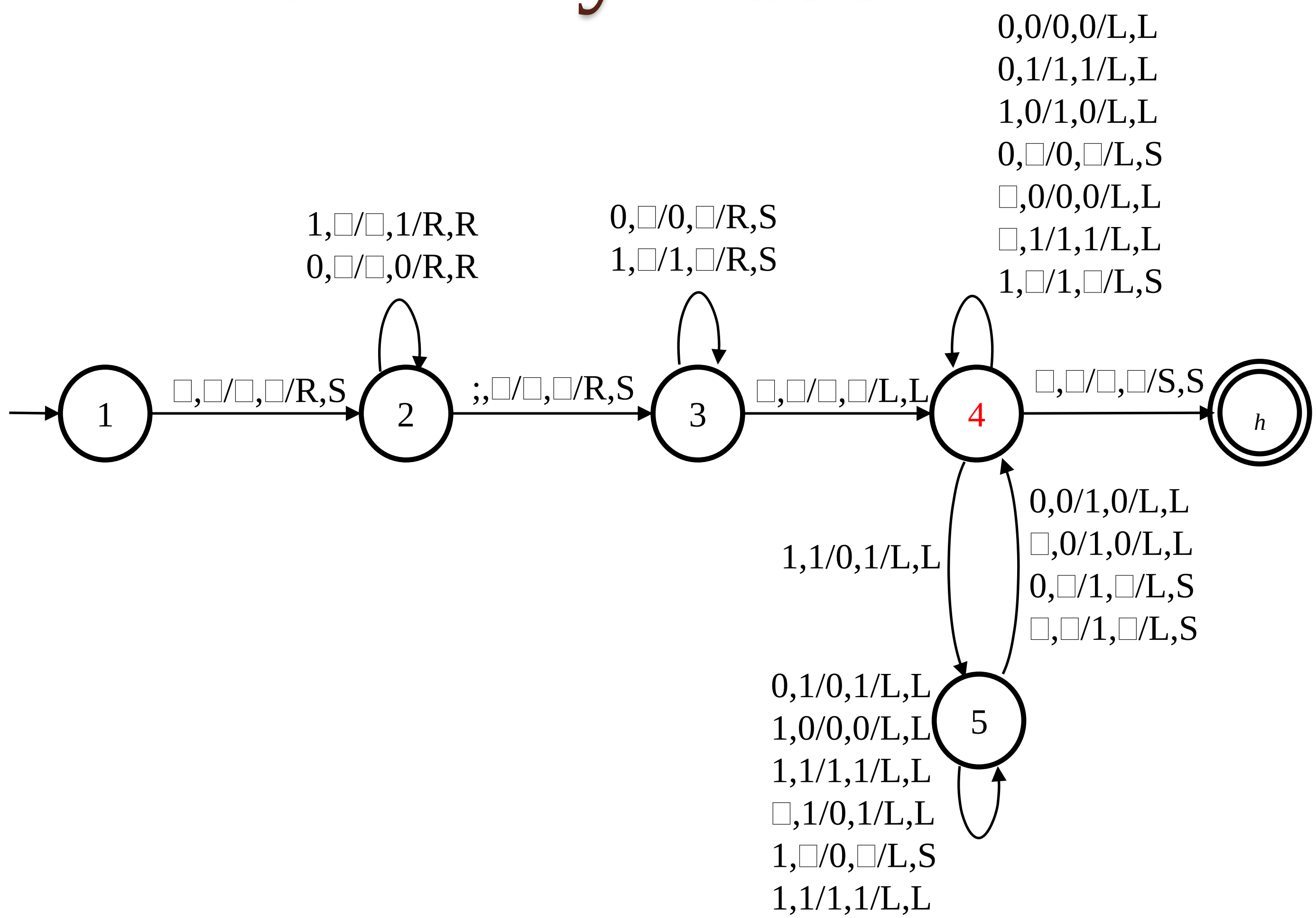
TM-2T: Duplicator



TM-2T: Binary Adder

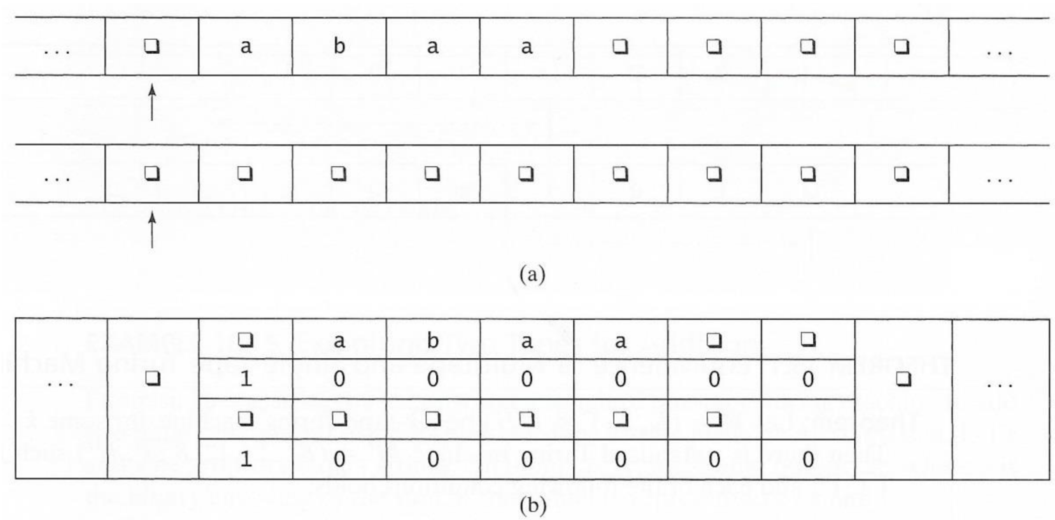


TM-2T: Binary Adder



Teorema: Ekivalensi TM Multitape & TM Single-tape

- Setiap TM Mutitape dapat disimulasikan oleh TM single-tape dengan ide:
  - Single tape dipandang sebagai multitrack tape dengan tambahan track-track untuk status posisi-posisi head masing-masing track.

