CENG 306 Biçimsel Diller ve Otomatlar
Formal Languages and Automata

TURING MACHINE
(III)

Turing Machines

Tanım: *Karmaşık Turing makineleri basit makinelerin birleşimi şeklinde oluşturulabilir.*

Basit Makineler:

Symbol-writing machines (sembol yazma makineleri):

Her $a \in \sum \cup \{\leftarrow,\rightarrow\} - \{\Delta\}$ *için* $M_a = (\{s, h\}, \sum, \delta, s, \{h\})$ *tanımlanabilir ve*

$\delta(s, b) = (h, a)$ *olur, burada* $b \in \sum \cup \{\Delta\}$.

Burada da $\delta(s, \Delta) = (s, \rightarrow)$ *otomatik geçişi geçerlidir.*

Bu makine sadece a işlemini yapar. Eğer $a \in \sum$ *ise a yazılır,* $a \in \{\leftarrow,\rightarrow\}$ *ise sola veya sağa gidilir ve makine durur.*

Yazma makinesi çok sık kullanılacağı için $\boldsymbol{M_a}$ *yerine kısaca* ***a*** *kullanılır.*

Head-moving machines:

$M_\leftarrow$ *ve* $M_\rightarrow$ *şeklindedir ve kısaca* ***L*** *(left) ve* ***R*** *(right) makinesi olarak gösterilir.*

Turing Machines

Tanım: *Karmaşık TM'ler basit TM'leri (a, L, R) birleştirerek oluşturulabilir.*

Makine birleştirme kuralları:

- Makineler FA'daki durumlar gibidir ve durumların bağlanması şeklinde birleştirilir
- Bir makineden diğerine yapılan **bağlantı** ilkinin **halt durumuna geçmesiyle** çalışır ve ikinciye geçilir.
- İkinci makine başlangıç durumuyla çalışmaya başlar.

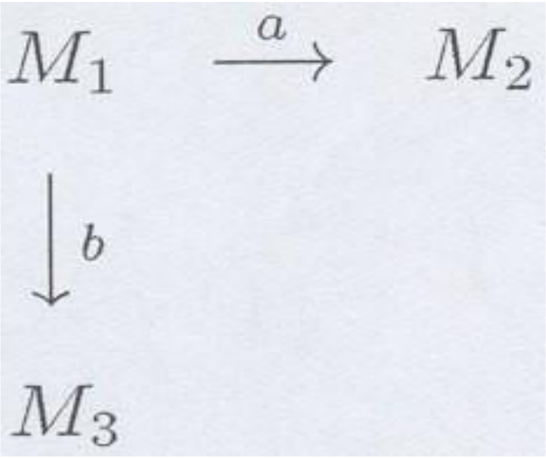
Örnek: *Yandaki şekilde M_1, M_2 ve M_3 Turing makinesidir.*

M_1 başlangıç durumunda çalışmaya başlar.

M_1 halt durumuna geçince okunan sembol :

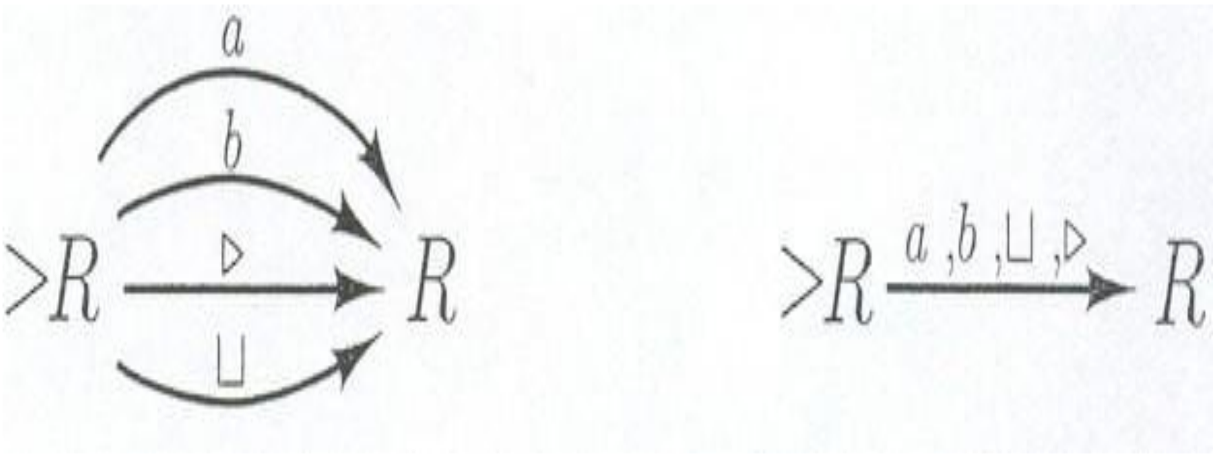
a ise M_2 başlangıç durumunda çalışmaya başlar,

