

Otomata Teorisi (BİL 2114)

Fırat İsmailođlu

Hafta 12:
Turing Makinesi (Bölüm 3)



Hafta 12

Plan

1. Çok Bantli TM'ni Tek Bantli TM'ye Denkligi
2. Nondeterministik TM Örneđi
3. Karar Verilebilirlik (Decidability)
4. Bir Problemi Bir Dile Dönüştürmek
5. Halting Problemi
6. Evrensel Turing Makinesi



Çok Bantlı TM'nin Tek Bantlı TM'ye Denkliği

Teorem: Her çokbantlı TM'nin denk olduğu bir tek bantlı (standart) TM vardır. Yani bu tek bantlı TM, çok bantlı TM'nin tanıdığı her dili tanır, aynı görevleri yerine getirir.

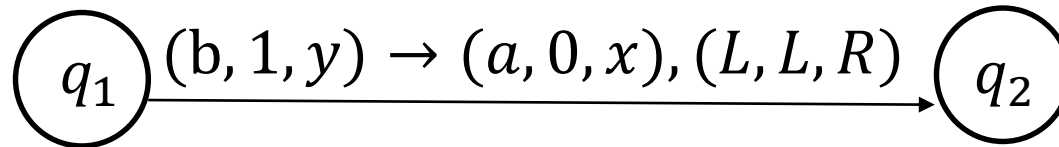
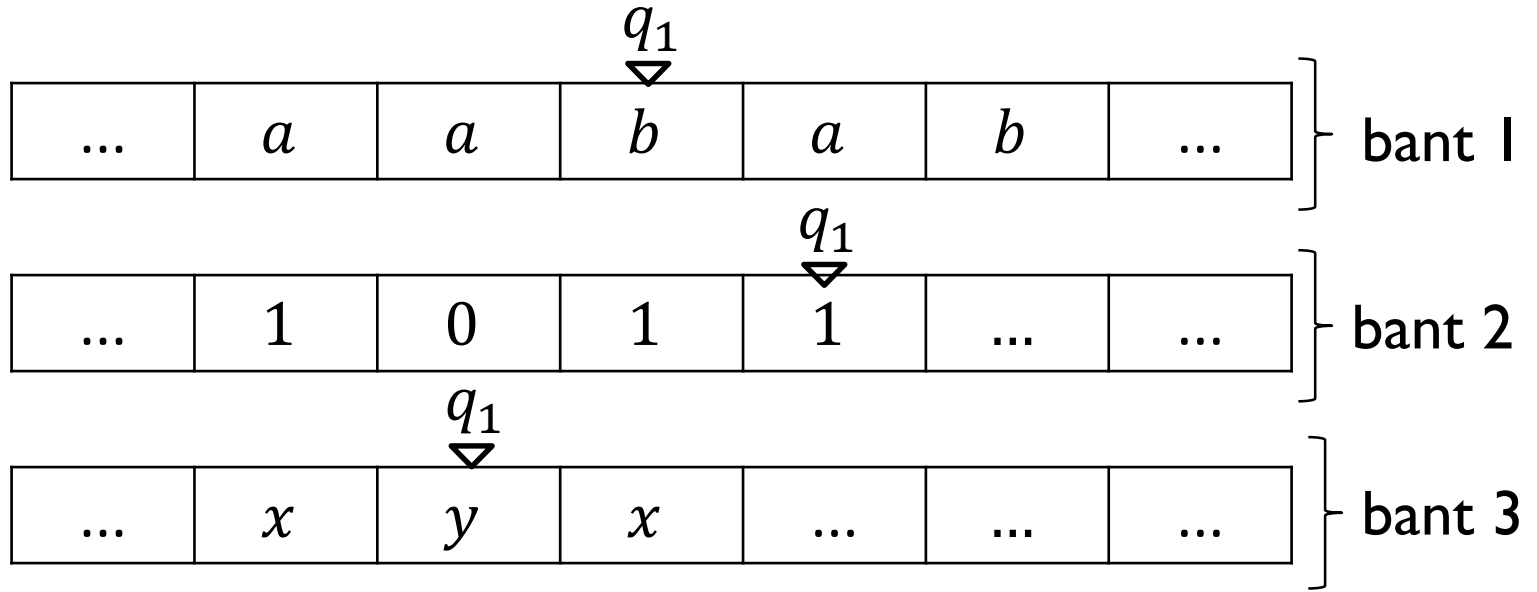
İspat: Bu teoremi ispatlamak için verilen bir çokbantlı TM'ye karşılık buna denk bir tek bantlı TM bulabileceğimizi göstereceğiz. Bunun için

1. Tüm bantları tek bir bantta depolayabilemiz gerekir.
2. Her bir bantın kendi okuma-yazma kafasının olması gerekir.
3. Çok bantlı TM'deki hareketlerin (her bir okuma-yazma kafasının sağa sola gitmesi) tek bantlı TM'de ifade edilebilmesi gerekir.



