Biçimsel Diller ve Otomata Teorisi

Sunu VI
Kleene Kuramı

İZZET FATİH ŞENTÜRK

Kleene Teoremi

- Düzenli ifade veya sonlu otomat veya geçiş grafiği ile tanımlanabilen herhangi bir dil, üç yöntemle de tanımlanabilir

- 1956'da kanıtlanmış tır. Sonlu otomata teorisinin en önemli ve temel sonucu

Kleene Teoremi

- ZAPS kümesinin,
  ZEPS ve ZIPS seti tamamen aynıdır
- Üç parçaya ihtiyacımız var
  - Bölüm I, tüm ZAPS'lerin ZEPS olduğ unu göstereceğ iz
  - Bölüm II, tüm ZEPS'lerin ZIPS olduğ unu göstereceğ iz
  - Bölüm III, tüm ZIP'lerin ZAPS olduğ unu göstereceğ iz
- [ZAPS ZEPS ZIPS ZAPS] [ZAPS = ZEPS = ZIPS]

Kanıt

- Bölüm 1: Bir FA ile tanımlanabilen her dil, bir TG tarafından da tanımlanabilir.
- Bölüm 2: Bir TG tarafından tanımlanabilen her dil, bir RE tarafından da tanımlanabilir.
- Bölüm 3: RE ile tanımlanabilen her dil, FA ile de tanımlanabilir.

Kanıt - Bölüm 1

- En kolay kısım •

Her FA'nın kendisi zaten bir TG'dir

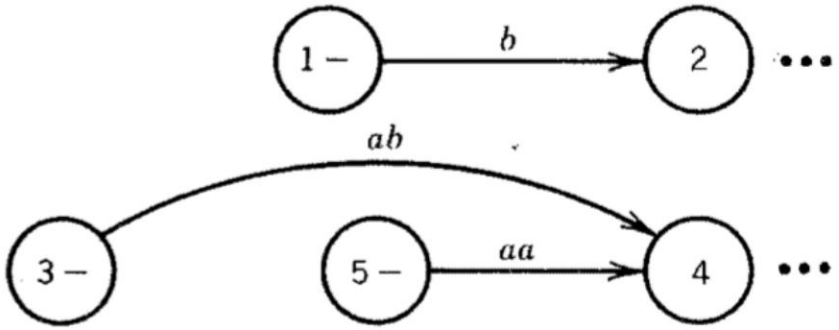
- Bir FA tarafından tanımlanan herhangi bir dil zaten bir TG tarafından tanımlanmış tır

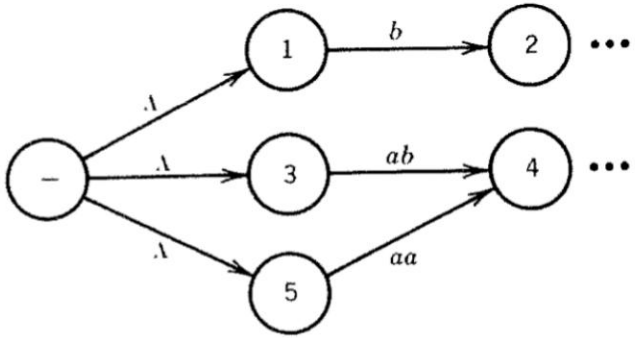
Kanıt – Bölüm 2

- Kanıt yapıcı algoritma ile olacaktır • Aynı dili tanımlayan bir TG ile baş layan ve bir RE ile biten bir prosedür sunuyoruz.
- Kabul edilebilir olması için herhangi bir algoritmanın iki kriteri karş ılaması gerekir • Her TG için çalış ması gerekir • İ ş ini sonlu bir süre içinde (adım sayısı) bitirmeyi garanti etmelidir

Kanıt – Bölüm 2

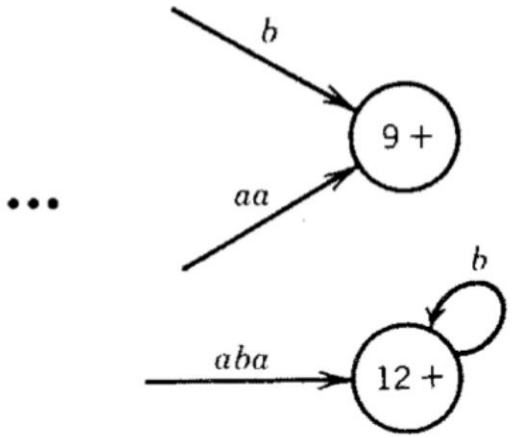
- Soyut bir geçiş grafiği i olan T ile başlayalım.
  - T'nin birçok başlangıç durumu olabilir. İlk önce T'yi basitleştirmek istiyoruz, böylece yalnızca bir başlangıç durumu vardır
  - Eksi işaretiyle etiketlediğimiz ve önceki tüm başlangıç durumlarına Λ etiketli kenarlarla bağladığımız yeni bir durum tanıtın.
  - Önceki başlangıç durumlarından eksi işaretlerini kaldırın

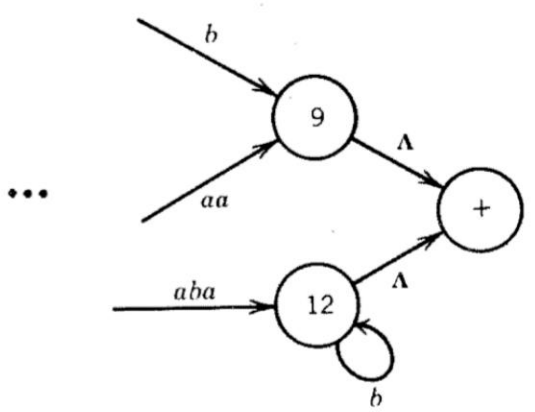




Kanıt – Bölüm 2

- Yapabileceğimiz başka bir basitleştirme, kabul ettiği dili değiştirmeden benzersiz bir nihai duruma sahip olmaktır.
  - Eğer T'nin nihai durumu yoksa, o zaman hiçbir dize kabul etmez ve dili yoktur. Sıfırdan (φ) başka bir RE üretmemize gerek yok
  - Eğer T'nin birkaç nihai durumu varsa, bunların sonunu kaldıralım ve artı işaretiyle etiketlenmiş yeni, benzersiz bir son durum tanıtalım. Tüm eski nihai durumlardan, her biri Λ ile etiketlenmiş yenisine kenarlar çiziyoruz.





Kanıt – Bölüm 2

- Benzersiz nihai durumun, benzersiz baş langıç durumundan farklı bir durum olmasını ş art koş acağ ız • Eski bir durumda ± varsa, o zaman her iki iş aret de ondan kaldırılır
  yeni oluş turulan devletlere
- Unutulmamalıdır ki yeni durumlar,

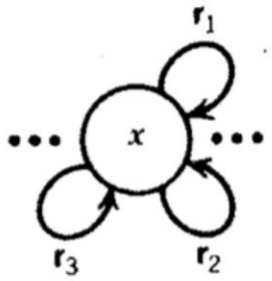
  T'nin kabul ettiğ i dil • T

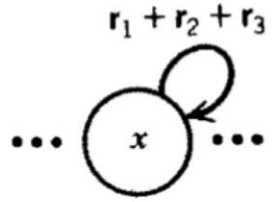
  tarafından kabul edilen herhangi bir kelime yeni makine tarafından da kabul edilir

  • T tarafından reddedilen herhangi bir kelime yeni makine tarafından da reddedilir

Kanıt – Bölüm 2

• Şimdi T ile aynı dili tanımlayan RE'yi parça parça oluşturacağız • T'yi bir GTG olarak değiştireceğiz • Varsayalım ki T'nin içinde bir x durumu var (- veya + durumu yok) • x birden fazla var döngü • Üç döngüyü, RE etiketli bir döngüyle değiştirebiliriz.



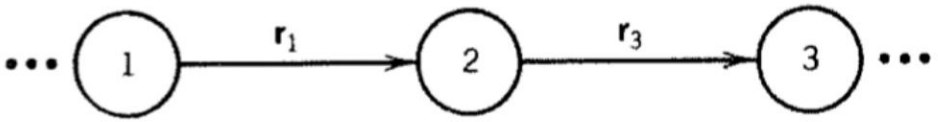


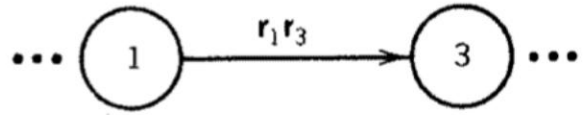
Kanıt – Bölüm 2

- Benzer şekilde, iki durumun aynı yönde giden birden fazla kenarla birbirine bağlı olduğunu varsayalım.
  - Bunu, RE ile etiketlenmiş tek bir kenarla değiştirebiliriz.