b ise M_3 ba,5langıç durumunda çalışmaya başlar.



Turing Machines

Örnek: iki *R makinesi aşağıdaki gibi birleştirilsin.*



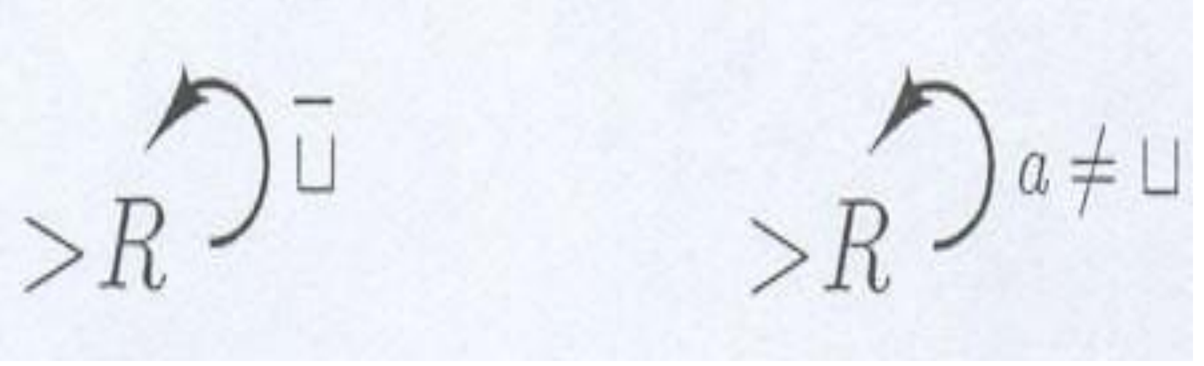
- *Bu makine okuma kafasını önce bir sağa geçirir ve okunan sembol a, b, Δ veya ⊔ ise bir sağa daha geçirir.*
- *Eger bir geçiş alfabedeki tüm sembolleri içerirse etiket yazılmadan* $R \rightarrow R$ *şeklinde gösterilir. Daha da basitleştirilerek ise*

$$RR \quad veya \quad R^2$$

şeklinde gösterilebilir.

Turing Machines

Örnek: Eger $a \in \sum$ ise, birçok sembol kullanılan oklar yerine a (sembolik ifade) şeklinde gösterim de kullanılabilir.