Çok Bantlı TM'nin Tek Bantlı TM'ye Denkliği

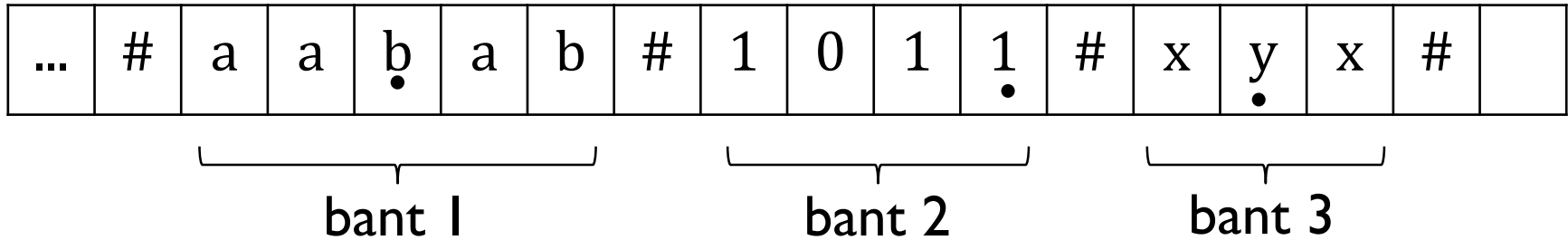
ör.



Şimdi bu bantları tek bir banta dönüştürelim..