Baypas ve Durum Eleme Operasyonu

- RE'ler (veya basit diziler) ile iş aretlenmiş kenarlarla birbirine bağ lanan bir satırda üç durumumuz varsa
  - Aracıyı ortadan kaldırabilir ve önceki etiketlerin birleş tirilmesi olan bir RE ile etiketlenmiş yeni bir kenarla doğ rudan bir dış durumdan diğ erine gidebiliriz.

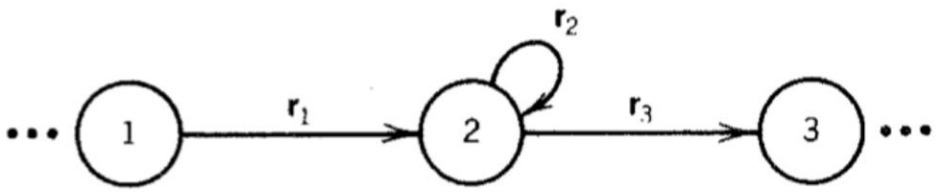


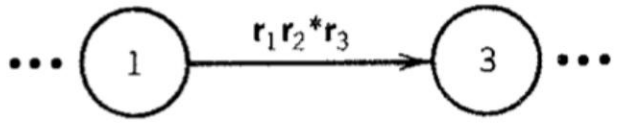


- Durum 1'den durum 2'ye ve durum 2'ye eski kenarları tutmuyoruz 3 belirtmek
  - Durum 1'den durum 3'e kadar olanlar dış ındaki yollarda kullanılmadıkları sürece

Baypas ve Durum Eleme Operasyonu

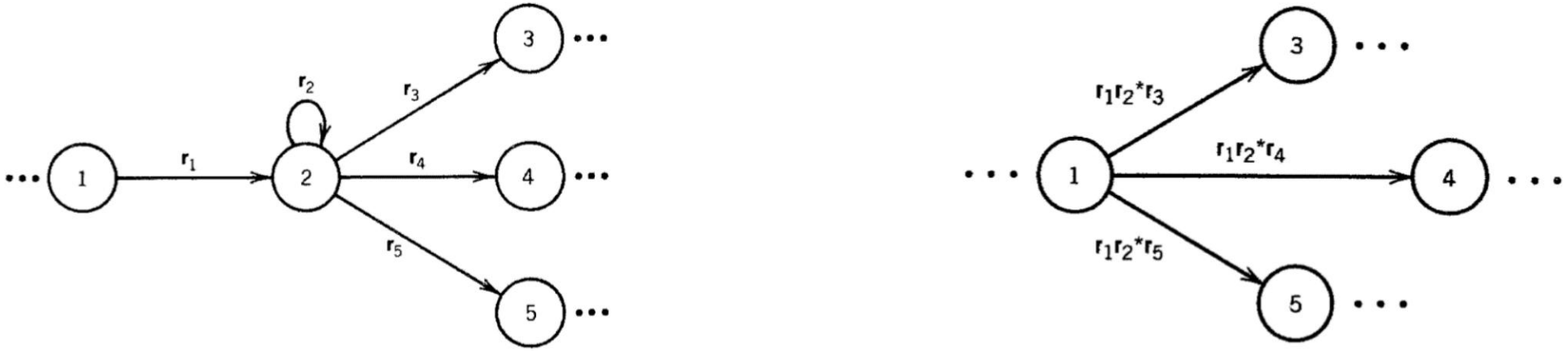
- Baypas iş lemini ancak durum 2'nin kendisine geri dönen bir döngüsü olmadığ ı sürece yapabiliriz.
- Durum 2'nin bir döngüsü varsa, bu modeli kullanmalıyız.





Baypas ve Durum Eleme Operasyonu

- Durum 1, durum 2'ye bağlıysa ve durum 2 bağlıysa birden fazla başka duruma (örneğin 3, 4, 5 durumları)
  - Durum 1'den durum 2'ye giden kenarı ortadan kaldırdığımızda, durum 1'den diğer durumlara (örneğin durum 3, 4, 5) nasıl gidileceğini gösteren kenarlar eklemeliyiz.



Baypas ve Durum Eleme Operasyonu

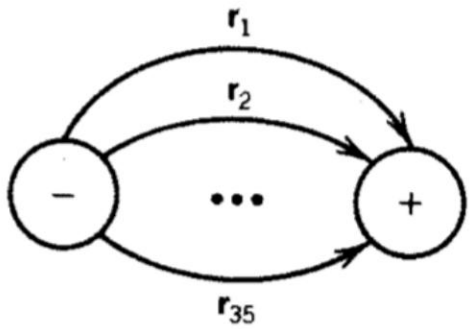
- Durum 2'ye götüren her durum baypas edilebilir
  durum 2
  - Durum 9, durum 2'ye yönlendiriyorsa, durum 9'dan durum 3, 4 ve 5'e doğ rudan kenarlar ekleyerek durum 9'dan durum 2'ye giden kenarı ortadan kaldırabiliriz.
  - Hiçbir ş ey durum 2'ye yol açmayana kadar bu iş lemi tekrarlayabiliriz
  - Bir yolun parçası olmayacağ ı için durum 2'yi tamamen ortadan kaldırabiliriz
    bir dili kabul eden • T'nin

kabul ettiğ i sözcük kümesini değ iş tirmeden, durumlarından birini ortadan kaldırdık

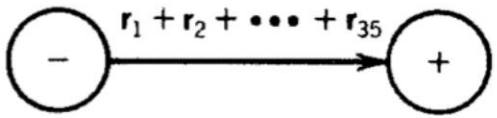
- Bu iş lemi T'deki tüm durumları ortadan kaldırana kadar tekrarlayabiliriz.
  benzersiz baş langıç ve bitiş durumları hariç

Kanıt – Bölüm 2

- Son olarak, buna sahip olacağ ız

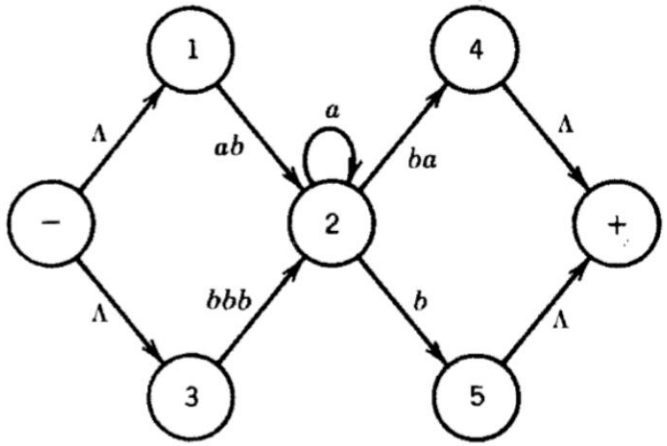


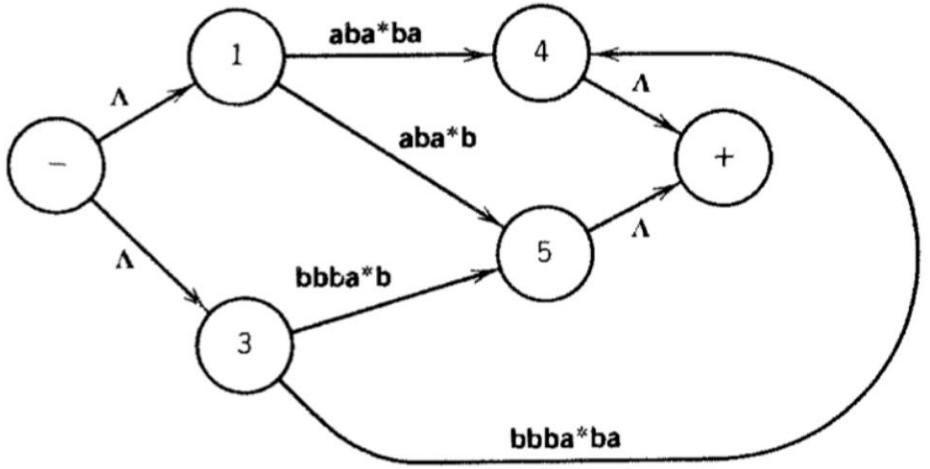
- Daha sonra bunu üretmek için bir kez daha birleş tirebiliriz.