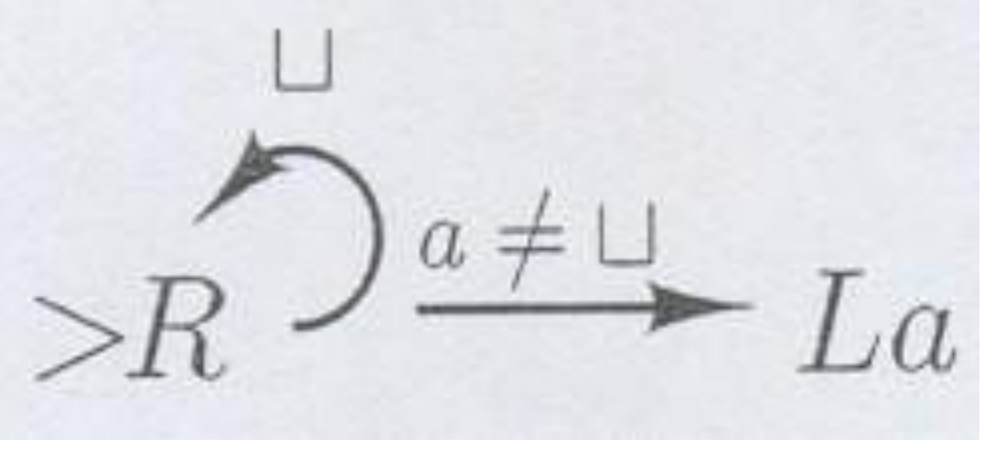


- *Yukarıdaki şekilde soldaki makine* $\sqcup$ *bulana kadar sağa gider ve* $R_{\sqcup}$ *şeklinde gösterilir.*

- *Yukarıdaki sağdaki şekil de aynı işlemi ifade etmektedir. Ancak okunan a (sembolik ifade) sembolünün daha sonra kullanılmasını sağlamaktadır.*

Turing Machines

Örnek: Aşağıdaki makine bir sembol bulana kadar sağa gider ve bulduğu sembolü bir soldaki alana kopyalar.

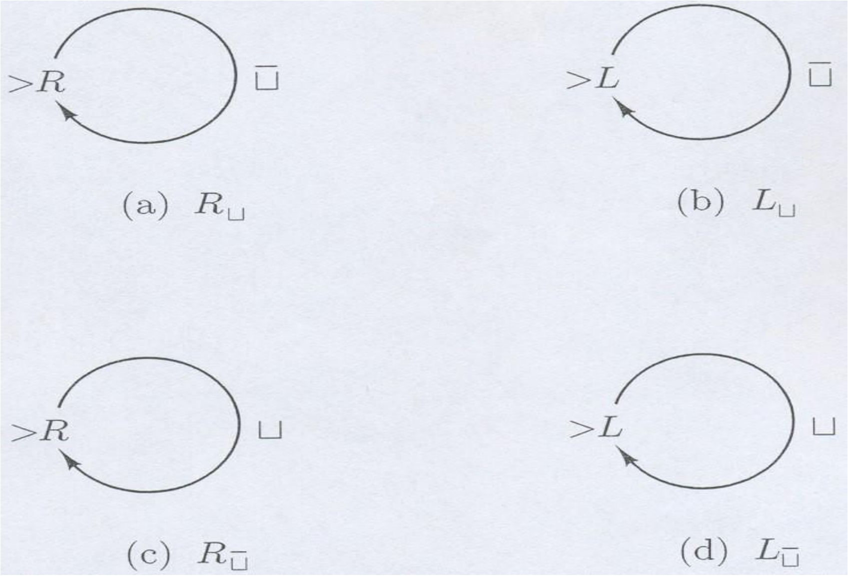


La bir sola gitmeyi ve en son okunan $a \in \Sigma$ *sembolünü yazmayı ifade etmektedir.*

Turing Machines

Örnek: Aşağıdaki makineler hep sağa veya sola gider ve bir sembol arar.

Aradığını bulur bulmaz çalışması sonlanır.



(a)$R_{\sqcup}$, sağa doğru tarama yapar ve ilk bulduğu boşlukta durur.

(b)$L_{\sqcup}$, sola doğru tarama yapar ve ilk bulduğu boşlukta durur.