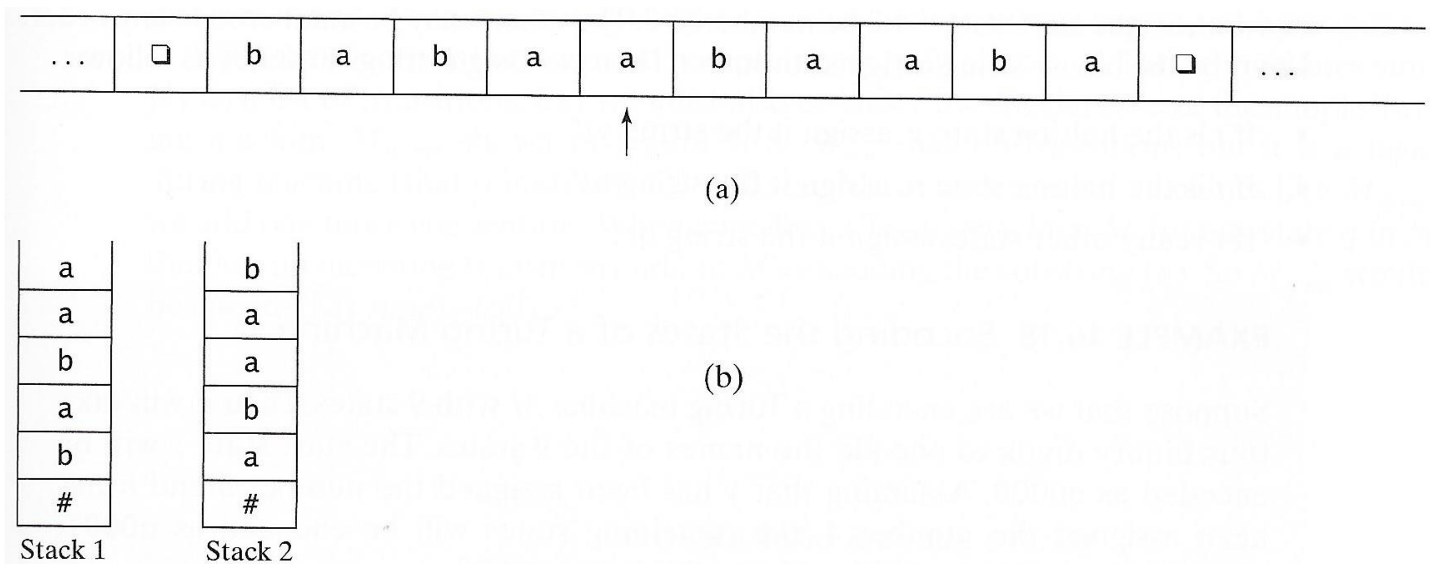


Mesin Turing One-way

- Mesin Turing yang telah dibahas adalah two-way.
- Mesin Turing one-way memiliki tape berujung berhingga di kiri.
- Bisa terjadi situasi crash (non-halting termination) akibat head “Keluar” dari tape (yaitu ketika di posisi terkiri, head bergerak ke kirinya).
- Teorema: Setiap TM two-way dapat disimulasikan oleh TM One-way

PDA Double Stack vs TM

- Setiap TM dapat disimulasikan oleh PDA double stack.
  - Perpindahan head ke kanan disimulasikan dengan pop elemen dari stack 2 lalu push ke stack 1.



Tag Systems (Post Machines)

- Adalah PDA dengan stack digantikan queue.
- Dipublikasikan Emil Post thn 1920-an.
- Bukan hanya mengenali bahasa WW tetapi ekivalen dengan Mesin Turing
- bekerjanya head dan tape mesin Turing dapat disimulasi Tag System dengan mekanisme circular queue pada Tag System.
  - Queue berisi *n* non-blank active tape + satu blank di kirinya atau di kanannya.
  - “head” berada pada blank
  - “operasi R” adalah dequeue dari front dan enqueue ke back
  - “operasi L” enqueue n-1 dari front dan enqueue ke back
  - Dst..

Lambda Calculus

- Saat Alan Turing mengembangkan model komputasi mesin turing (TM), Alonzo Church mengembangkan model komputasi Lambda Calculus (LC) [keduanya thn 1936] .
  - TM dasar dari imperative programming paradigm
  - LC dasar dari functional programming paradigm (Lisp, Scheme, ML, Haskell).
- LC: Bahasa ekspresi yang mendefinisikan fungsi dengan argumen tunggal (notasi λ mengawali deklarasi variable. Contoh:
