MODUL 15: MESIN TURING

Model Komputasi

Mesin-mesin abstraks yang telah kita pelajari merupakan model algoritma-algoritma (model komputasi) untuk kelas-kelas masalah tertentu dengan string masukan masukan dapat memberikan jawaban "ya" atau "tidak".

- Mesin-mesin abstraks FA: algoritma yang hanya dapat menyimpan informasi dalam jumlah yang amat terbatas
- Mesin-mesin abstraks PDA: algoritma yang dapat menyimpan informasi dalam aturan LIFO dan selama algoritma dijalankan

Dari contoh-contoh yang dibahas terlihat bahwa model komputasi PDA atau apalagi FA tidak mampu mengenali bahasa-bahasa tertentu, contohnya $\{a^n b^n c^n\}$. Untuk bahasa semacam itu diperlukan bahasa yang memiliki kemampuan lebih tinggi dari PDA. Jika kita boleh merancang sendiri maka untuk bahasa tersebut kita gunakan dua stack: yang pertama untuk mencatat berapa banyak *a* muncul sebelum *b* dan yang kedua untuk mencatat berapa banyak *b* muncul sebelum *c*. Untuk bahasa $\{ss \mid s \in \{a, b\}^*\}$ kita berharap dapat menggunakan struktur data queue bukannya stack.

Human Computer

Alan Turing 60 tahun yang lalu membayangkan suatu "human computer" yang melakukan komputasi dengan:

- memiliki pensil dan kertas
- mengikuti algoritma tertentu dalam pemecahan masalah.

Algoritma berisi langkah-langkah primitif. Dalam modelnya Alan Turin melihat langkah-langkah primitif tersebut sebagai

- pengamatan terhadap suatu simbol tunggal pada kertas,
- menghapusnya dan
- menggantikannya dengan simbol lain.

Pada suatu saat aksi dari mesin ditentukan oleh simbol yang sedang diamati dan "state of mind" dari "human computer" tersebut. State of mind dapat berubah sejalan dengan simbol yang diamati dan komputasi yang dilakukan, namun dalam asumsi bahwa jumlah status adalah berhingga (artinya state of mind tidak dapat menyimpan seluruh langkah yang telah dilakukannya akibat keterbatasan jumlah status ini).

Jadi human computer ini yang kita seterusnya menyebutnya dengan nama Mesin Turing (TM) merupakan model komputasi yang lebih umum.dengan kemampuan mengeksekusi suatu program yang tersimpan pada string masukan.

Model Mesin Turing

Akan dibahas model dasar TM serta sejumlah formulasi dengan peningkatan efisiensi dan convenience dari model dasar tersebut untuk memperluas himpunan bahasa yang dapat dikenalinya. Dalam model dasarnya TM memiliki:

- Himpunan **status** yang berhingga
- **Tape linear** yang berfungsi sebagai media yang membawa string masukan dan sekaligus untuk penyimpanan
- **Tape head** yang membaca-tulis simbol-simbol dalam tape

Tape memiliki struktur sbb

- berujung di sebelah kiri tapi tak berujung (secara potensial) di sebelah kanan
- terbagi atas kotak-kotak yang masing-masing untuk dapat menyimpan satu simbol.

Jika pada suatu kotak tape tidak tersimpan simbol maka kita katakan kotak berisi **simbol blank**. Namun, pada mulanya tape hanya berisi masukan dengan simbol-simbol alfabet masukan tanpa adanya simbol-simbol blank. Selama berlangsungnya komputasi tape dapat berisikan juga selain alfabet masukan ditambah juga simbol-simbol lain yang bukan blank. Kita sebut himpunan ini sebagai alfabet tape.

Mesin akan melakukan suatu "move" yang bisa berisi tiga hal:

- Mengubah simbol dalam kotak yang sedang diamati
- Memindahkan head satu kotak ke sebelah kiri (kecuali kalau sudah berada di paling kiri) atau kanannya atau tetap pada posisinya
- Mengubah statusnya

Dalam TM terdapat perbedaan definisi antara "menerima suatu bahasa" dengan "mengenali suatu bahasa". Keduanya dibedakan karena adanya kemungkinan TM berada dalam keadaan beriterasi tanpa akhir (loop forever) untuk sejumlah masukan sehingga tidak mengembalikan suatu hasil. Kalau suatu TM untuk semua string akan selalu memberikan keputusan (tidak akan terjadi loop forever) maka suatu TM tersebut dapat mengenali atau tidak suatu bahasa L.

Dibandingkan dengan human computer yang sebenarnya tentu TM memiliki banyak penyederhanaan, misalnya kertas dalam kenyataannya adalah media nondiskrit dan dua dimensi (atau bahkan tiga) sementara move dilakukan secara linear dalam satu dimensi. Demikian pula, secara satu kali melihat (at a glance) maka manusia bisa mendapatkan posisi dimana ia akan membaca yang berikutnya.