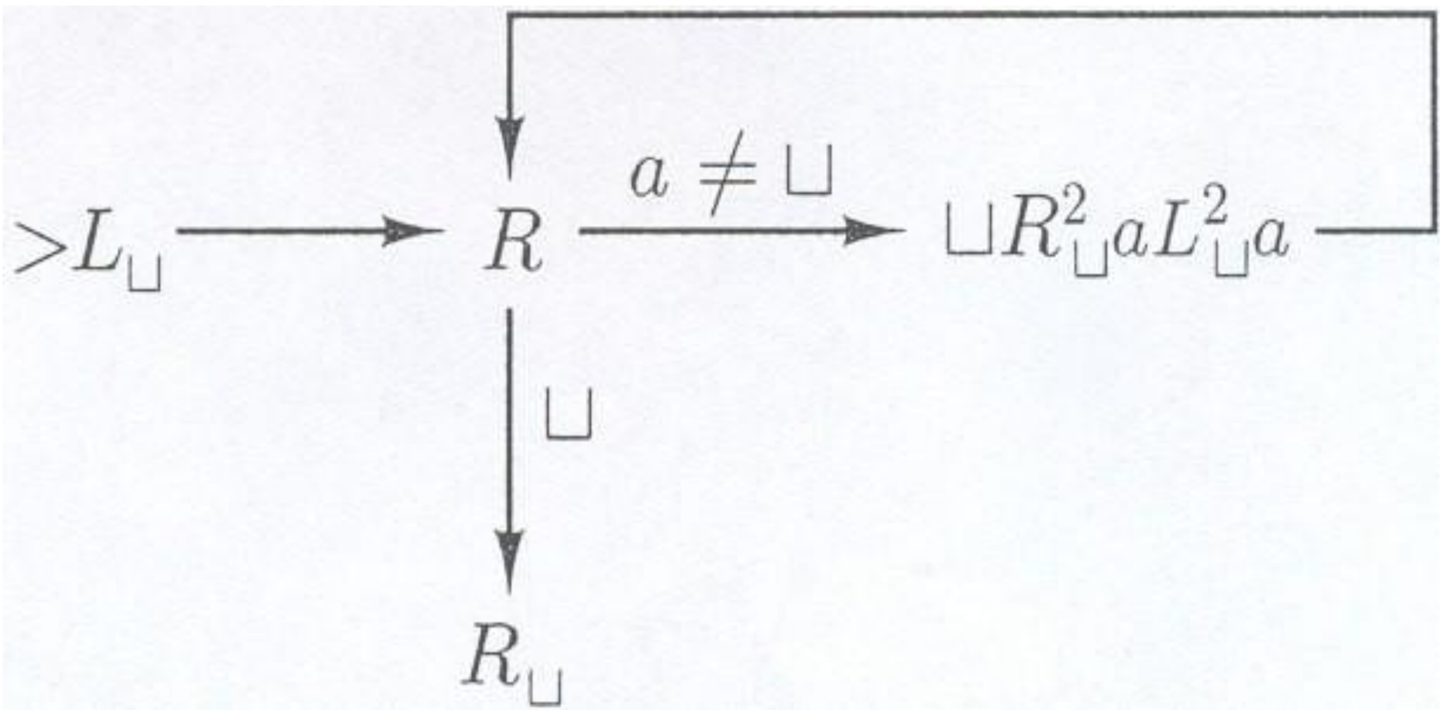
(c)$R_{\overline{\sqcup}}$, sağa doğru tarama yapar ve ilk bulduğu sembolde durur.

(d)$L_{\overline{\sqcup}}$, sola doğru tarama yapar ve ilk bulduğu sembolde durur.

Turing Machines

Örnek: Aşagıdaki birleşik TM ne işlem yapar?