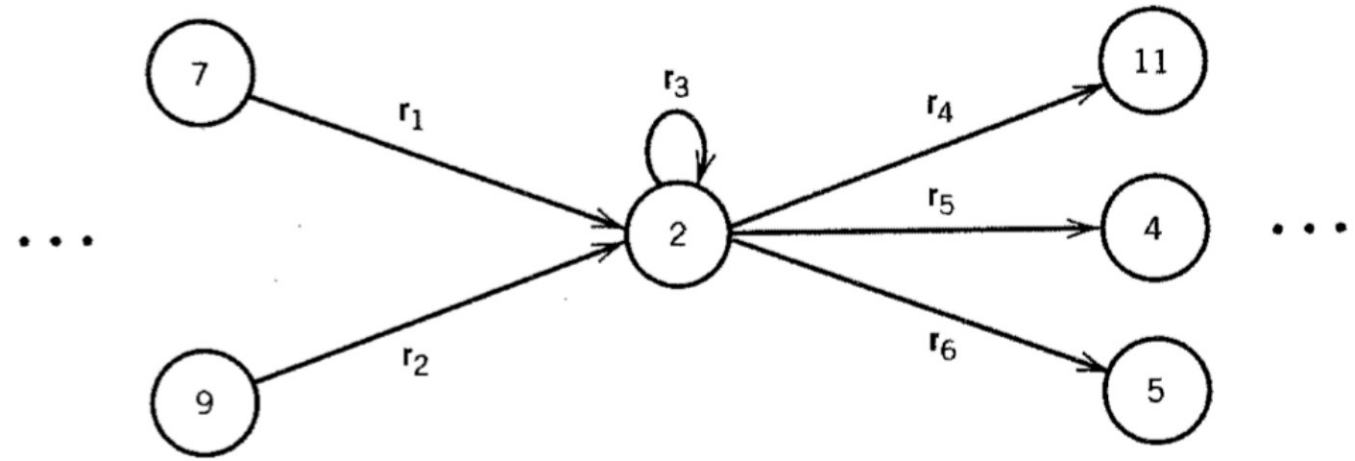
Örnek

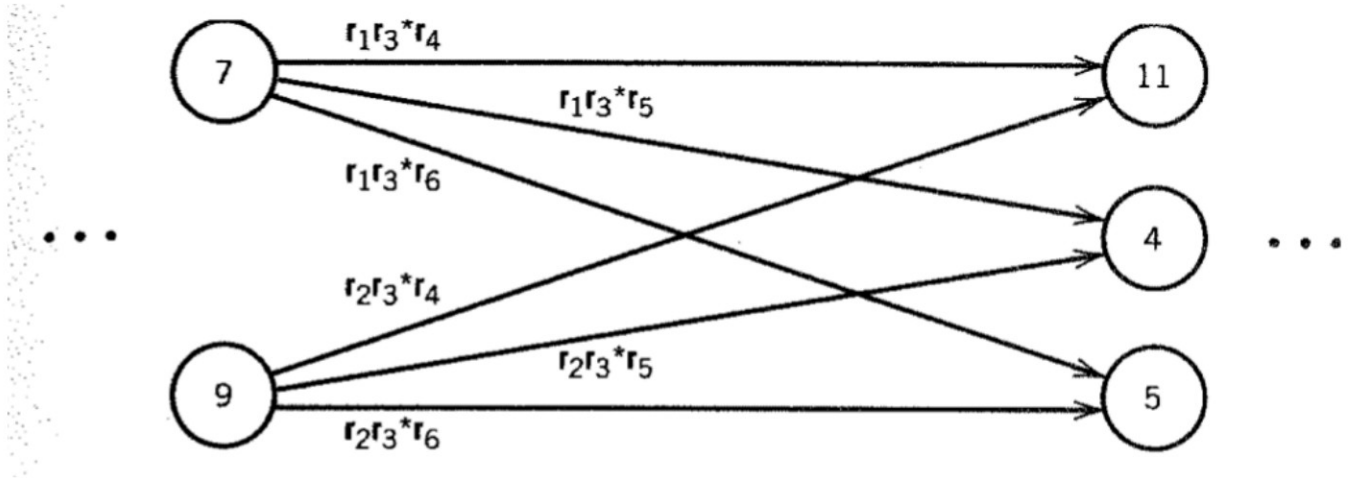
- Durum 2'yi bir yol ekleyerek atlayabiliriz.
  - Durum 1'den durum 4'e aba*ba etiketli
  - Durum 1'den durum 5'e aba*b etiketli
  - Durum 3'ten durum 4'e bba*ba etiketli
  - Durum 3'ten durum 5'e bba*b etiketli