Sekalipun adanya perbedaan ini Alonzo Church mengemukakan thesis yang menyatakan bahwa adalah bahwa suatu prosedur algoritmis yang dapat dikerjakan oleh seorang manusia, satu team manusia, atau suatu kompuer, dapat dikerjakan pula oleh TM. Thesis ini kemudian dikenal sebagai Church-Turing Thesis.

Definisi Mesin Turing

Definisi. Suatu Mesin Turing (TM) adalah 5-tuple $T = (Q, \Sigma, \Gamma, q_0, \delta)$ dimana Q adalah himpunan berhingga dari status, tidak berisi status halt h (simbol yang sama yang akan digunakan sebagai status halt untuk setiap TM)

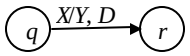
- Σ alfabet masukan
- Γ alfabet tape dengan $\Sigma \subseteq \Gamma$, dan Γ tidak termasuk Δ (simbol blank)
- q_0 status inisial. $q_0 \in Q$
- δ: $Q \times (\Gamma \cup \{\Delta\}) \rightarrow (Q \cup \{h\}) \times (\Gamma \cup \{\Delta\}) \times \{R, L, S\}$

TM dimodelkan memiliki media penyimpanan tape yang berujung di kiri tapi tak berujung di kanan, serta diformat dalam kotak-kotak (blok-blok) berindeks dari 0 yang terkiri, 1, 2, dst. kekanan. Masing-masing kotak dapat menyimpan suatu simbol $\in \Gamma \cup \{\Delta\}$. Pada mula TM bekerja simbol-simbol yang bukan blank pada tape hanya simbol $\in \Sigma$.

Untuk membaca dan menulis pada kotak-kotak tape, mesin memiliki tape head. Jika $q \in Q$, dan $r \in Q \cup \{h\}$, serta $X, Y \in \Gamma \cup \{\Delta\}$, dan $D \in \{R, L, S\}$, maka formula "$\delta(q, X) = (r, Y, D)$" memiliki arti: bila TM berada pada status q dan simbol pada kotak tape dimana head berada adalah X, maka mesin akan menggantikan X dengan Y, dan berubah status menjadi r. Posisi head kemudian bergantung harga D, jika D = L maka head bergeser satu kotak ke kiri, jika D = R maka head bergeser satu kotak ke kanan, atau jika D = S maka head tetap pada posisi sebelumnya. Jika pada saat head berada pada kotak 0 dan head digeserkan ke kiri maka terjadi situasi **crash**. Jika status head berubah menyadi h maka mesin akan **halt** (mesin tidak akan bererak lagi karena δ(h,X) tidak terdefinisi).

Mesin Turing Deterministik

Pada bentuk TM yang dasar ini, mesin didefinisikan deterministik (nondeterministik TM akan dibahas kemudian) namun $\delta(q, X)$ dimungkinkan tak terdefinisi. Formula $\delta(q, X) = (r, Y, D)$ digambarkan dalam diagram sbb.



Setiap saat TM dideskripsikan selain oleh statusnya sendiri, juga oleh isi dari tapenya serta posisi head saat itu. Untuk hal itu kita katakan bahwa suatu **konfigurasi** dari TM adalah pasangan:

$(q, x\underline{a}y)$

dengan q adalah status, x dan y adalah string dari $\Gamma \cup \{\Delta\}$ serta a adalah simbol $\in \Gamma \cup \{\Delta\}$. Sementara garis bawah pada a menyatakan bahwa posisi head berada a.

Jika x bukan string kosong maka x adalah string dengan simbol terkiri pada kotak 0 tape, kemudian a dan diikuti oleh string y. Jika x string kosong maka a berada pada kotak 0 tape. Notasi konfigurasi kadang-kadang juga ditulis sebagai

- $(q, x\underline{w})$ yang menyatakan bahwa tape head berada pada simbol terkiri dari string w
- $(q, x\underline{a}y\Delta)$ yang menyatakan secara eksplisit bahwa setelah yang menyatakan secara eksplisit bahwa setelah y tidak adalagi simbol lain kecuali blank.

Translasi Konfigurasi

TM T dapat berpindah dari satu konfigurasi ke konfigurasi yang lain dengan penulisan sbb.

$(q, x\underline{a}y) \vdash_T (r, z\underline{b}w)$

T berpindah dari konfigurasi kiri ke konfigurasi kanan dalam satu kali move.

$(q, x\underline{a}y) \vdash^*_T (r, z\underline{b}w)$

T berpindah dari konfigurasi kiri ke konfigurasi kanan dalam 0 kali atau beberapa kali move.

Contoh: jika T berada pada konfigurasi $(q, aab\underline{a}\Delta a)$ dan terdapat $\delta(q, a) = (r, \Delta, L)$ maka konfigurasi berikutnya setelah satu move adalah

$(q, aaba\Delta a) \vdash_T (r, aab\Delta\Delta a)$

T berpindah dari konfigurasi kiri ke konfigurasi kanan dalam satu kali move.

Jika tidak membingungkan (mesin TM yang digunakan sudah jelas) maka notasi $\vdash_T$ dan $\vdash^*_T$ masing-masing dapat dituliskan cukup dengan $\vdash$ dan $\vdash^*$.