Çok Bantlı TM'nin Tek Bantlı TM'ye Denkliği

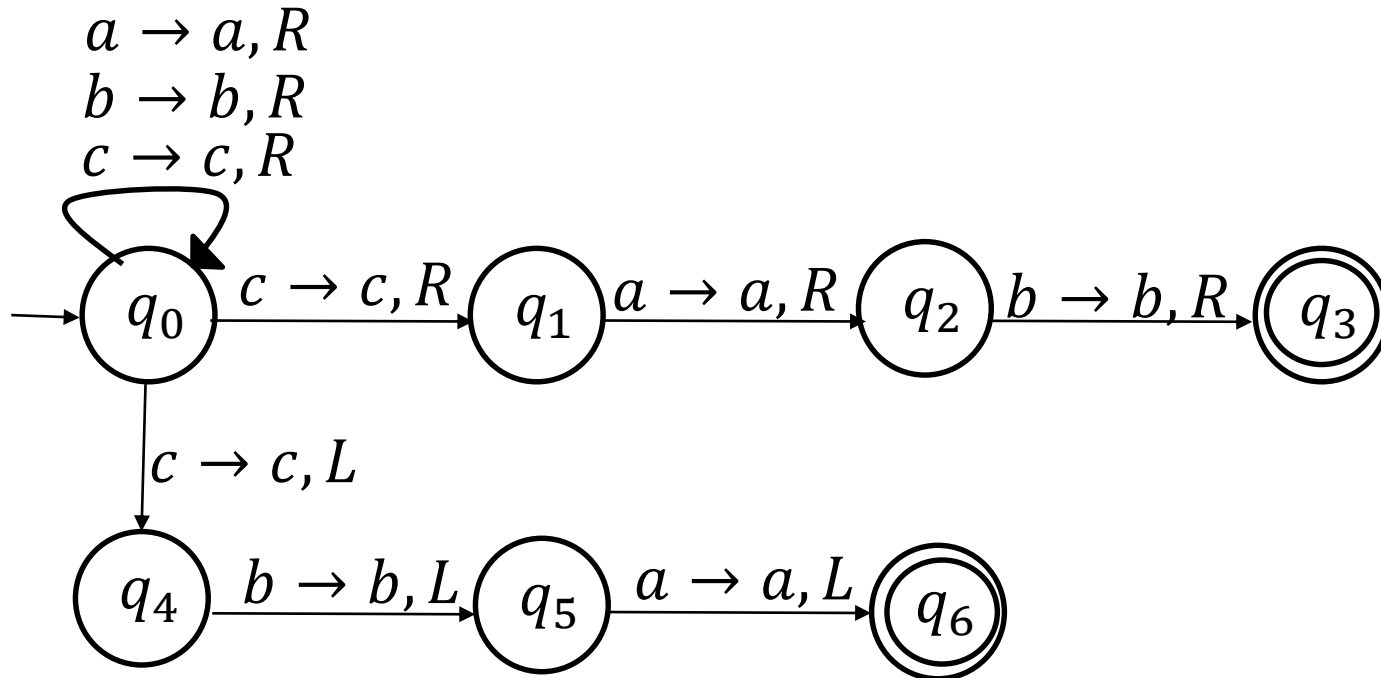


- ✦ Bantlarda yazılan kelimeler yanyana konur, ve bu kelimeler # ile ayrılır.
- ✦ Okuma-yazma kafalarının oldukları yeri işaretlemek için bu kafaların olduğu hücrelere harfle birlikte • yazılır.
- ✦ Herhangi bir kafanın #'ye gelme durumunda okunan hücreye B yazılır.
- ✦ Çok bantlı bir TM nin bir hareketi, tek bantlı TM'de, (tek) kafanın soldan başlayarak içinde • olan her hücreye ayrı ayrı gitmesi , oradaki harfi okuması-yazması ve daha sonra • sağa yada sola koyması şeklinde karşılık bulur.



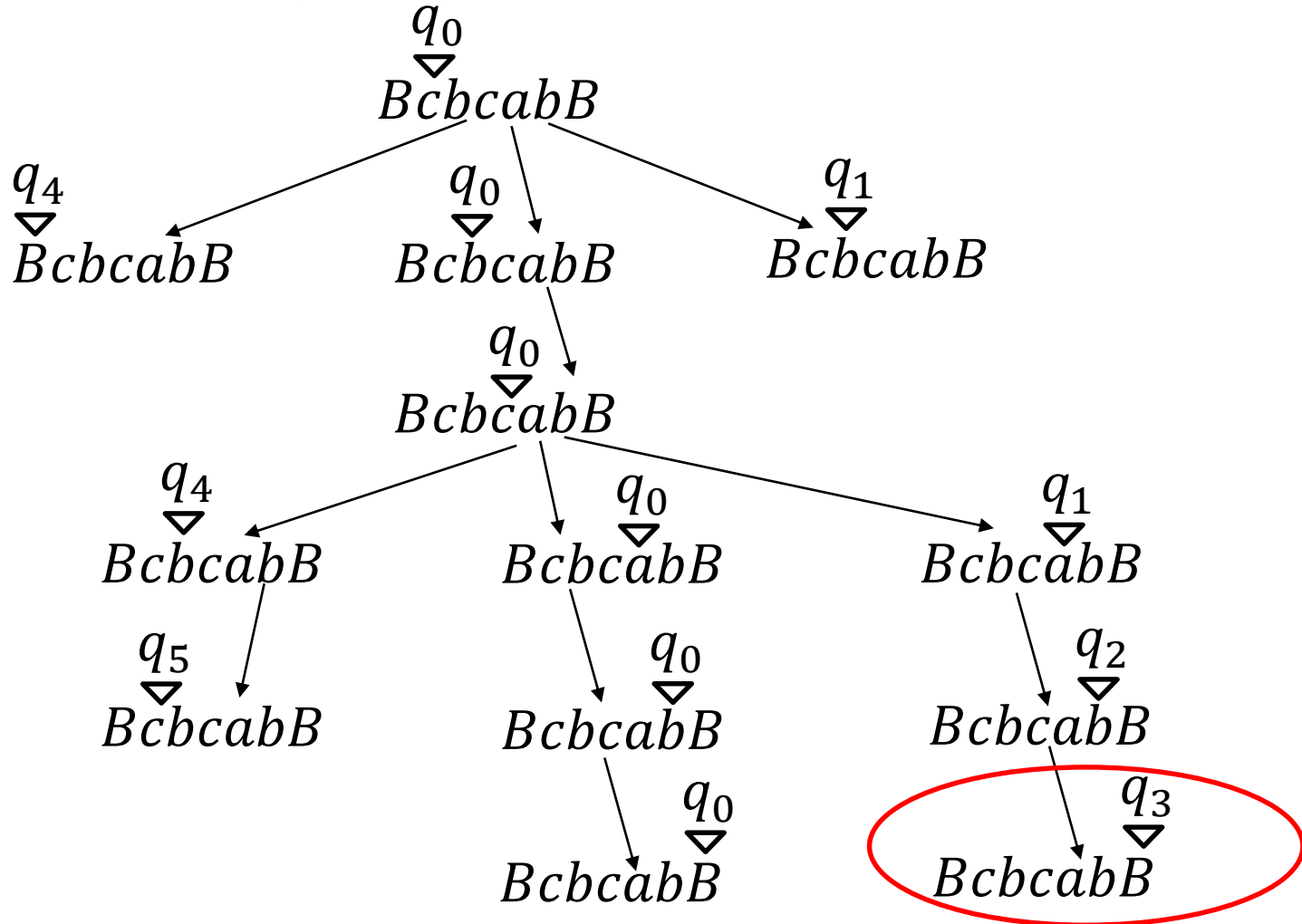
Nondeterministik Turing Makinesi (NTM)