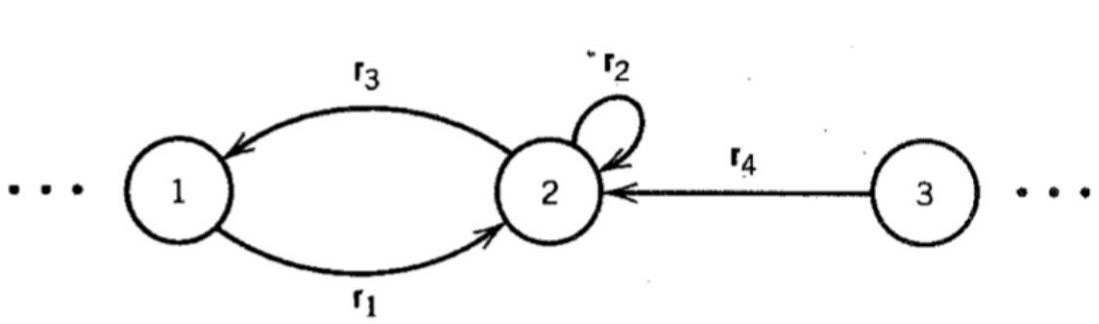
Example

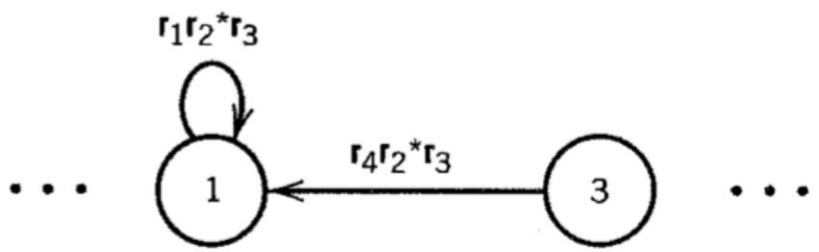




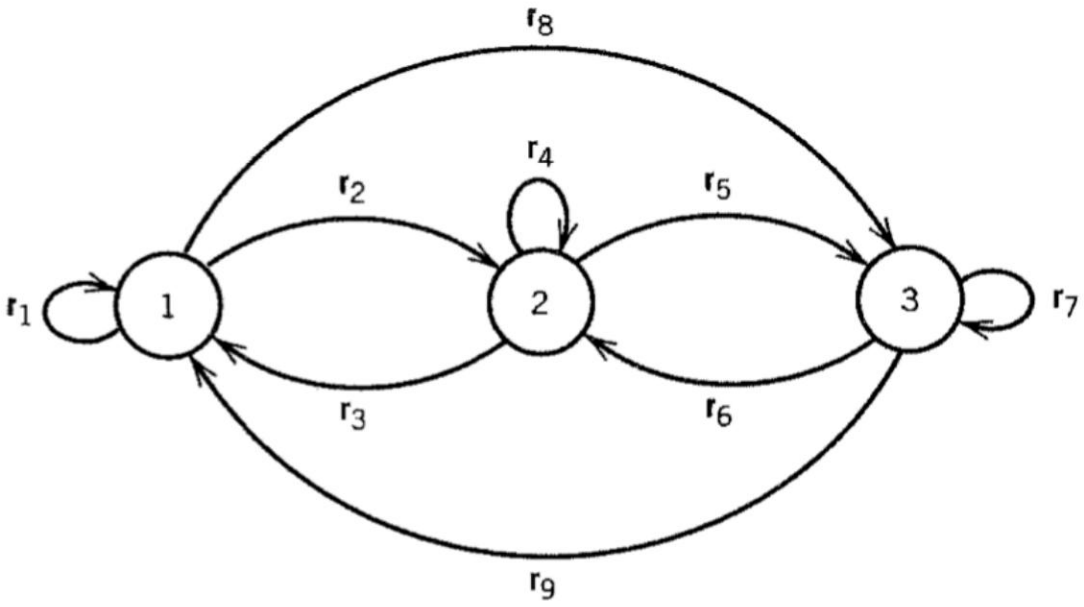
Örnek

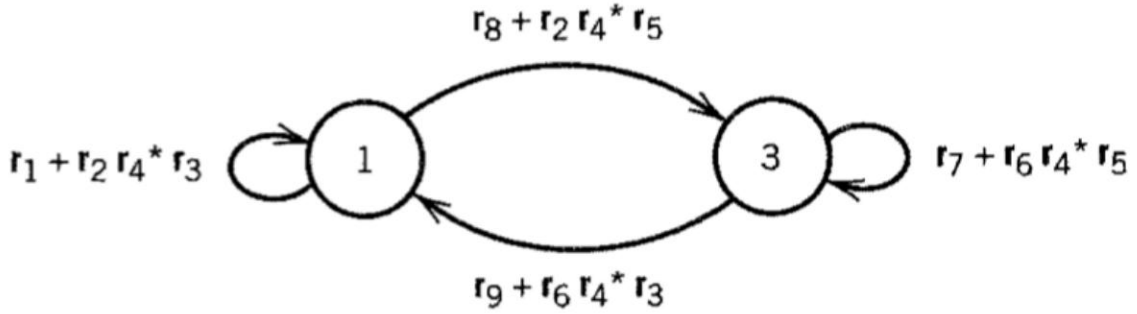
- Daha dikkatli incelememiz gereken özel bir durum
- Durum 2'yi ortadan kaldırmak istiyoruz
  - Muhtemel baypas edilmiş durumun kaynak durumlarından biri, aynı zamanda bu durumdan bir hedef durumdur.





Örnek





- Bir kenarı veya durumu her kaldırdığ ımızda, T üzerinden herhangi bir yolu yok etmediğ imizden veya yeni yollar oluş turmadığ ımızdan emin olmalıyız.

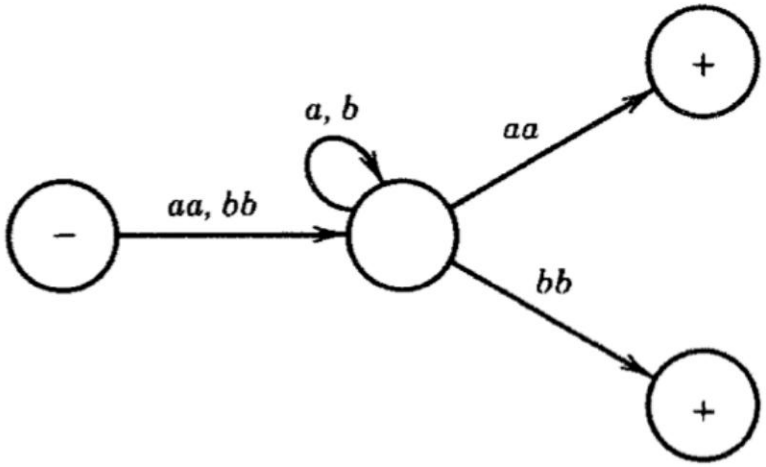
Kanıt – Bölüm 2

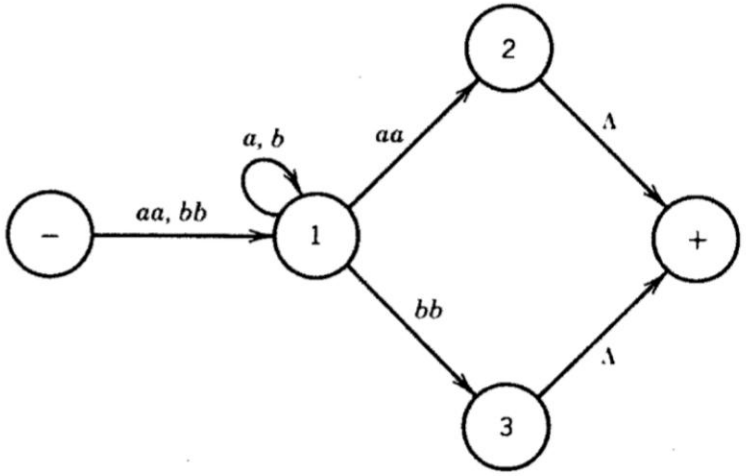
- Bu algoritma sınırlı sayıda adımda sona erer
  - T'nin yalnızca sonlu sayıda durumu vardır ve baypas prosedürünün her yinelemesinde bir durum elenir • Diğer önemli gözlem, yöntemin tüm geçiş grafiklerinde çalıştığıdır.

- Bu nedenle, bu algoritma tatmin edici bir kanıt sağlar.
  her TG için bir RE vardır

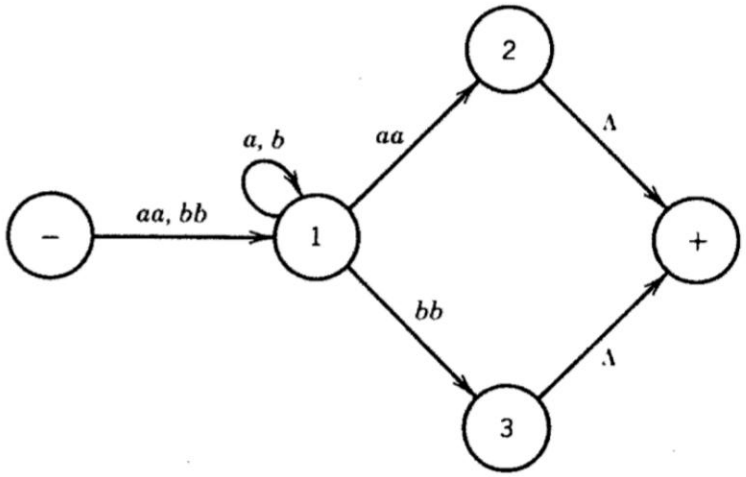
Örnek

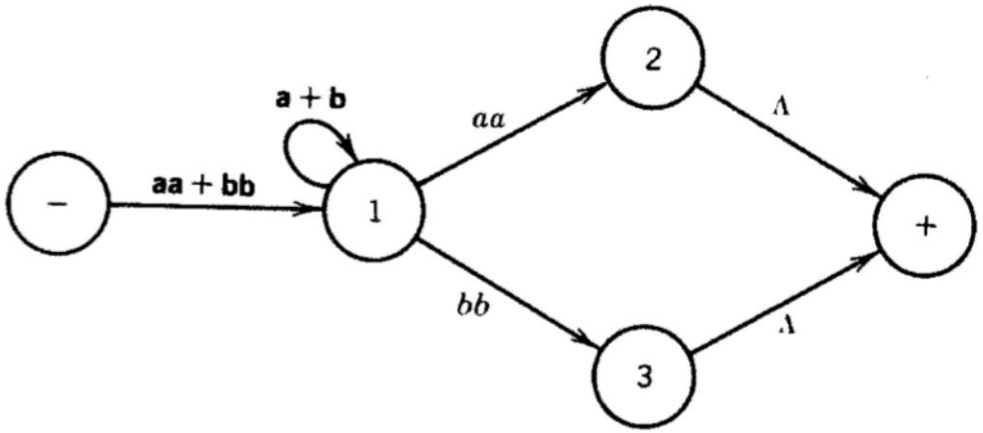
- TG, çift harfle başlayan ve biten tüm kelimeleri kabul eder (en az 4 uzunluğunda)