Beberapa Mesin Turing pada Tape yang Sama

Untuk memungkinan beberapa TM secara berurutan dijalankan maka suatu TM start bisa dimana saja pada tape (tidak harus mulai dari kotak tertentu tape) namun suatu TM harus berjalan secara terisolasi (tidak overlap dengan working space TM lain) dan sebagai konvensi ia akan start dari simbol blank tepat di sebelah kiri simbol pertama *x*. Jadi konfigurasi inisial adalah $(q_0, \underline{\Delta}x)$.

Definisi 9.2. $T = (Q, \Sigma, \Gamma, q_0, \delta)$ merupakan suatu TM, dan $x \in \Sigma^*$, maka *x* diterima oleh *T* bila mulai dari konfigurasi awal berkaitan dengan masukan *x*, *T* pada akhirnya mencapai konfigurasi **halt**. Dengan kata lain, *x* diterima bila terdapat $y, z \in (\Gamma \cup \{\Delta\})^*$ dan $a \in \Gamma \cup \{\Delta\}$

$(q_0, \underline{\Delta}x) \vdash^*_T (h, y\underline{a}z)$

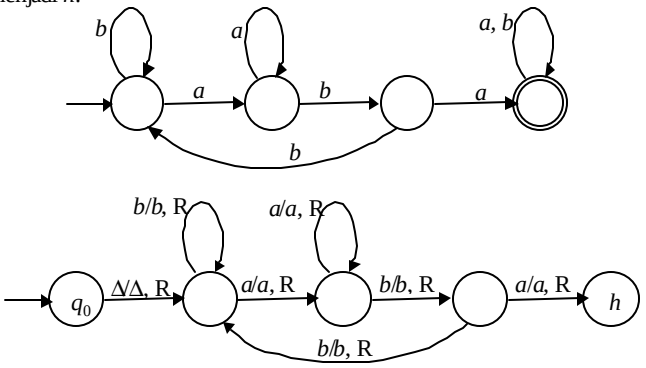
Dalam situasi ini kita katakan bahwa *T* halt pada masukan *x*. Bahasa yang diterima *T* adalah *L*(*T*) sebagai himpunan string-string yang menyebabkan *T* halt.

Ketiga kasus yang akan terjadi jika suatu string gagal diterima:

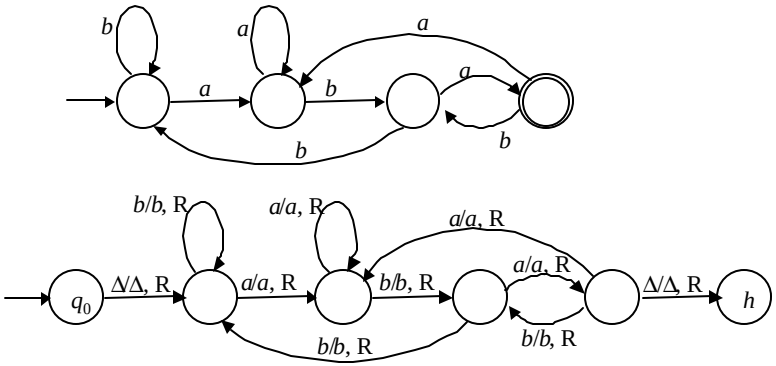
- **Berhenti di konfigurasi bukan halt** dan TM tidak dapat move lagi misalkan pada status *q* TM berada di atas simbol *X* dan $\delta(q, X)$ tidak terdefinisi; dalam hal ini dikatakan ditolak karena terjadi crash.
- TM bergerak ke sebelah kiri dari kotak 0 tape; juga ditolak karena **crash**.
- Suatu string masukan tertentu menyebabkan TM berada dalam situasi **infinite loop** yaitu dapat terus bergerak tapi tidak pernah dapat mencapai status halt (terjadi perulangan konfigurasi); dalam hal ini TM tidak dapat memutuskan menerima atau tidak.

Adanya infinite loop ini dalam TM berimplikasi penting terhadap teori komputasi yang nanti akan dibahas. Dalam contoh-contoh berikut ini kasus tersebut akan dihindari (setiap string masukan akan diterima atau ditolak).

Contoh 9.1: $L = \{x \in \{a, b\}^* \mid x \text{ berisi substring } aba\}$. Ini sekedar contoh suatu TM dari bahasa yang sebenarnya merupakan bahasa regular sehingga akan ada FA yang mengenalinya. Meniru FA tersebut suatu TM dapat dibuat. Setiap move akan bergerak ke kanan dan isi kotak tidak berubah. Ditambahkan satu status agar TM dapat start dari kotak sebelum simbol pertama string *x*. Dalam kasus ini karena jika FA sudah berada dalam status menerima ia akan tetap disitu pada setiap simbol apapun berikutnya, maka status menerima tersebut dapat langsung diubah menjadi *h*.