  - $(\lambda x.x{+}1)$ → $f(x) = x{+}1$
  - $(\lambda x\lambda y.x{+}\mathrm{y})$ → $f(x,y) = x{+}\mathrm{y}$
- Dan, binding dengan parameter formal. Contoh:
  - $(\lambda x.x{+}1)\ 3$ → x=3, maka $f(3) = 3+1 = 4$
  - $(\lambda x.\lambda y.x{+}\mathrm{y})\ 3\ 4$ → x=3 dan y=4 maka $f(3,4) = 3+4 = 7$
- → LC memiliki power yang ekivalen dengan TM.

Nondeterministik Pada Mesin Turing

- Ingat bahwa:
  - Nondeterminisme dalam FSM hanya mempermudah perancangan mesin tetapi tidak menambah “power”.
  - Nondeterminisme dalam PDA menambah “power” (terdapat bahasa yang diterima oleh NDPDA tetapi tidak dapat diterima oleh DPDA)
- Untuk TM?
  - Tidak menambah “power” (terkait penerimaan bahasa)
  - Kompleksitas DTM vs NDTM terjadi peningkatan jumlah step secara eksponensial.
- Istilah NP dalam teori kompleksitas berasal dari waktu **P**olinomial untuk **N**ondeterministic TM (artinya waktu eksponensial untuk algoritma sikuensial).

Mesin Turing Universal (UTM)

- UTM dapat dibuat untuk memproses input string yang terdiri dari sepasang string:
    - String <*M*>: merupakan kasil pengkodean dari TM M
    - String <*w*>: merupakan hasil pengkodean dari input string *w*
- Proses yang dijalankan UTM adalah mensimulasikan bekerjanya <*M*> saat memproses input <*w*>.
    - Analogi: interpreter (PHP/Javascript/JVM/Perl) yang mengeksekusi script (berupa teks) dengan input data (berupa teks).
- Aturan pengkodean <*M*> dan <*w*> dapat dibuat misalnya pada slide berikutnya (spt pada textbook).