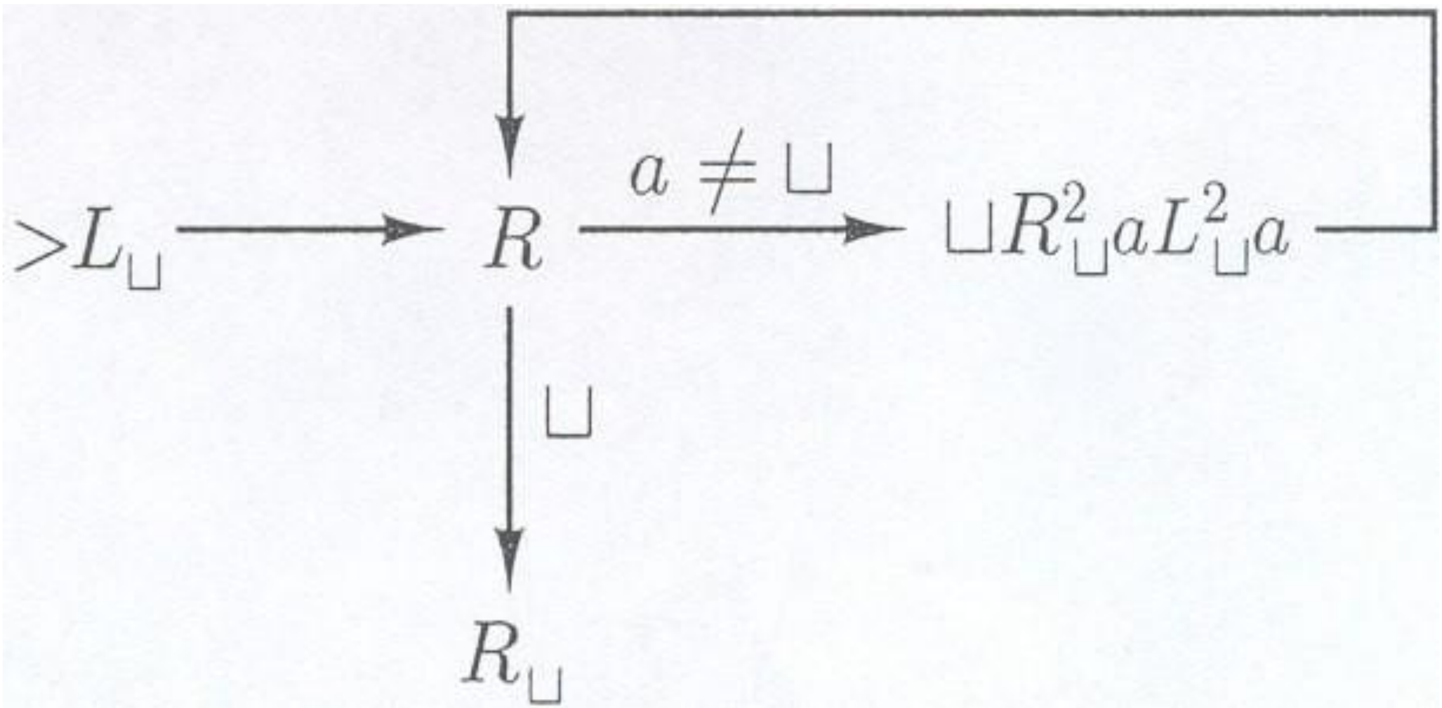
$\square w \underline{\square}$ *string'i için sonuç string'i ne olur?*



Turing Machines

Örnek: Aşagıdaki birleşik TM ne işlem yapar?

