Otomata Teorisi

Fırat İsmailoğlu, PhD

Hafta 12:
Turing Makinesi (Bölüm 3)

Hafta 12
Plan

1. Çok Bantli TM'ni Tek Bantli TM'ye Denkligi
2. Nondeterministik TM Örneği
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

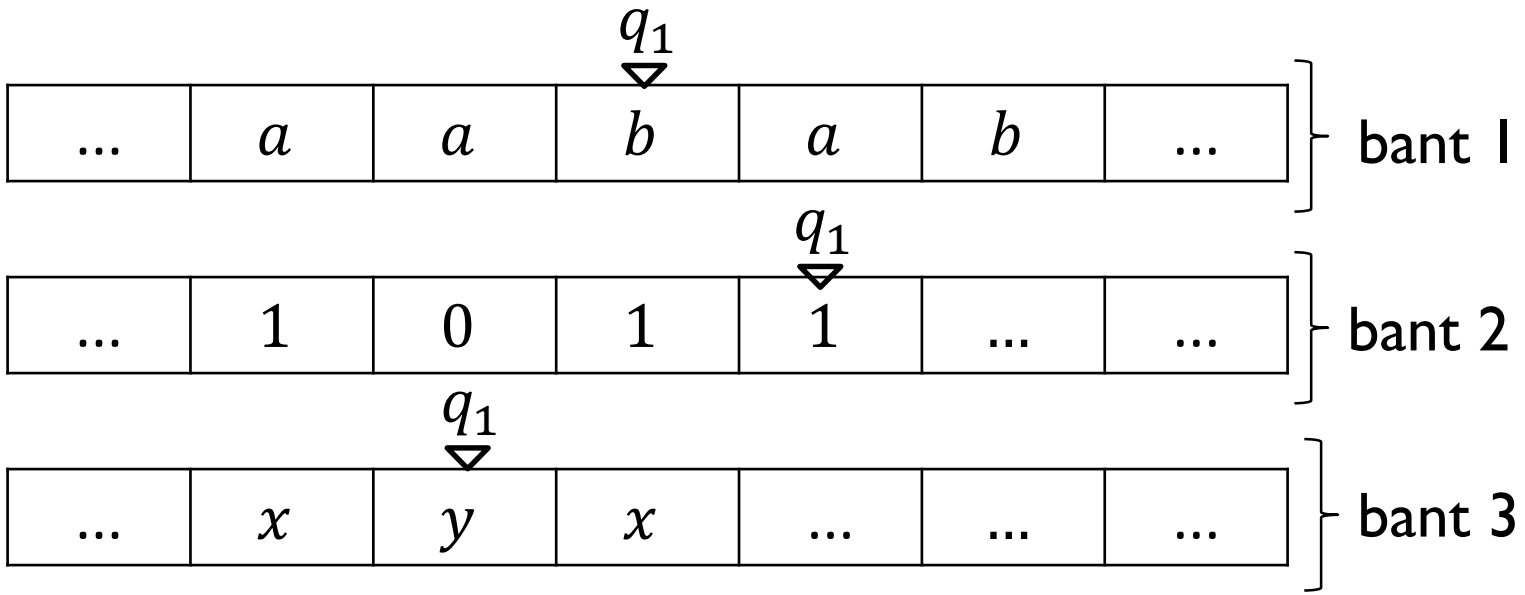
Teorem: Her çokbantlı TM'nin denk olduğu bir tek bantlı (standart) TM vardır. Yani bu tek bantlı TM, çok bantlı TM'nin tanidiği her dili tanır, aynı görevleri yerine getirir.

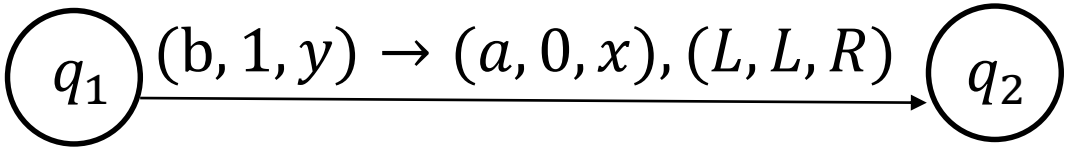
İspat: Bu teoremi ispatlamak için verilen bir çokbantlı TM'ye karşılık buna denk bir tek bantli TM bulabileceğimizi göstereceğiz. Bunun için

1. Tüm bantları tek bir bantta depolayabilemiz gerekir.
2. Her bir bantin kendi okuma-yazma kafasinin olmasi gerekir.
3. Cok bantlı TM'deki hareketlerin (her bir okuma-yazma kafasinin saga sola gitmesi) tek bantli TM'de ifade edilebilmesi gerekir.