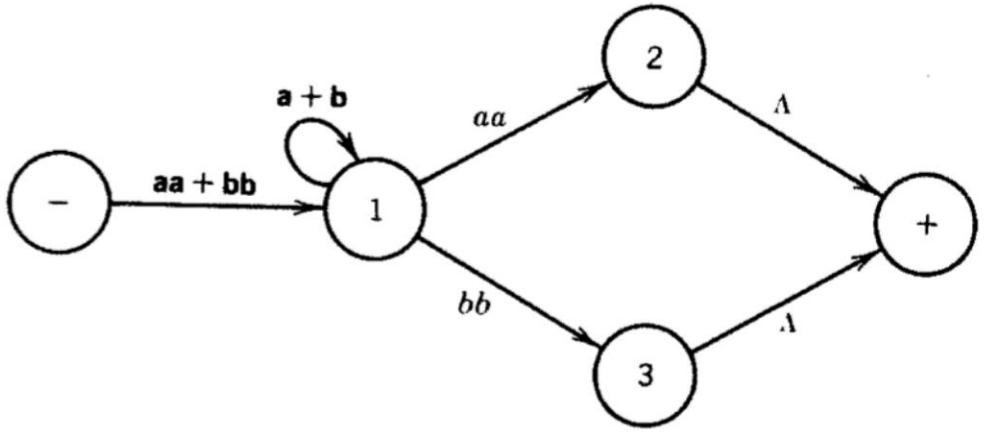
Örnek, devam ediyor

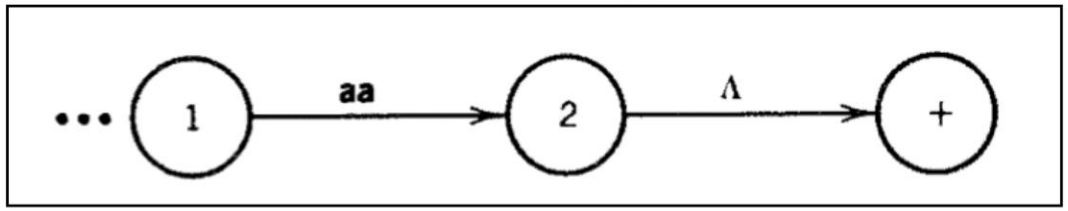


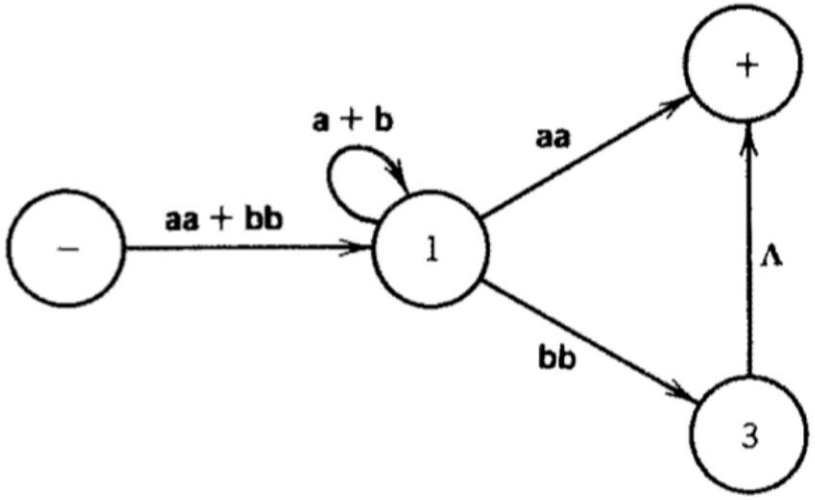


Örnek, devam ediyor

- Algoritma bize bundan sonra TG'nin hangi durumunu atlamamız gerektiğ ini söylemez. Eleme sırası bize kalmış
- Eleme için durum 2'yi seçelim

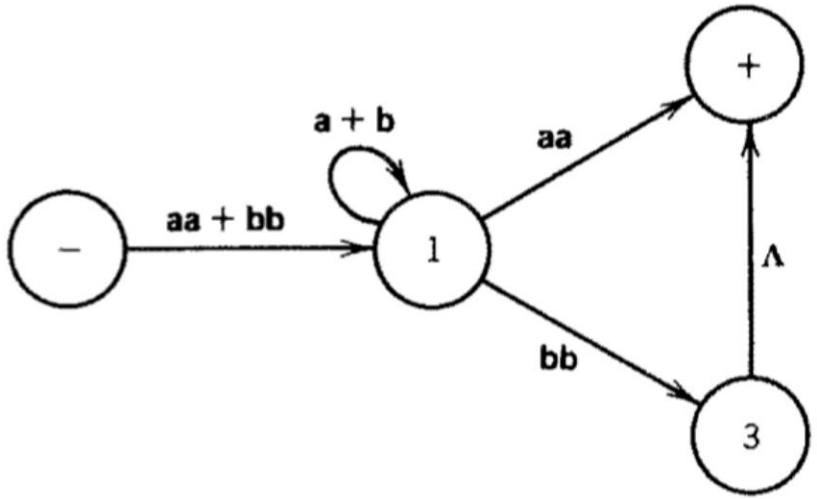


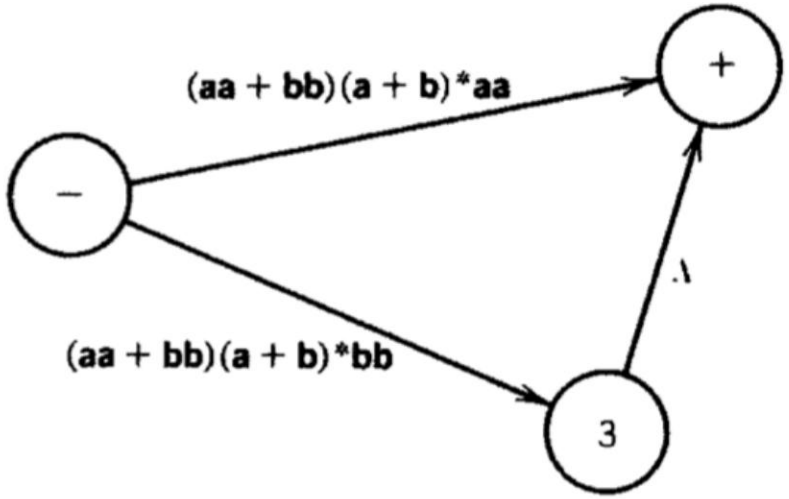




Örnek, devam ediyor

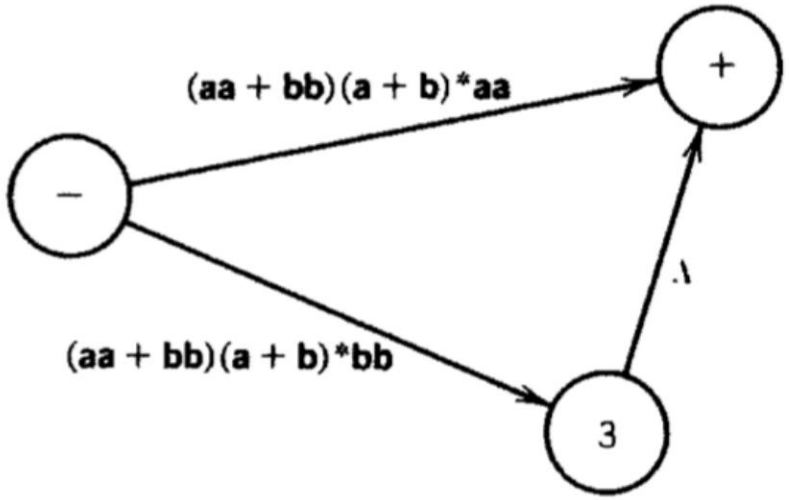
- Sonraki durum 1'i baypas edin (durum 3'ten önce)

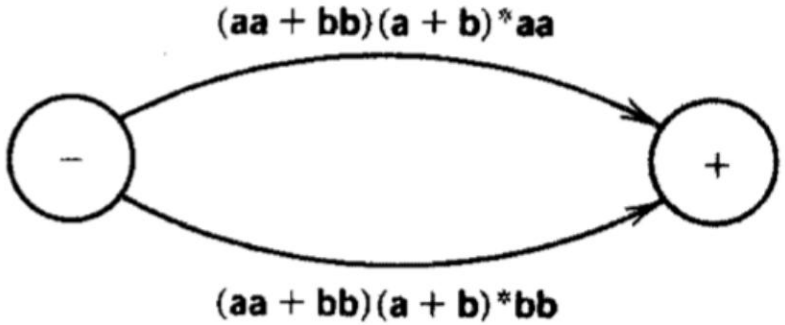




Örnek, devam ediyor

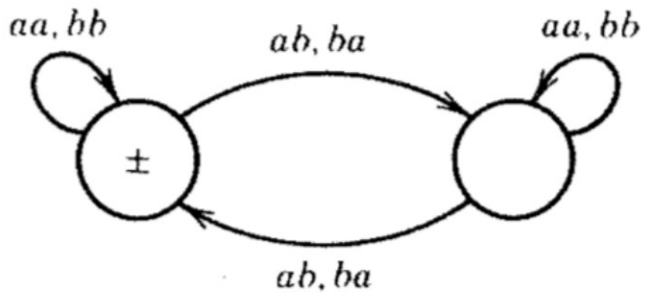
- Şimdi durum 3'ü ortadan kaldırmalıyız (bu kalan tek baypas edilebilir durum)