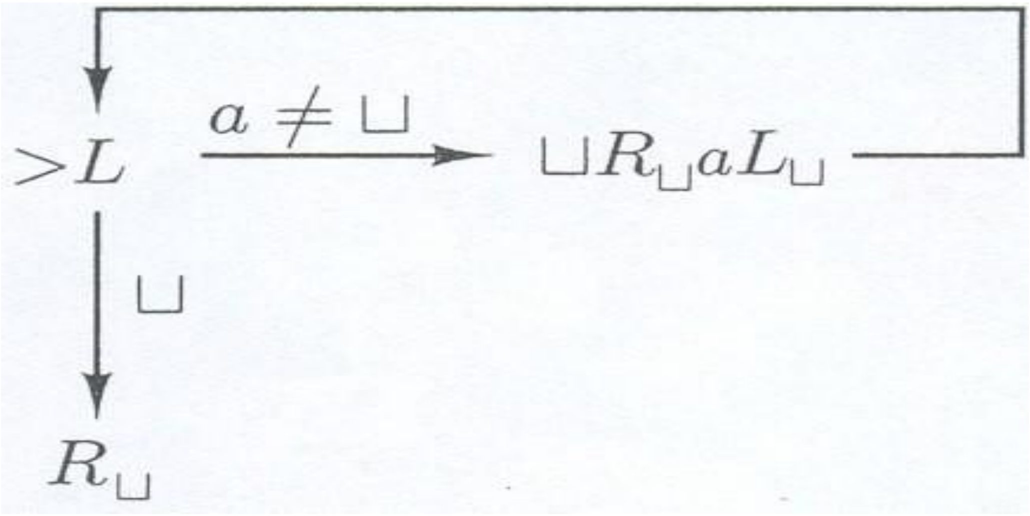
□w<u>□</u> string'i için sonuç string'i ne olur.



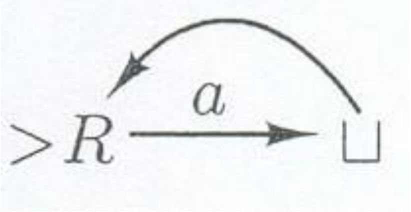
Turing Machines

Örnek: Aşağıdaki sağa kaydırma makinesi bir w stringini bir sağa kaydırır.

$\sqcup w \underline{\sqcup}$ *string'i için sonuç string'i* $\sqcup\sqcup w \underline{\sqcup}$ *olur.*



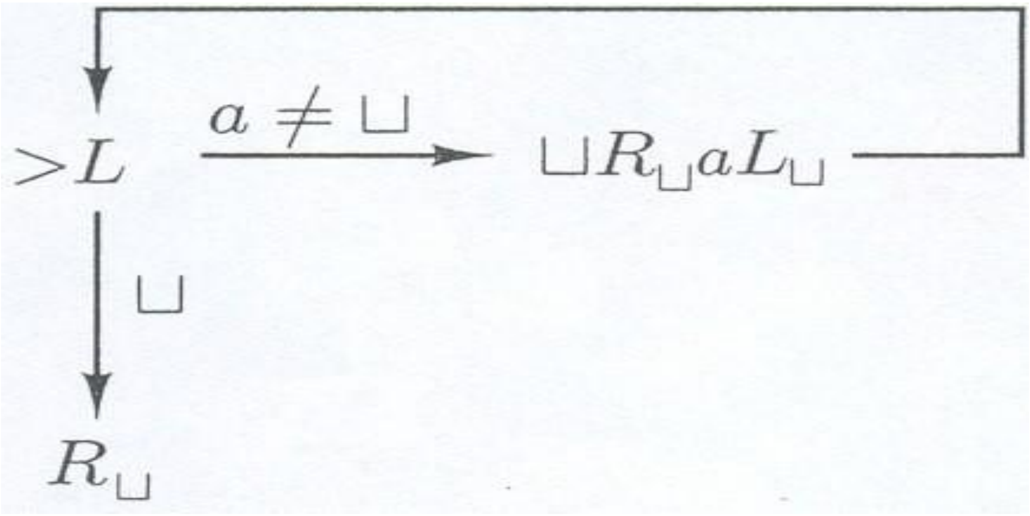
Örnek: Aşağıdaki makine ne işlem yapar?



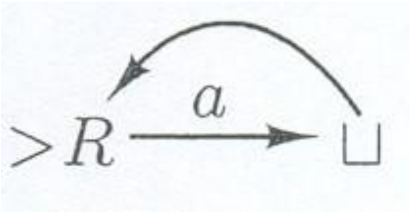
Turing Machines

Örnek: Aşağıdaki sağa kaydırma makinesi bir w stringini bir sağa kaydırır.

□w□ string'i için sonuç string'i □□w□ olur.



Örnek: Aşağıdaki makine tape üzerindeki tüm a ları siler.