Çok Bantlı TM'nin Tek Bantlı TM'ye Denkliği

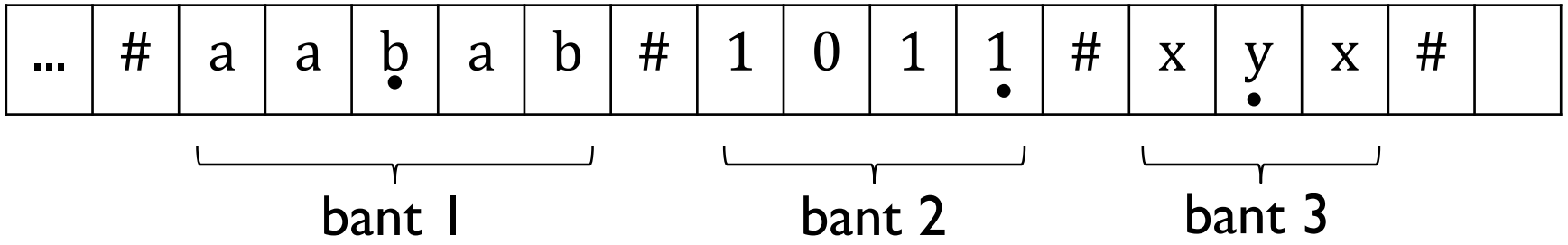
ör.





Şimdi bu bantları tek bir banta dönüştürelim..

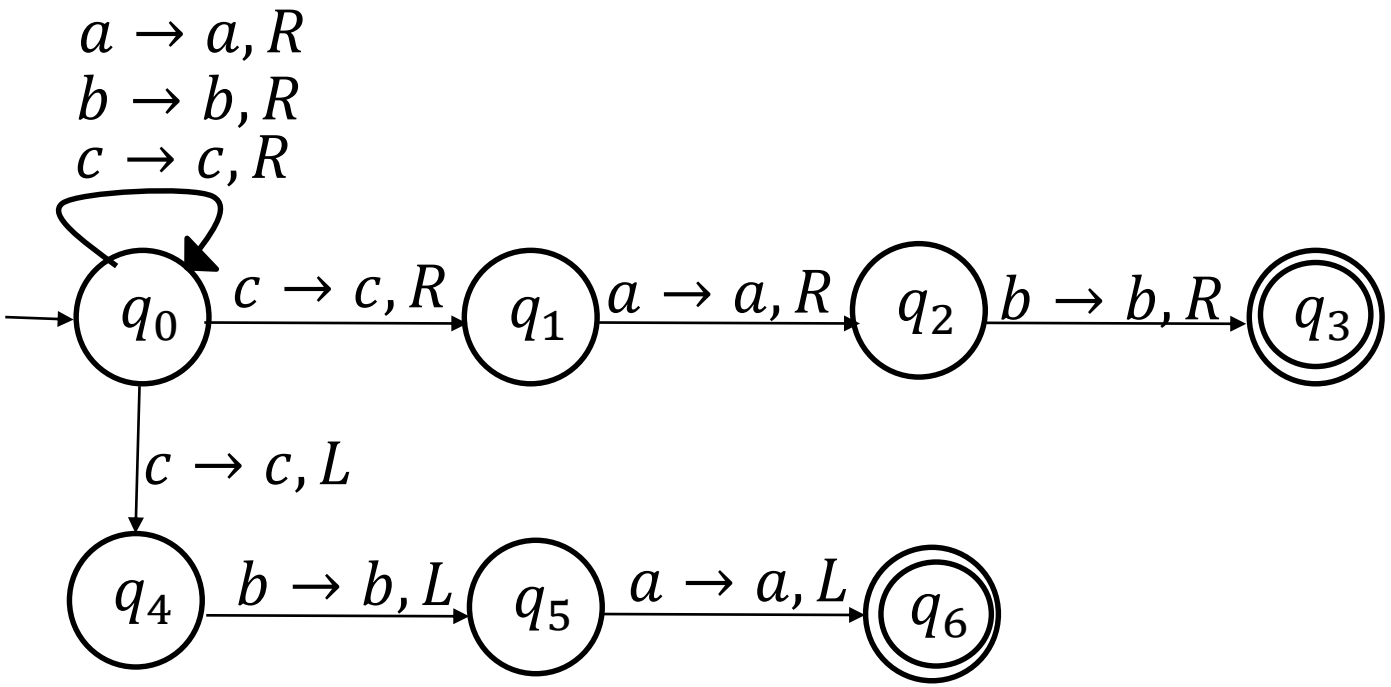
Çok Bantlı TM'nin Tek Bantlı TM'ye Denkliği