ör. Aşağıdaki NTM $\Sigma = \{a, b, c\}$ harfleri kullanılarak üretilen kelimelerden içinde 'cab' yada 'abc' geçen kelimeleri tanır.



Nondeterministik Turing Makinesi (NTM)

Örnek olarak $w = cbcab$ kelimesini banta yazıp okuyalım. Bu kelimenin NTM de işlenmesini ağaç diyagramı ile göstereceğiz.



Karar Verilebilirlik (Decidability)

Bir problemin karar verilebilir (hesaplanabilir) olması için; bu problem için bir Turing makinesini bulabiliyor olmamız gerekir oyleki bu TM problemin her bir örneği için kabul yada değil kararini verebiliyor olması gerekir.

Yani bu TM hiç bir zaman sonsuz döngüye girmez (her zaman bir durumda durur (“halt” olur)).

Boyle bir TM'nin bulunabilmesi demek o problemi çözen-hesaplayan bir algoritmanın bulunabilmesi demektir.

Bir problemin karar verilebilirliğine bakılırken öncelikle bu problem bir dile dönüştürürüz.



Bir Problemi Bir Dile Donusturmek

Bir problem bir dile donusturuldugunde, problemin her bir ornegi dilin bir kelimesi olur. Boylece bu dile karar veren TM'nin bu kelimeyi kabul etmesi demek verilen problemin cevabinin evet olmasi anlamina gelir.

Kelime dilin elemanıdır \equiv Problemin cevabi evet
Kelime dilin elemanı degildir \equiv Problemin cevabi hayir

or. Problem: Verilen bir graf bagli midir (connected graph) ?

Bu probleme karsilik gelen dil:

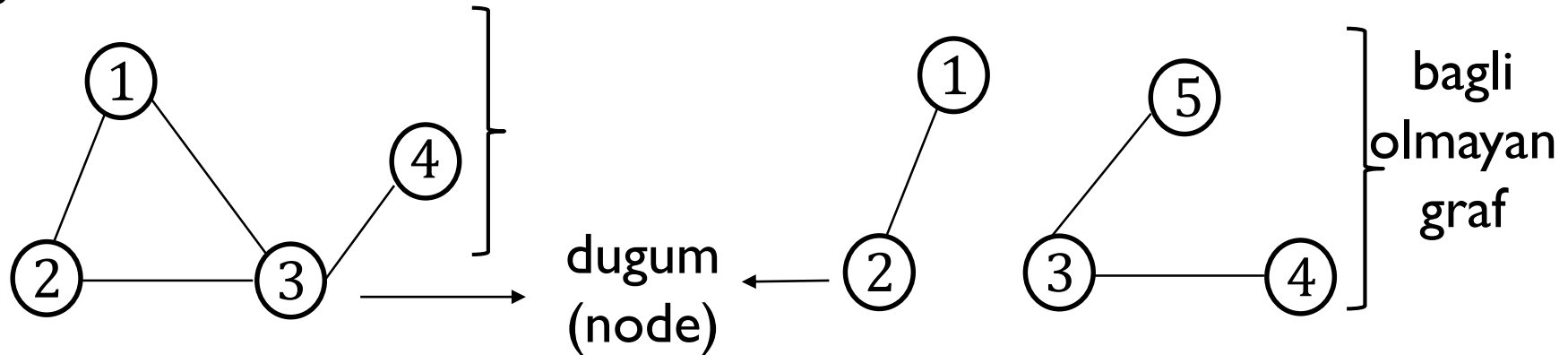
$$L = \{ \underbrace{\langle G \rangle}_{G \text{ grafinin kelime}} \mid G \text{ bagli bir graftir} \}$$