Contoh: Pengkodean $M \rightarrow \langle M \rangle$

- Diberikan $M = (K, \Sigma, \Gamma, \delta, s, H)$
- Status-status dalam K dikodekan secara biner (i) dalam $\lceil {}^2\log |K| \rceil$ digit,
  - misanya jika $|K| = 9$ maka $i = 4$, dikodekan menjadi q0000, q0001, q0010,…, q1000.
- Simbol-simbol dalam Γ dikodekan secara biner dalam $\lceil {}^2\log |\Gamma| \rceil$ digit,
  - Misalnya $\Gamma = \{ \square, a, b, c\}$, dikodekan menjadi $\square$=a00, a=a01, b=a10, c=a11
- Rule-rule dalam δ, setiap komponennya dikodekan dengan pengkodean status dan simbol di atas, serta satu simbol untuk arah (R atau L).

Contoh: Pengkodean K dan Γ

- **Encoding status-status** dalam *K* dalam format **xBB..B** dengan BB..B representasi biner dari nomor urut status dan jumlah digit sesuai dengan |*K*| sementara x berharga q (*non-halting state*) atau y (*accepting state*) atau n (*rejecting state*).
- **Start state** dibuat yang pertama (eg. q000 untuk 3 digit).
  - Contoh: |*K*| = 9, status-status adalah q000, q0001, q0010, y0011, n0100, q0101,…, q1000 dengan q000 start state, y0011 accepting state, n0100, rejecting state.
- **Encoding alfabet** Γ dalam format **aBB..B** dengan BB..B representasi biner dari nomor urut simbol dan jumlah digit sesuai dengan |Γ|.
  - Contoh: Γ = {□, a, b, c} , alfabet menjadi a00, a01, a10, a11

Contoh: Pengkodean Arah dan δ

- Arah head selalu hanya 2: "→" atau "←" maka dapat menggunakan satu digit biner (untuk readability bisa tetap dengan simbol arah tsb).
- δ dapat merupakan list dari 5-tuple terpisahkan koma (untuk readability): (state, input, state, output, arah)
- Contoh: (`q000`,`a000`,`q110`,`a000`,→)
- Secara implisit <*M*> cukup diwakili pencodean δ karena anggota *K* dan anggota Γ dapat disimpulkan dari δ.

Contoh Kasus

Consider $M = (\{s, q, h\}, \{a, b, c\}, \{❑, a, b, c\}, \delta, s, \{h\})$, where δ =

state	symbol	δ
s	❑	$(q, ❑, \rightarrow)$
s	a	$(s, \mathrm{b}, \rightarrow)$
s	b	$(q, \mathrm{a}, \leftarrow)$
s	c	$(q, \mathrm{b}, \leftarrow)$
q	❑	$(s, \mathrm{a}, \rightarrow)$
q	a	$(q, \mathrm{b}, \rightarrow)$
q	b	$(q, \mathrm{b}, \leftarrow)$
q	c	$(h, \mathrm{a}, \leftarrow)$

$i = \lceil {}^2\log |K| \rceil = 2$

state/symbol	representation
s	q00
q	q01
h	q10
❑	a00
a	a01
b	a10
c	a11

<*M*>=
(q00,a00,q01,a00,→),(q00,a01,q00,a10,→),
(q00,a10,q01,a01,←),(q00,a11,q01,a10,←),
(q01,a00,q00,a01,→),(q01,a01,q01,a10,→),
(q01,a10,q01,a10,←),(q01,a11,q10,a01,←)

Pengkodean Input <w>

- Jika $w = x_1x_2x_3\ldots x_n$ maka $<w>$ diperoleh sebagai string tersusun atas pengkodean masing-masing x_i mengikuti pengkodean simbol dalam Γ.
- Untuk UTM maka $<M>$ dan $<w>$ ditulis sebagai sebuah string $<M, w>$ (dua string terpisahkan satu separator).