- Bantlarda yazılan kelimeler yanyana konur, ve bu kelimeler # ile ayrılır.
- Okuma-yazma kafalarının oldukları yeri işaretlemek için bu kafaların olduğu hucrelere 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 hucreye ayri ayri gitmesi , oradaki harfi okumasi-yazmasi ve daha sonra • sağa yada sola koymasi şeklinde karşilik bulur.

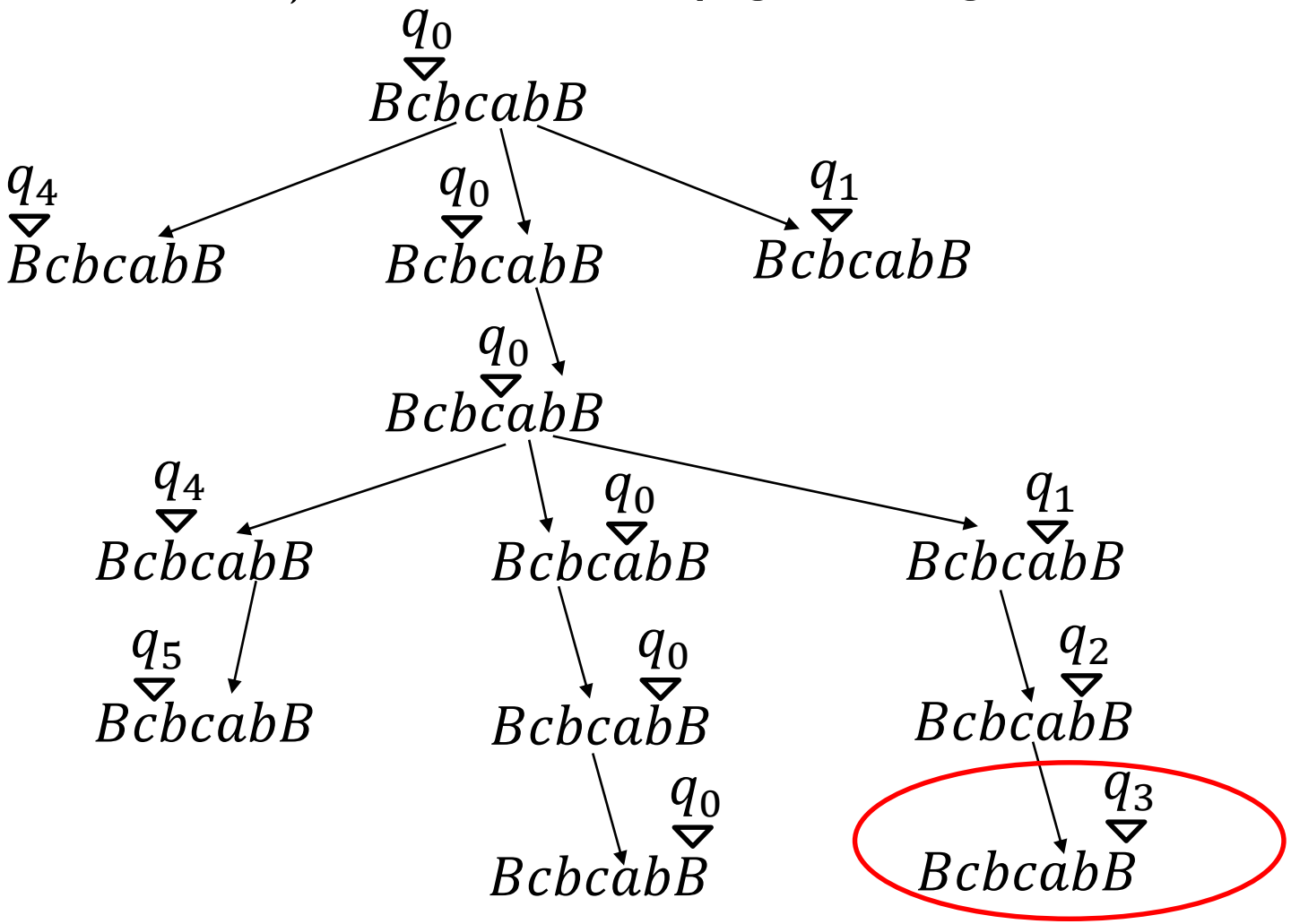
Nondeterministik Turing Makinesi (NTM)