Computing with Turing Machines

- *Turing makinesinde işlemler için gerekli giriş string'i Δ sembolünün sağına yazılır ve içinde boşluk sembolü yoktur.*
- *Giriş string'inin sağındaki kısım tümüyle boşluk sembolüdür.*
- *Bundan sonraki örneklerde giriş string'i ile Δ sembolü arasında bir boşluk sembolü* ($\sqcup$) *vardır.*
- *Eger $M = (K, \Sigma, \delta, s, H)$ bir Turing makinesi ve*

  $w \in (\Sigma - \{ \sqcup, \Delta \})^*$ ise M makinesinin w girişi için başlangıç konfigürasyonu $(s, \Delta\underline{\sqcup}w)$ şeklindedir.

Computing with Turing Machines

Tanım : *M = (K, ∑, δ, s, H) bir Turing makinesi*

H = {y, n}

şeklinde iki tane halt state'e sahip olsun. Bunlar

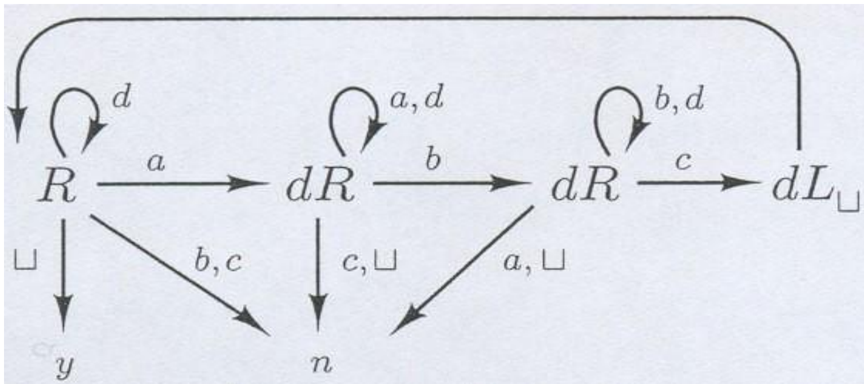
*y kabul konfigürasyonu **(accepting configuration)** ve*

*h red konfigürasyonu **(rejecting configuration)** olsun.*

- *Eger w∈ (∑ – {⊔, Δ })* girişi için (s, Δ$\underline{⊔}$w) konfigürasyonu M makinesini accepting configuration'lardan birisinde sonlandırırsa w bu dile aittir, halting configuration'lardan birisinde sonlandırırsa dile ait değildir.*
- *M makinesi bir dili belirler (**decide**) ve*
    - *eğer w∈ L ise w string'ini kabul eder*
    - *eger w∉ L ise w string'ini red eder*
- *Bir dili belirleyen bir TM varsa bu dil özyinelemeli dil (**recursive**) olarak adlandırılır.*

Computing with Turing Machines

Örnek : *L = {$a^n b^n c^n$: n ≥0} dilini tanıyan Turing makinesi aşağıdadır.*



- *M makinesi n döngü yapar. Her döngüde makine, girişin en solundan başlar ve ilk buldugu a yerine d, ikinci buldugu b yerine d ve üçüncü olarak buldugu c yerine d yazar.*
- *Okuma kafası yeniden string'in en soluna gider.*
- *Makine bir a ararken b veya c ye, b ararken c veya ⊔'ya, c ararken a veya ⊔'ya rastlarsa n durumuna gider.*
- *Eger bir a ararken ⊔ gelirse (sağ kısmın tamamı d olmuştur) çalışmasını y durumuna geçerek sonlandırır.*

Computing with Turing Machines

Tanım : *$M = (K, \Sigma, \delta, s, \{h\})$ bir Turing makinesi ve $\Sigma_0 \subseteq \Sigma - \{ \sqcup, \Delta \}$ ve $w \in \Sigma_0$ olsun.*

Eger M makinesi, w girişi için halt state'lere ulaşıyorsa ve bazı y'ler için

$$(s, \Delta \underline{\sqcup} w) \vdash^*_M (h, \Delta \underline{\sqcup} y) \text{ ise,}$$

y durumu M makinesinin çıkışı olarak adlandırılır ve M(w) şeklinde gösterilir.