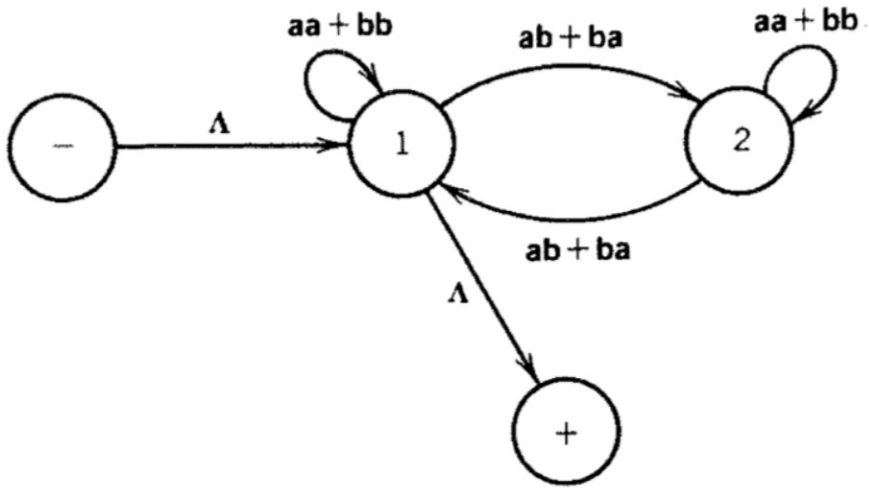


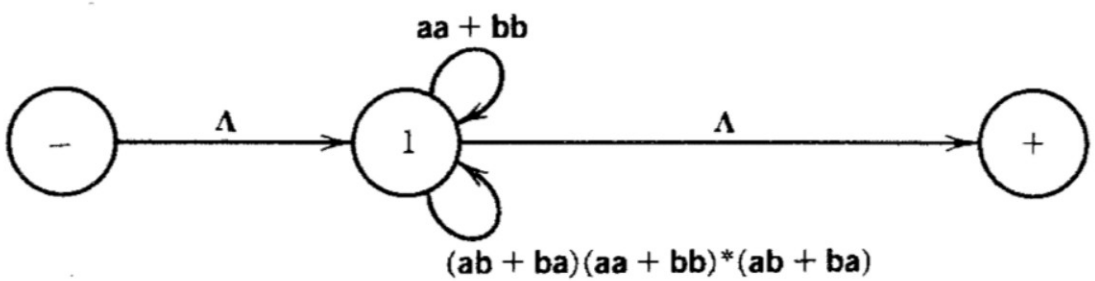


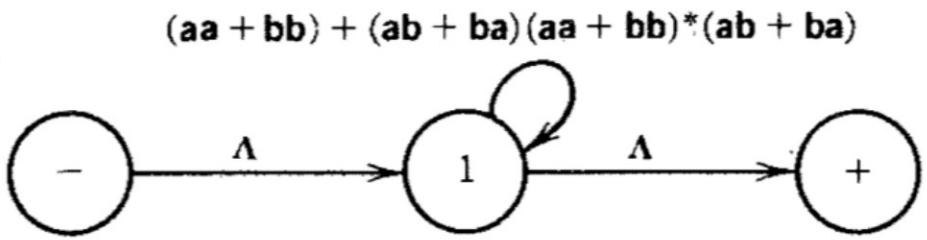
- Bu makine RE ile aynı dili tanımlar • (aa + bb)(a + b)*(aa) + (aa + bb)(a + b)*(bb)