Dalam hal ini segera setelah TM menemukan substring *aba* maka ia akan halt tanpa memeriksa seluruh string masukan. Hal ini tidak berlaku umum karena ada bahasa yang memerlukan TM untuk memeriksa seluruh string masukan, contohnya untuk $L_1 = \{x \in \{a, b\}^* \mid x \text{ berakhiran } aba\}$. Dalam kasus ini TM harus memeriksa seluruh string dan juga *h* tidak bisa langsung menggunakan status menerima FA.



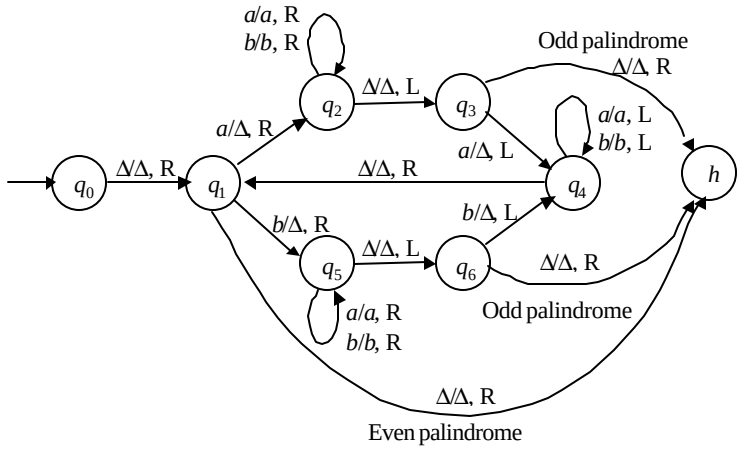
Contoh-contoh di atas masih terlalu sederhana sehingga tidak menggambarkan kemampuan TM yang sebenarnya. Contoh berikutnya adalah TM untuk bahasa palindrom.

Contoh 9.2. Bahasa palindrom dapat dikenali oleh suatu PDA. Jika kita diijinkan untuk menerapkan nondeterminisme dalam TM maka suatu TM nondeterministik yang dapat mengenai palindrom dapat langsung dibuat mengan memodifikasssi PDA yang mengenali palindrom. Tapi karena kita masih membahas TM yang deterministik maka kita merancangnya dari nol sbb. Idenya adalah memeriksa ujung-ujungnya, yang jika sama maka simbol-simbol di kedua ujung dihapuskan dan pemeriksaan diulangi. Misalnya string masukan adalah *x*.

- Jika simbol nonblank terkiri adalah *a* maka simbol tsb diganti dengan blank dan mesin bergerak ke kanan dan berhenti pada kotak nonblank terkanan;
- jika kotak nonblank terkanan tidak berisi *a* maka halt, jika ya maka *a* di kotak terkanan juga diganti blank, lalu mesin bergerak ke kiri untuk mendapatkan kotak nonblank terkiri

dalam menjalankan langkah-langkah itu terjadi situasi

- setelah langkah pertama , semua kotak langsung blank maka x adalah palindrom yang panjangnya gasal (odd palindrome)
- setelah langkah kedua, semua kotak langsung blank maka x adalah palindrom yang panjangnya genap (even palindrome)



Untuk string masukan *abaa*:

(q_0, $\underline{\Delta}abaa$) (q_1, $\Delta\underline{a}baa$) (q_2, $\Delta\Delta\underline{b}aa$) ($q_2$, $\Delta\Delta b\underline{a}a$) ($q_2$, $\Delta\Delta ba\underline{a}$)
(q_2, $\Delta\Delta baa\underline{\Delta}$) ($q_3$, $\Delta\Delta ba\underline{a}$) ($q_4$, $\Delta\Delta b\underline{a}\Delta$) ($q_4$, $\Delta\Delta\underline{b}a\Delta$)
(q_4, $\Delta\underline{\Delta}ba\Delta$) ($q_1$, $\Delta\Delta\underline{b}a\Delta$) ($q_5$, $\Delta\Delta\Delta\underline{a}\Delta$) ($q_5$, $\Delta\Delta\Delta a\underline{\Delta}$)
(q_6, $\Delta\Delta\Delta\underline{a}\Delta$) (crash)

Untuk string masukan *aba*:

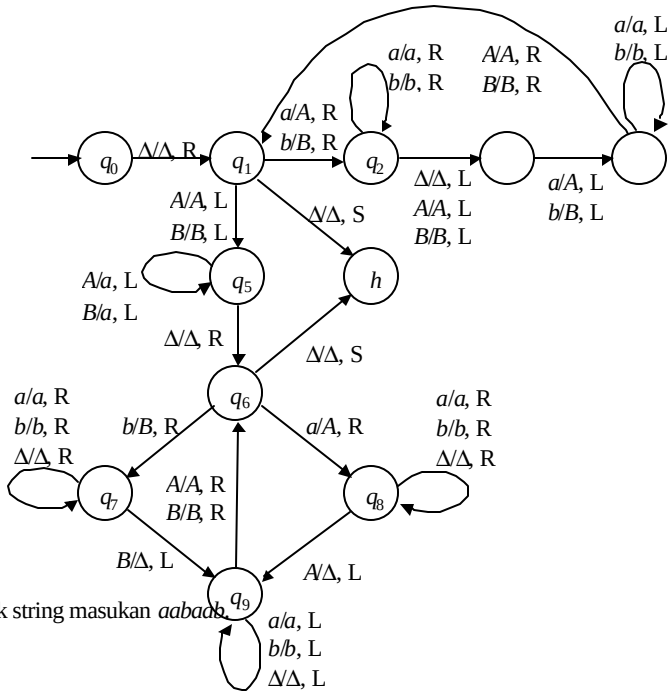
(q_0, $\underline{\Delta}aba$) (q_1, $\Delta\underline{a}ba$) (q_2, $\Delta\Delta\underline{b}a$) ($q_2$, $\Delta\Delta b\underline{a}$) ($q_2$, $\Delta\Delta ba\underline{\Delta}$) ($q_3$, $\Delta\Delta b\underline{a}$)