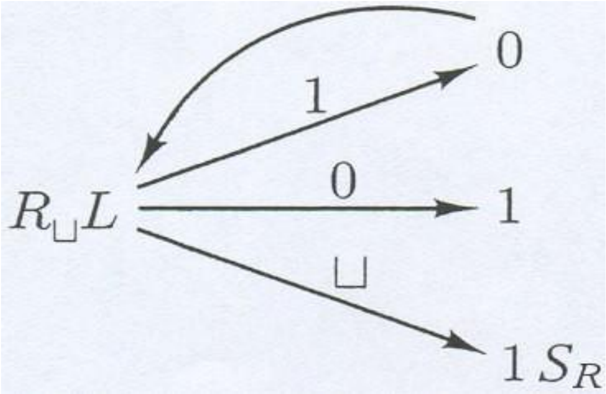
M(w) sadece makinenin halt durumuna ulaşması halinde tanımlıdır.

*Eğer bir **f** fonksiyonu Σ_0 dan Σ_0 a tanımlı ve tüm $w \in \Sigma_0$ için $M(w) = f(w)$ ise M makinesi **f** fonksiyonunu hesaplar.*

*M makinesinin çalışması bittiğinde tape üzerinde $\Delta \underline{\sqcup} f(w)$ vardır ve bu fonksiyona özyinelemeli (**recursive**) denilir.*

Computing with Turing Machines

Örnek : Aşağıda binary olarak yazılmış sayının bir fazlasını hesaplayan bir TM görülmektedir. (succ(n) = n + 1)



- *M makinesi önce girişin en sağını bulur.*
- *Sonra 1 gördüğü sürece sola gider ve her 1 değerini 0 olarak değiştirir.*
- *ilk gördügü 0 yerine 1 yazarak çalışmasını sonlandırır.*
- *Eğer sola giderken ⊔ sembolü görürse yerine 1 yazar ve tüm girişi sağa bir pozisyon shift ederek çalışmasını sonlandırır.*

Computing with Turing Machines

Tanım: $M=(K, \sum, \delta, s, H)$ *bir Turing makinesi,* $\sum_0 \subseteq \sum - \{\sqcup, \Delta\}$

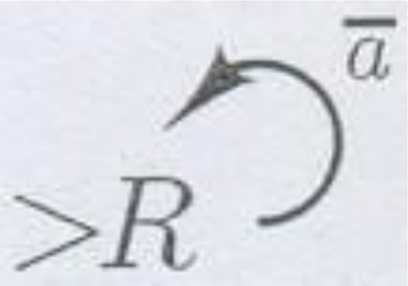
alfabe ve $L \subseteq \sum_0{}^*$ *olsun.*

- *Eger sadece* $w \in L$ *iken M makinesi halt durumuna geçerse, M makinesi L dilini yarı belirler (**semidecides**) denir.*

- *Bir dil bir TM tarafından semidecide ediliyorsa bu dil özyinelemeli -sıralı (**recursively enumerable**) olarak adlandırılır.*

Computing with Turing Machines

Örnek: *L={w ∈ {a, b}*: w içinde en az bir a vardır} şeklinde tanımlı bir dil aşağıdaki TM tarafından semidecide edilir.*



- *w ∈ {a, b}* * *girişi için makine* $(q_0, \Delta\sqcup w)$ *başlangıç konfigürasyonundan çalışmaya başladığında sağa doğru ilk a okuduğunda çalışmasını sonlandırır.*
- *Eger a bulamazsa sonsuza kadar çalışır ve hiçbir zaman halt durumuna ulaşamaz.*
- *M makinesi L dilini semidecide yapar ve L recursively enumarable dildir.*