ör. Aşağıdaki NTM $\Sigma = \{a, b, c\}$ harfleri kullanilarak uretilen kelimelerden icinde 'cab' yada 'abc' gecen kelimeleri tanir.



Nondeterministik Turing Makinesi (NTM)

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



Karar Verilebilirlik (Decidability)

Bir problemin karar verilebilir (hesaplanabilir) olmasi icin; bu problem icin bir Turing makinesini bulabiliyor olmamiz gerekir oyleki bu TM problemin her bir ornegi icin kabul yada degil kararini verebiliyor olmasi gerekir.

Yani bu TM hic bir zaman sonsuz donguye girmez (her zaman bir durumda durur ("halt" olur)).

Boyle bir TM'nin bulunabilmesi demek o problemi cozen-hesaplayan bir algoritmanin bulunabilmesi demektir.

Bir problemin karar verilebilirligine bakilirken oncelikle bu problem bir dile donustururuz.

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 elemanidir $\equiv$ Problemin cevabi evet
Kelime dilin elemani degildir $\equiv$ Problemin cevabi hayir

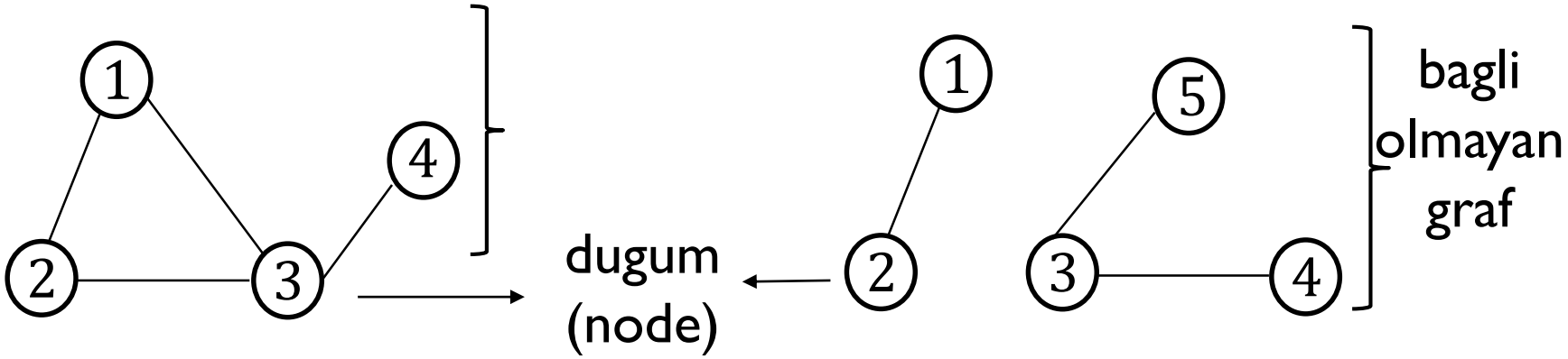
or. Problem: Verilen bir graf bagli midir (connected graph) ?

Bu probleme karsilik gelen dil:

$$L = \{\underbrace{< G >}_{\substack{G \text{ grafinin kelime} \\ \text{olarak gosterilmis hali}}} | G \text{ bagli bir graftir}\}$$

Bagli Graf

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



Yukaridaki bagli grafin kelime olarak gosterimi:

$$< G >= \underbrace{(1,2,3,4)}_{\text{dugumler listesi}}\Big(\underbrace{(1,2),(1,3),(2,3),(3,4)}_{\text{baglar listesi}}\Big)$$

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

Bu grafi banta su sekilde yazabiliriz:

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

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

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 -1

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 girmeyecegininin 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 yalnizca bir gorev yapiyordu (ornegin, sadece cikarma yada toplama..) .