(q_4, ΔΔ$\underline{b}$Δ) (q_4, Δ$\underline{Δ}$$b$Δ) ($q_1$, ΔΔ$\underline{b}$Δ) ($q_5$, ΔΔΔ$\underline{Δ}$) ($q_6$, ΔΔ$\underline{Δ}$Δ)
(h, ΔΔΔ$\underline{Δ}$) (accepted)

Untuk string masukan *abba*:

(q_0, $\underline{Δ}$*abba*) (q_1, Δ$\underline{a}$*bba*) (q_2, ΔΔ$\underline{b}$*ba*) (q_2, ΔΔ$b\underline{b}a$) (q_2, ΔΔ$bb\underline{a}$)
(q_2, ΔΔ$bba\underline{Δ}$) (q_3, ΔΔ$bb\underline{a}$) (q_4, ΔΔ$b\underline{b}$Δ) (q_4, ΔΔ$\underline{b}b$Δ)
(q_4, ΔΔ$\underline{Δ}bb$Δ) (q_1, ΔΔ$\underline{b}b$Δ) (q_5, ΔΔΔ$\underline{b}$Δ) (q_5, ΔΔΔ$\underline{b}$Δ)
(q_6, ΔΔΔ$\underline{b}$Δ) (q_4, ΔΔ$\underline{Δ}$ΔΔ) (q_1, ΔΔ$\underline{Δ}$ΔΔ) (h, ΔΔΔ$\underline{Δ}$) (accepted)

Contoh 9.3. $L = \{ss \mid s \in \{a, b\}^*\}$. Bahasa ini sudah diketahui bukan suatu CFL. Suatu TM dirancang untuk menemukan dan menandakan pertengahan string kemudian membandingkan kedua bagian tsb.



Untuk string masukan *aabaab*.

(q_0, $\underline{Δ}$*aabaab*Δ) (q_1, Δ$\underline{a}$*abaab*Δ) (q_2, Δ*A*$\underline{b}$*baab*Δ) * (q_2, Δ*Aaabaab*$\underline{Δ}$)
(q_3, Δ*Aabaa*$\underline{b}$Δ) (q_4, Δ*Abaa*$\underline{B}$Δ) * (q_4, Δ$\underline{A}$*abaa*BΔ)
(q_1, Δ*A*$\underline{a}$*baa*BΔ) (q_2, Δ*AA*$\underline{b}$*aa*BΔ) * (q_2, Δ*AAbaa*$\underline{B}$Δ)
(q_3, Δ*AAbaa*$\underline{B}$Δ) (q_4, Δ*AAb*$\underline{a}$*AB*Δ) * (q_4, Δ*A*$\underline{A}$*baAB*Δ)
(q_1, Δ*AA*$\underline{b}$*aAB*Δ) (q_2, Δ*AAB*$\underline{a}$*AB*Δ) (q_2, Δ*AABa*$\underline{A}$*B*Δ)
(q_3, Δ*AABaAB*Δ) (q_4, Δ*AABAAB*Δ) (q_1, Δ*AABAAB*Δ)
posisi tengah sudah ditemukan, s kiri dikembalikan
(q_5, Δ*AA*$\underline{B}$*AAB*Δ) * (q_5, $\underline{Δ}$*aabAAB*Δ)
pemeriksaan simbol pertama s kiri dan s kanan
(q_6, Δ$\underline{a}$*abAAB*Δ) (q_8, Δ*A*$\underline{a}$*bAAB*Δ) * (q_8, Δ*Aab*$\underline{A}$*AB*Δ)
(q_9, Δ*Aa*$\underline{b}$ΔΔ*AB*Δ) * (q_9, Δ$\underline{A}$*ab*ΔΔ*AB*Δ)
pemeriksaan simbol kedua s kiri dan s kanan
(q_6, Δ*Aab*ΔΔ*AB*Δ) (q_8, Δ*AAb*ΔΔ*AAB*Δ) * (q_8, Δ*AAb*ΔΔ*AB*Δ)
(q_9, Δ*AAb*$\underline{Δ}$ΔΔ*B*Δ) * (q_9, Δ*A*$\underline{A}$*b*ΔΔ*B*Δ)
pemeriksaan simbol ketiga s kiri dan s kanan
(q_6, Δ*AA*$\underline{b}$ΔΔΔ*AB*Δ) (q_7, Δ*AAB*$\underline{Δ}$Δ*B*Δ) * (q_7, Δ*AAB*ΔΔ$\underline{B}$Δ)
(q_9, Δ*AAB*ΔΔΔΔ) * (q_9, Δ*AA*$\underline{B}$ΔΔΔΔ)
(q_6, Δ*AAB*$\underline{Δ}$ΔΔΔ)
selesai
(h, Δ*AAB*$\underline{Δ}$ΔΔΔ)