EVEN-EVEN

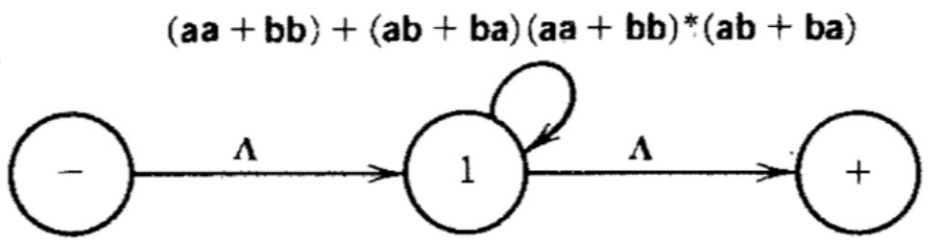


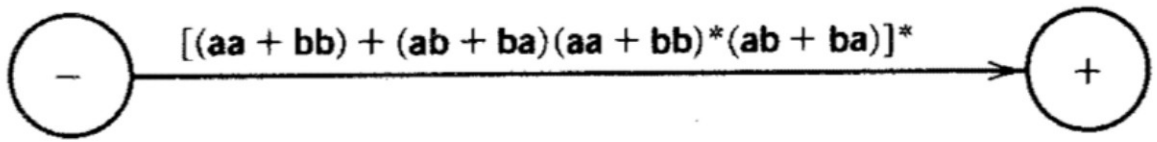






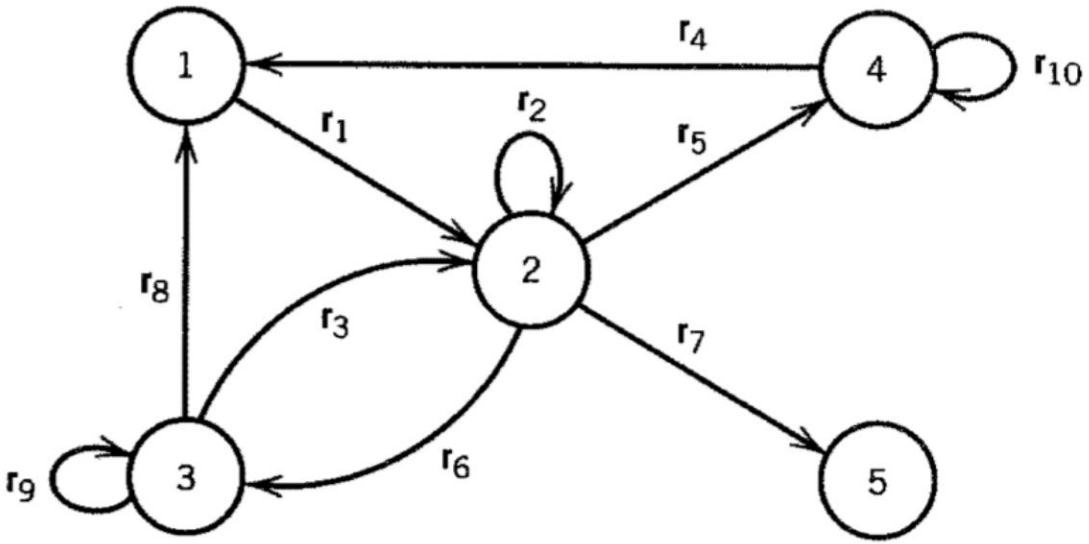
EVEN-EVEN





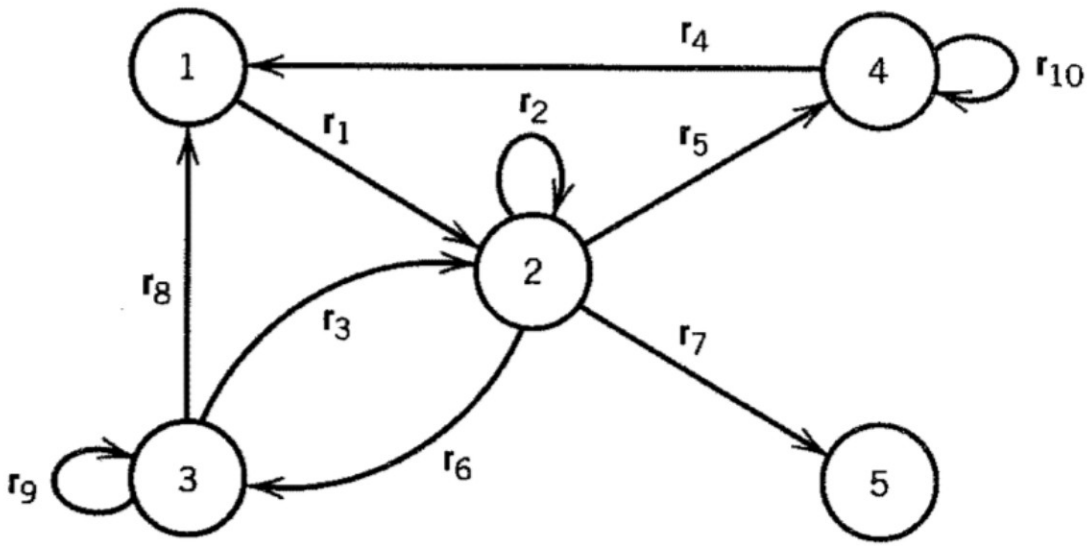
Örnek

- Atlanacak bir sonraki durum, durum 2'dir.
  - Altı yeni kenar eklememiz gerekiyor (3'teki döngü dahil)

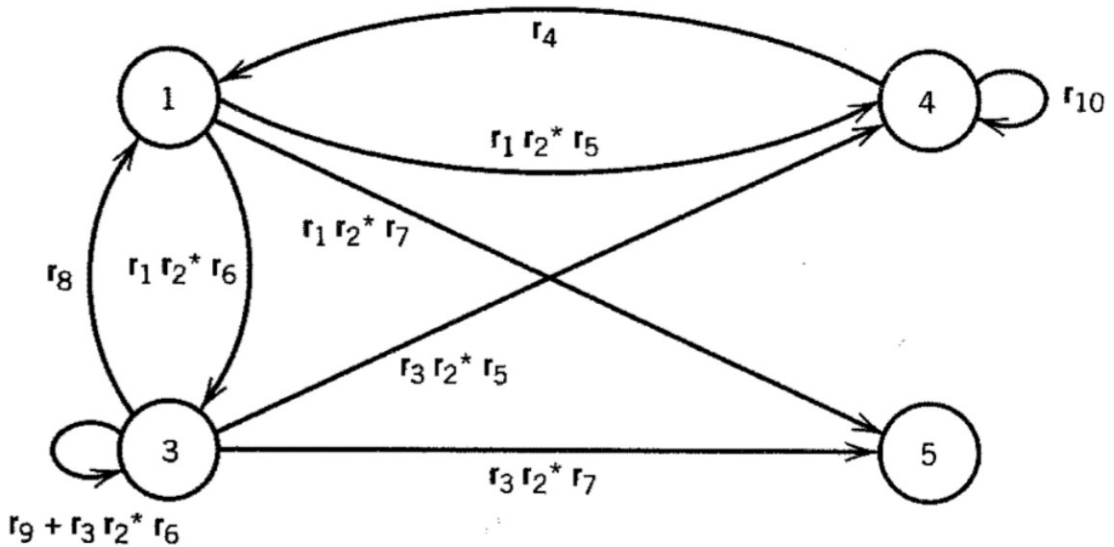


From	To	Labeled
1	3	$r_1 r_2^* r_6$
1	4	$r_1 r_2^* r_5$
1	5	$r_1 r_2^* r_7$
3	3	$r_3 r_2^* r_6$
3	4	$r_3 r_2^* r_5$
3	5	$r_3 r_2^* r_7$

Example



From	To	Labeled
1	3	$r_1 r_2^* r_6$
1	4	$r_1 r_2^* r_5$
1	5	$r_1 r_2^* r_7$
3	3	$r_3 r_2^* r_6$
3	4	$r_3 r_2^* r_5$
3	5	$r_3 r_2^* r_7$



algoritma

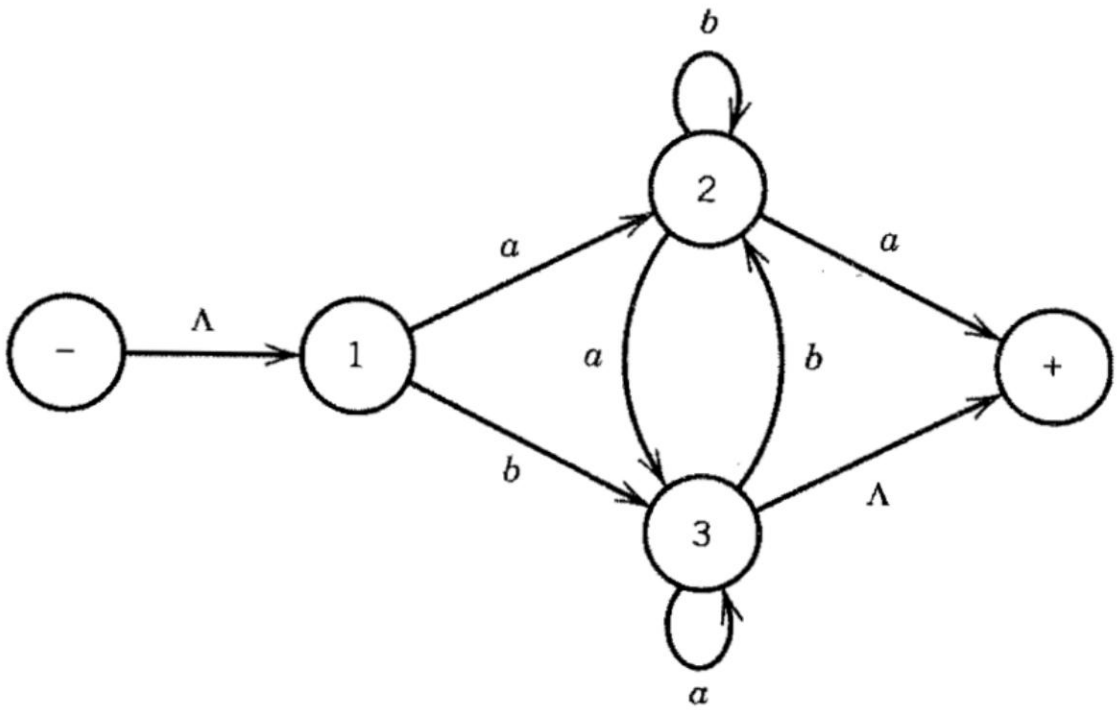
1. Benzersiz, girilemez bir eksi durumu ve benzersiz, bırakılamaz bir durum oluş turun artı durum

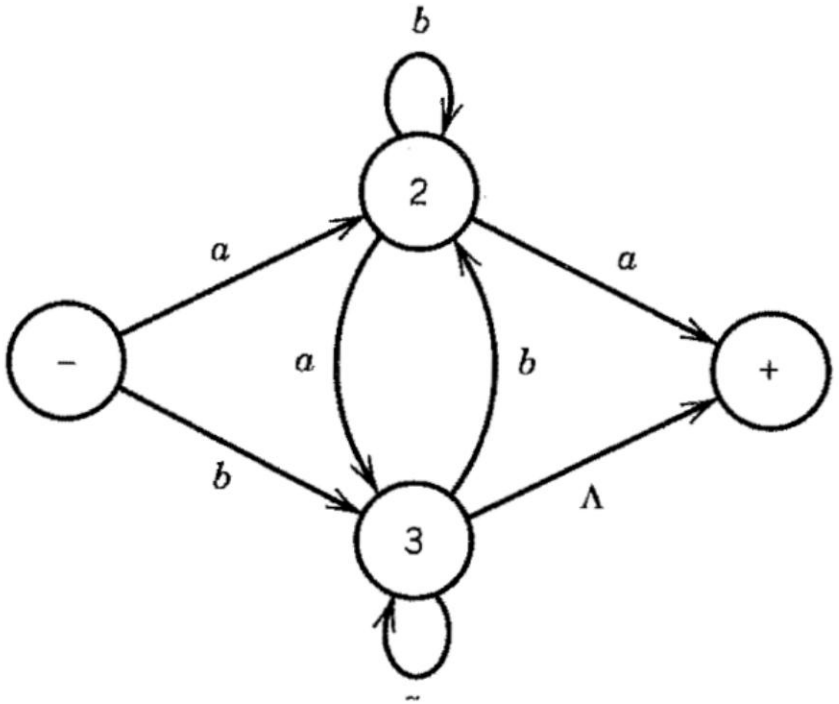
2. Herhangi bir sırayla, TG'deki - veya + olmayan tüm durumları baypas edin ve ortadan kaldırın. Gelen her kenar her giden kenara bağ lanarak bir durum atlanır. Ortaya çıkan her bir kenarın etiketi, gelen kenardaki etiketin, varsa döngü kenarındaki etiketle ve varsa giden kenardaki etiketin birleş tirilmesidir 3. İ ki durum, aynı yöne giden birden fazla kenarla birleş tirildiğ inde yönde, etiketlerini ekleyerek bunları birleş tirin 4. -'den +'ya bir kenar kaldığ ında, o kenardaki etiket bir