Gunumuz bilgisayarlari ise cok amaclidir, birden fazla gorev yapabiilirler. Bu bilgisayarlar genel olarak bir program ve bu programin bir girdisi verildiginde verilen programi verilen girdi icin calistirirlar.

Evrensel Tuirnmg Makinesi (ETM) gununmuz cok amacli bilgisayarlari gibidir. Bir TM ve bu TM'nin girdilerini girdi olarak alir ve TM'yi verilen girdiler uzerinde calistirarak sonuc uretir.

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

Evrensel Turing Makinesi (Universal Turing Machine)

Bu dusunce gunumuzde de her zaman kullanilir. Ornegin derleyici (compiler) in kendisi bir programdir, yazilmis programlari girdi olarak alir.

ETM'yi insa etmek icin girdi oloarak verilen TM'yi banta yazabilmemiz gerekir. Bunun icin tek yapmamiz gereken TM'nin gecislerini ikili (binary) olarak yazmaktir. Bunun genel formulu soyledir:

$$\delta(q_i, X_j) = (q_k, X_l, D_m) \equiv 0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

Burada q_1, q_2, q_3 ... durumlardir. Ozel olarak q_1 her zaman baslangic ve q_2 final durumudur.

X_1, X_2, X_3, X_4 ... bantin harfleridir. Ozel olarak X_1 her zaman 0, X_2 her zaman 1 ve X_3 her zaman B dir. Boylece 0'ı 0 ile, 1'i 00 ile B'yi 000 ile gosterecegiz.

Evrensel Turing Makinesi (Universal Turing Machine)

D_m, solu yada sağı gosterir. Sol için 0, sağ (R) icin 00 a esittir.

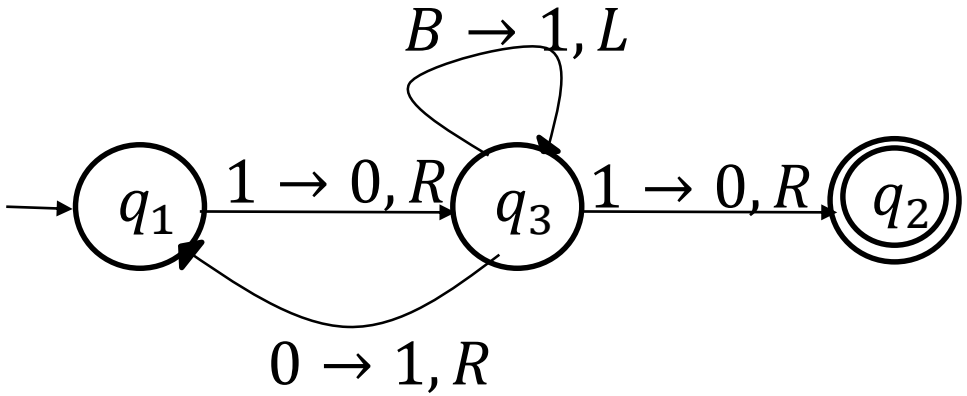
Ayrica geçişler biribirinden 11 kullanarak ayrılır:

C_i'ler bir TM'nin geçişleri olmak uzere, n tane geçise sahip bir TM, ETM'nin bantina şu şekilde yazılır:

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

ETM'nin bantina TM ile beraber bu TM'nin okuyacagi kelime yazilir. (Bu kelime eger ikili (0-1) olarak verilmemisse ikili hale donusturulur.) ETM'nin bantinda TM ve bunun okuyacagi kelime 111 kullanilarak ayrilir.

ör.



Yukaridaki gibi bir TM ve $w = 1011$ kelimesi verilsin. Bu TM'yi w uzerinde calistiracak bir ETM olustruralim.

Bu TM'nin gecisleri

$\delta(q_1, 1) = (q_3, 0, R) \equiv 0100100010100$
$\delta(q_3, 0) = (q_1, 1, R) \equiv 0001010100100$
$\delta(q_3, 1) = (q_2, 0, R) \equiv 0001001001010 0$
$\delta(q_1, B) = (q_3, 1, L) \equiv 0001000100010010$

ETM'nin bantı:

B	B	0	1	0	0	1	0	0	0	1	0	1	0	0	1	1	.	.	0	0	0	1	0	0	0	1	0	0	1	0	1	1	1	1	0	1	1	B	B