Kombinasi Mesin Turing

Suatu TM dapat dispesifikasikan sebagai komposit dari sejumlah TM yang sederhana. Jika diberikan T_1 dan T_2 adalah masing-masing TM yang memiliki fungsi transisi d_1 dan d_2 dan himpunan status nonhalting yang disjoin. T_1T_2 adalah suatu TM dengan status nonhalting merupakan gabungan dari himpunan status masing-masing mesin itu serta status inisialnya adalah status inisial T_1.

- Dari status inisial itu, TM T_1T_2 akan melakukan move sesuai jika T_1 melakukan move hingga T_1 halt atau crash.
- Jika T_1 halt maka T_1T_2 tidak halt tapi move ke status inisial T_2 dan dari situ berlanjut dengan melakukan move sesuai dengan apa T_2 akan lakukan.
- Jika T_1 atau T_2 crash untuk suatu masukan maka T_1T_2 juga akan crash.

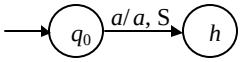
Dalam pengambarannya tanpa kita menggambarkan masing-masing dalam diagram transisinya secara eksplisit, T_1T_2 dapat digambarkan sebagai

$$T_1 \rightarrow T_2$$

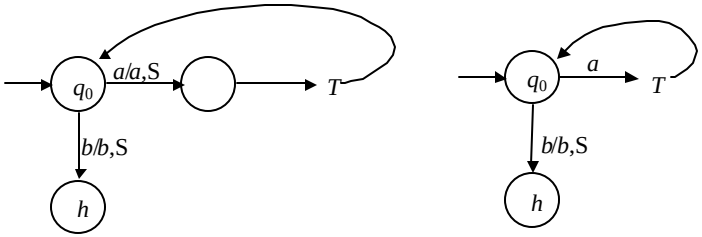
Bila diharapkan bahwa komposit itu bersifat kondisional yaitu bergantung pada simbol tape dimana head T_1 berada saat halt, misalnya a, maka notasi itu ditulis sebagai

$$T_1 \xrightarrow{a} T_2$$

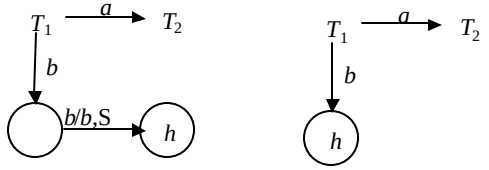
Yang berarti suatu $T_1T'T_2$ dimana T' adalah suatu TM yang digambarkan sbb.



Untuk menyederhanakan penggambarannya suatu mesin Turing komposit boleh digambarkan secara campuran dengan mengganti bagian-bagian yang sudah jelas dengan notasi seperti di atas. Contohnya pada gambar-gambar berikut ini yang merupakan versi campuran dari suatu komposit. Selain itu gambar di kanan merupakan versi yang lebih ringkas dari yang dikiri.



Namun, penggambaran dengan notasi tersebut bisa menyebabkan kerancuan. Misalkan suatu TM dalam bentuk $T = T_1 \xrightarrow{a} T_2$
Jika T_1 halt dalam scanning simbol selain a yang tidak dispesifikasikan dengan eksplisit maka T crash. Namun bila T_2 halt maka T halt sekali pun tidak ada simbol tape yang dispesifikasikan secara eksplisit. Inkonsistensi ini bisa dicegah dengan menyatakan bahwa "Jika T_1 halt untuk simbol selain a, maka T juga halt" namun T sudah tidak ekivalen lagi dengan $T_1T'T_2$ di atas. Sebagai gantinya maka situasi itu digambarkan secara eksplisit dengan adanya pilihan halting itu pada diagram (gambar kanan bentuknya yang lebih ringkas).