Orijinal makine tarafından tanınanla aynı dili üreten RE

Örnek

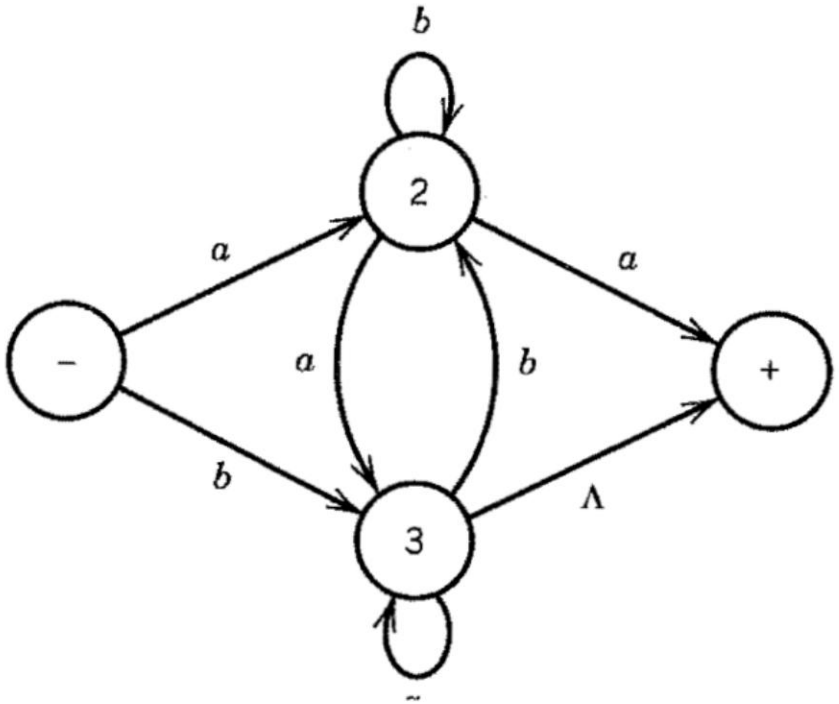
- 1, 2, 3 sıralamasındaki durumları ortadan kaldırın

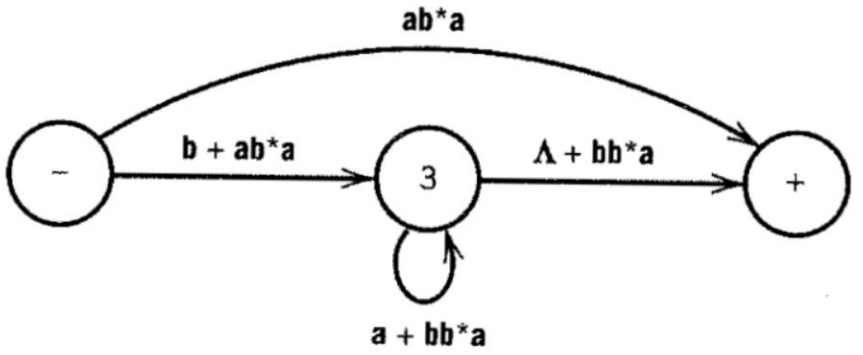


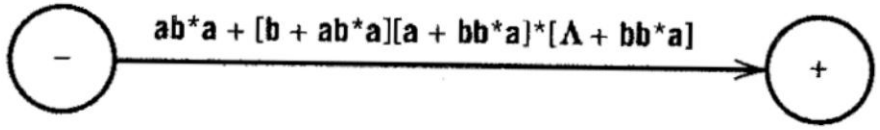


Örnek

- 1, 2, 3 sıralamasındaki durumları ortadan kaldırın

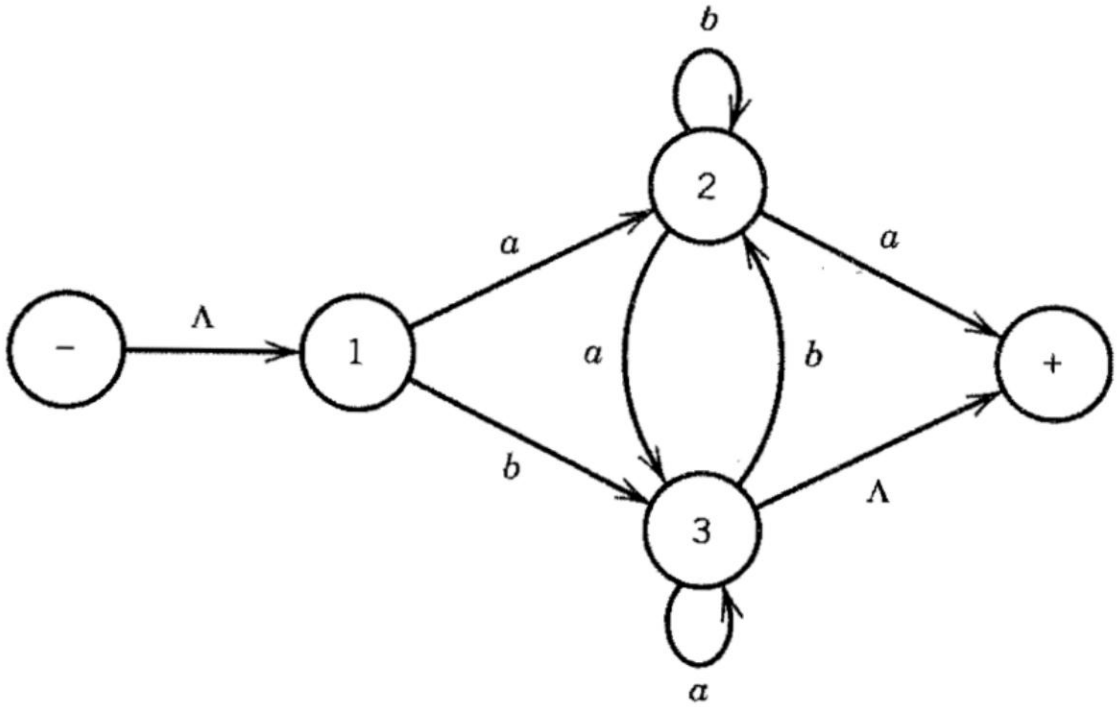


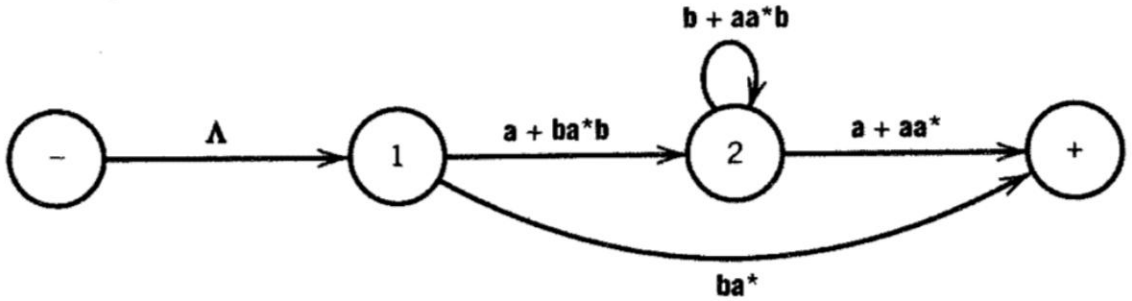




Örnek

- 3, 2, 1 sırasına göre durumları ortadan kaldırın





Örnek

- 3, 2, 1 sırasına göre durumları ortadan kaldırın