G grafinin kelime
olarak gosterilmis hali



Bagli Graf

Bagli graf (connected graph): Eger bir grafın her bir dugumune (node) diger butun dugumlardan ulasabiliyorsak bu graf bagli bir graftir.



Yukaridaki bagli grafın kelime olarak gosterimi:

$$\langle G \rangle = \underbrace{(1,2,3,4)}_{\text{dugumler listesi}} \left(\underbrace{((1,2), (1,3), (2,3), (3,4))}_{\text{baglar listesi}} \right)$$



Bize verilen bir grafin bagli olup olmadigini anlamak icin bu grafin L dilinin elemani oldugunu gostermemiz gerekir ($\langle G \rangle \in L ?$)

Bu grafi banta su sekilde yazabiliriz:

...	B	(1	,	2	,	3	,	4)	((1	,	2)	...	(3	,	4))	B	...
-----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	-----	---	---	---	---	---	---	---	-----

Bantin alfabetesi $\Gamma = \{ (,), ,, 1, 2, \dots, 9, B, \dots \dots \}$

Bu dile karar veren TM'nin yuksek seviye tanimi soyledir:

1. G' 'nin ilk dugumunu sec ve isaretle.
2. Asagidaki durumu yeni bir dugum isaretlenmeyene kadar tekrar et.
3. G' 'nin her bir dugumu icin, eger bu dugum isaretlenmis bir dugume bagli ise isaretle.
4. G' 'nin tum dugumlerini tara, eger hepsi isaretlenmis ise G' yi kabul et, degilse etme.



Uygulama Detaylari - I

Yuksek seviye tanimini (high level description) verdigimiz TM'yi su sekilde insa edebiliriz.

Oncelikle TM verilen graf uygun formda mi diye banti tarar. Yani iki tane listeden mi olusuyor, ilk listedeki sayilar birbirinden farkli mi, ikinci listedeki sayilar birinci listedeki sayilardan mi olusuyor diye kontrol eder. Boylece girilen grafi control eder.

Yuksek seviye tanimda:

1. durum icin TM en soldaki sayiya gider ve bunu nokta ile isaretler.
2. durum icin TM, dugumlerin listesine denk gelen hucreleri tarar, (nokta ile) isretlenmemis bir sayi gordugunde bunu # ile isaretler, sonra tekrar tum dugumleri tarar. Nokta ile isaretlenmis bir dugum okudugunda bunu da # ile isaretler.



Sonuc olarak verilen bir grafin bagli olup olmadiginin anlasilmasi problemi icin bu probleme karar veren bir TM bulabildigimizden karar-verilebilir bir problemdir.

Halting (Durma) Problemi

Verilen bir programin (C , Java, yada bir Turing Makinesi) her zaman sonlanip sonlanmayacaginin, yani sonsuz donguye girip girmeyeceginin bilinmesi problemine “Halting Problemi” denir.

Halting Problemi karar verilemez bir problemdir. Yani oyle bir program yazamayizki (oyle bir TM olusturamayizki) girdi olarak verilen her program icin bunun her zaman sonlanip sonlanmayacagini bize soylesin.

Hic bir zaman verilen tum problemlerin icinde bug yoktur diyemeyiz. Oyle bir program yazamayizki verilen bir programin icindeki tum hatalari bulsun. Hata yoksa program sonlanir desin.



Evrensel Turing Makinesi (Universal Turing Machine)

TM'nin bir cesit bilgisayar oldugunu soylemistik. Fakat su ana kadar gordugumuz TM'ler yalnızca bir gorev yapıyordu (ornegin, sadece cikarma yada toplama..) .