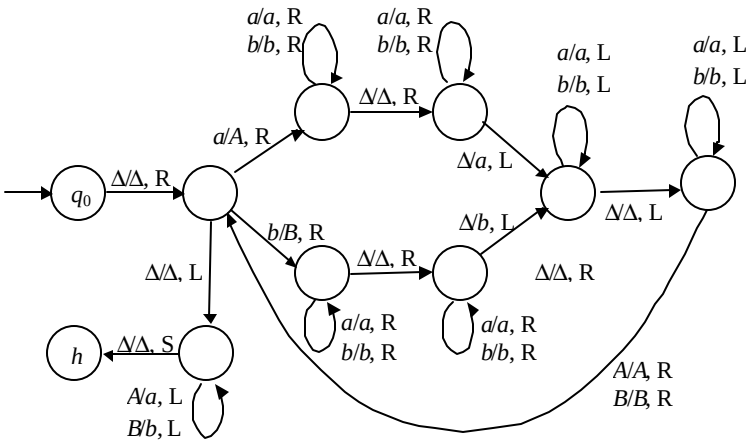


Sejumlah TM yang hanya akan crash ketika dipandang sebagai mesin-mesin *self-contained* (contohnya TM yang mengeksekusi algoritma "move head satu kotak ke kiri"). Jika masing-masing mesin akan halt jika dijalankan secara terpisah, maka sebagai komponen dari mesin komposit juga tidak akan crash. Misalnya suatu TM T mengharapkan menemukan suatu string input x saat memulai dengan konfigurasi $(q, y\underline{\Delta}x)$. Move-move selanjutnya tidak akan bergantung pada y serta tidak akan melakukan move ke sebelah kiri dari blank itu (tidak akan mendapatkan simbol dari y).

Berikut ini sejumlah TM yang melakukan operasi-operasi dasar pada tape dan seringkali berguna dalam menyusun struktur TM yang lebih kompleks yang melakukan operasi pada tape.

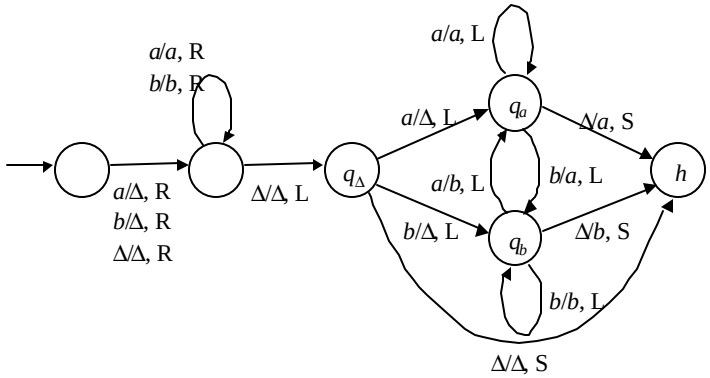
Contoh 9.4 (TM Copy). Suatu TM yang dapat melakukan penyalinan string masukan ke bagian setelah kanannya yang terpisahkan oleh satu kotak blank. Jadi dengan masukan string $x \in \{a, b\}^*$ menghasilkan transisi konfigurasi $(q_0, \Delta x\Delta)$

*(*h*, $\underline{\Delta}$xΔxΔ). TM membaca simbol pertama, lalu menyalinnya, membaca simbol kedua, menyalinnya, dan seterusnya. Untuk menandakan simbol yang telah disalin, simbol itu diubah ke huruf kapital. Saat penyalinan selesai, maka simbol masukan kembali ke huruf kecil.



Contoh 9.5 (TM Delete): menghapus suatu simbol dalam string masukan. Seringkali kita memerlukan operasi penghapusan suatu simbol dari suatu string.TM melakukan isi tape dari $y\underline{a}z$ menjadi $y\underline{z}$ dimana $y \in (\Sigma \cup \{\Delta\})^*$, $a \in (\Sigma \cup \{\Delta\})$, dan $z \in \Sigma^*$.

Berikut ini TM melakukannya untuk $\Sigma = \{a, b\}$. Simbol *a* tsb di hapus dengan menggantinya dengan blank sambil bergerak ke kanan. Kemudian TM bergerak ke ujung kanan string. Selanjutnya dalam satu kali pass dari kanan ke kiri, TM mencatat simbol yang sedang dibaca head & menggantinya dengan simbol yang sebelumnya. Diagram TM ini memiliki status q_a dan q_b untuk mencatat simbol yang akan dipindahkan, dan berhenti jika simbol yang dibaca adalah blank.



Penyisipan suatu simbol *a* (atau mengubah isi tape dari $y\underline{z}$ ke $y\underline{a}z$) akan dilakukan dengan cara yang mirip, kecuali pass dilakukan dari kiri ke kanan serta pada move pertamanya melakukan penulisan *a* (bukannya Δ).

Mengkomputasi Fungsi Parsial dengan Suatu TM

Suatu TM dengan alfabet masukan Σ dapat menjalankan suatu fungsi dengan domain adalah subset dari Σ^* dan kodomain adalah subset lain dari Σ^*. TM yang memiliki sifat tersebut bisa disebut sebagai TM dengan fungsi parsial karena bisa terdapat sejumlah titik dimana fungsi tidak terdefinisi. Jika terdefinisi di seluruh Σ^* maka disebut fungsi total. Selain itu suatu TM dapat menangani fungsi dengan sejumlah variabel. Dalam tape maka variabel-variabel ini adalah string-string yang dipisahkan oleh satu simbol blank.

Definisi 9.3. Jika $T = (Q, \Sigma, \Gamma, q_0, \delta)$ suatu TM, dan *f* adalah fungsi parsial pada $\Sigma^* \rightarrow \Gamma^*$ maka dikatakan bahwa *T* mengkomputasi *f* untuk setiap $x \in \Sigma^*$ dimana *f* terdefinisi untuk *x* tersebut,

$$(q_0, \underline{\Delta}x) \;\vdash_T^* (h, \underline{\Delta}f(x))$$

sementara untuk *x* lain, *T* akan gagal untuk halt untuk masukan *x* tersebut.

Jika f adalah fungsi parsial pada $(\Sigma^*)^k \to \Gamma^*$, T mengkomputasi f untuk setiap $(x_1, x_2, \ldots, x_k)$ yang mana f terdefinisi,

$$(q_0, \underline{\Delta} x_1 \Delta x_2 \ldots \Delta x_k) \quad {}^*_T (h, \underline{\Delta} f(x_1, x_2, \ldots, x_k))$$

sementara untuk masukan lain, T akan gagal untuk halt. Untuk dua alfabet Σ_1 dan Σ_2, dan untuk bilangan bulat $k \geq 0$ fungsi parsial $(\Sigma_1^*)^k \to \Sigma_2^*$, adalah *Turing-computable*, atau disingkat komputabel, jika terdapat suatu TM yang mengkomputasi f.

Bagaimana dengan masukan bilangan? Misalnya, bilangan natural (bilangan bulat nonnegatif) n dapat direpresentasikan oleh $111\ldots1 = 1^n$.

Definisi 9.4. Jika $T = (Q, \{1\}, \Gamma, q_0, d)$ suatu TM, dan f fungsi parsial $\mathbb{N} \to \mathbb{N}$, T mengkomputasi f untuk n yang mana f terdefinisi,

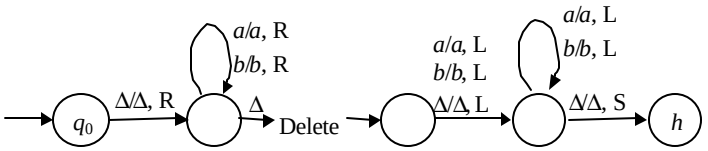
$$(q_0, \underline{\Delta} 1^n) \quad {}^*_T (h, \underline{\Delta} 1^{f(n)})$$

dan untuk bilangan natural lain maka T akan gagal untuk halt. Demikian halnya, jika f adalah fungsi parsial $\mathbb{N}^k \to \mathbb{N}$, T mengkomputasi f untuk $(n_1, n_2, \ldots, n_k)$ yang man f terdefinisi,

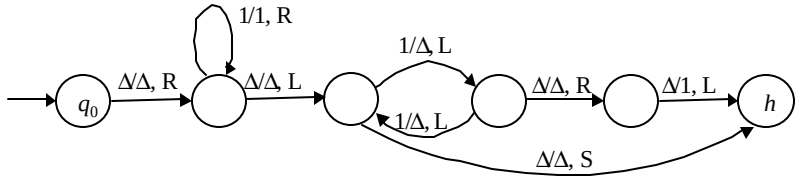
$$(q_0, \underline{\Delta} n_1 \Delta n_2 \ldots \Delta n_k) \quad {}^*_T (h, \underline{\Delta} f(n_1, n_2, \ldots, n_k))$$

dan untuk masukan lain maka T akan gagal untuk halt.

Contoh (TM Konkatenasi): Fungsi konkatenasi *cat*: $\Sigma^* \times \Sigma^* \to \Sigma^*$. TM yang mengkomputasinya untuk alfabet $\{a, b\}$ dapat dibuat dengan mudah dari TM Delete. Karena bilangan natural direpresentasikan oleh $\{1\}^n$ dan operasi penambahan dua bilangan dalam representasi ini sama dengan konkatenasi kedua string ybs maka TM ini jika digunakan untuk alfabet {1} akan menjadi TM Penambahan untuk dua bilangan natural.



Contoh (TM modulo 2). Fungsi numerik ini yang menghitung sisa pembagian n oleh 2 dapat dilakukan dengan melakukan pass dari kanan ke kiri dan menghapus setiap dua simbol 1. Diakhir pass akan tersisa satu simbol 1 jika n ganjil dan tidak ada yang tersisa jika n genap.



Contoh (Fungsi Karakteristik dari suatu himpunan). Untuk suatu bahasa $L \subseteq \Sigma^*$, fungsi karakteristik dari L adalah fungsi

$$c_L = \begin{cases} 1 & \text{jika } x \in L \\ 0 & \text{yang lain} \end{cases}$$

Mengkomputasi fungsi χ_L mirip dengan menerima L. Perbedaannya adalah mesin akan selalu halt. Setiap string yang menyebabkan halt pada mesin semula akan membuat TM ini halt pada konfigurasi $(h, \underline{\Delta}1)$, dan string lainnya (yang menyebabkan crash atau loop forever) akan membuat TM ini halt pada konfigurasi $(h, \underline{\Delta}0)$. Misalnya untuk TM Palindrom sebagai berikut.