Gunumuz bilgisayarları ise cok amaclidir, birden fazla gorev yapabiliirler. Bu bilgisayarlar genel olarak bir program ve bu programın bir girdisi verildiginde verilen programı verilen girdi için calistirirlar.

Evrensel Turing Makinesi (ETM) gunumuz cok amacli bilgisayarları gibidir. Bir TM ve bu TM'nin girdilerini girdi olarak alır ve TM'yi verilen girdiler üzerinde calistirarak sonuc uretir.

Yani ETM nin girdisi yine bir TM ve bu TM'nin kendi girdileridir.



Evrensel Turing Makinesi (Universal Turing Machine)

Bu düşünce günümüzde de her zaman kullanılır. Örneğin derleyici (compiler) in kendisi bir programdır, yazılmış programları girdi olarak alır.

ETM'yi inşa etmek için girdi olarak verilen TM'yi bantta yazabilmemiz gerekir. Bunun için tek yapmamız gereken TM'nin geçişlerini ikili (binary) olarak yazmaktır. Bunun genel formulu şöyledir:

$$\delta(q_i, X_j) = (q_k, X_l, D_m) \equiv 0^i 10^j 10^k 10^l 10^m$$

Burada $q_1, q_2, q_3 \dots$ durumlardır. Özel olarak q_1 her zaman başlangıç ve q_2 final durumudur.

$X_1, X_2, X_3, X_4 \dots$ bantın harfleridir. Özel olarak X_1 her zaman 0, X_2 her zaman 1 ve X_3 her zaman B dir. Böylece 0'ı 0 ile, 1'i 00 ile B 'yi 000 ile göstereceğiz.



Evrensel Turing Makinesi (Universal Turing Machine)

D_m , solu yada sağı gösterir. Sol için 0, sağı (R) için 00 a esittir.

Ayrıca geçişler birbirinden 11 kullanarak ayrılır:

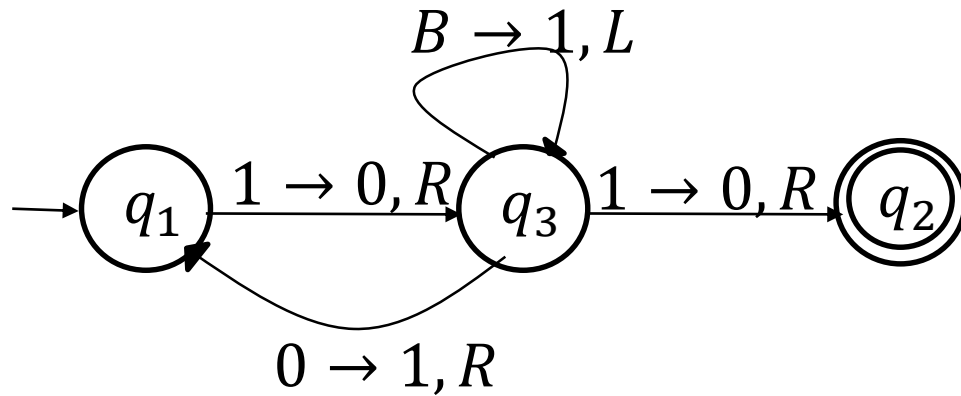
C_i 'ler bir TM'nin geçişleri olmak üzere, n tane geçişe sahip bir TM, ETM'nin bantına şu şekilde yazılır:

$$C_1 11 C_2 11 \dots C_{n-1} 11 C_n$$

ETM'nin bantına TM ile beraber bu TM'nin okuyacağı kelime yazılır. (Bu kelime eğer ikili (0-1) olarak verilmemişse ikili hale dönüştürülür.) ETM'nin bantında TM ve bunun okuyacağı kelime 111 kullanılarak ayrılır.



ör.



Yukarıdaki gibi bir TM ve $w = 1011$ kelimesi verilsin. Bu TM'yi w üzerinde çalıştıracak bir ETM oluşturalım.

Bu TM'nin geçişleri

$$\delta(q_1, 1) = (q_3, 0, R) \equiv 0100100010100$$

$$\delta(q_3, 0) = (q_1, 1, R) \equiv 0001010100100$$

$$\delta(q_3, 1) = (q_2, 0, R) \equiv 00010010010100$$

$$\delta(q_1, B) = (q_3, 1, L) \equiv 0001000100010010$$

ETM'nin bantı:

