BİÇİMSEL DİLLER VE OTOMATLAR

Hazırlayanlar:

Prof.Dr. Emre HARMANCI
Yard.Doç.Dr. Osman Kaan EROL

İçindekiler:

1. **Sonlu Durumlu Makinalar**
   1.1. Tanım ve modeller (Mealy ve Moore Modelleri)
   1.2. Algoritmik Durum Modeli ile ardışıl sistem tasarımı
   1.3. Durum eşdeğerliliği, durum uyuşması ve durum indirgemesi
2. **Biçimsel Dillerin Matematiksel Temelleri**
   2.1. Kümeleri tümevarım ile tanımlama
   2.2. Alfabe ve diller
   2.3. Bağıntılar ve kapanış bağıntıları
   2.4. Diller ve gramerler
   2.5. Dilbilgisi, Chomsky Sınıflandırması
   2.6. Düzenli ifadeler
3. **Otomatlar**
   3.1. Determinist Sonlu Otomat (DFA) ve düzenli ifadelerin tanınması
   3.2. Determinist Olmayan Otomat (NFA) ve düzenli ifadelerin tanınması
   3.3. DFA ile NFA eşdeğerliği
4. **Yığın Yapılı Otomat (PDA) ve bağlamdan bağımsız dillerin tanınması**
5. **Turing Makinası ve hesaplama kuramlarına giriş**

Program:

Hafta	**Konu**
1	1.1, 1.2
2	1.2, 1.3 + ödev
3	1.3, 2.1 + uygulama
4	2.1, 2.2 + ödev
5	2.3, 2.4 + uygulama
6	**1.Vize**
7	2.5, 2.6
8	3.1 + ödev
9	3.2 + uygulama
10	3.3 + ödev
11	4 + uygulama
12	**2.Vize**
13	4, 5 + ödev
14	5 + uygulama
	Final

Kaynakça:

1. M. Mano, *Digital Design*, Prentice-Hall, 1984
2. D.F. Stanat, D.F. McAllister, *Discrete Mathematics in Computer Science*, Prentice-Hall, 1977
3. H.R. Lewis, C.H.Papadimitriou, *Elements of the Theory of Computation*, Prentice-Hall, 1981
4. J.C. Martin, *Introduction to Languages and the Theory of Computation – Second Edition*, Mc Graw Hill, 1997
5. P.J. Denning, J.B.Dennis, J.E. Qualitz, *Machines, Languages, and Computation*, Prentice Hall 1978
6. J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages and Computation – Second Edition*, Addison-Wesley, 2001
7. P. Dehornoy, *Mathematiques de l'Informatique, Cours et Exercises Corrigés*, Dunod, 2000

BÖLÜM 1

Sonlu Durumlu Makinalar (Finite State Machine - FSM)

1.1 Tanım ve modeller (Mealy ve Moore Modelleri)

Sonlu durumlu makinalar;

G: Giriş büyüklükleri alfabesi ve G ∈ ***G***
D: Durum büyüklükleri alfabesi ve D ∈ ***D***
Ç: Çıkış büyüklükleri alfabesi ve Ç ∈ ***Ç***
U: ***G*** x ***D*** -> ***D*** tanımlı gelecek durum fonksiyonu
V: ***D*** ->***Ç*** veya ***G*** x ***D*** -> ***Ç*** tanımlı çıkış fonksiyonunu göstersin.

Örnek:

$\boldsymbol{G} = B^n_2 = \{0, 1\}^n$

G_0 = 0100111 olabilir (bu durumda n = 7' dir)

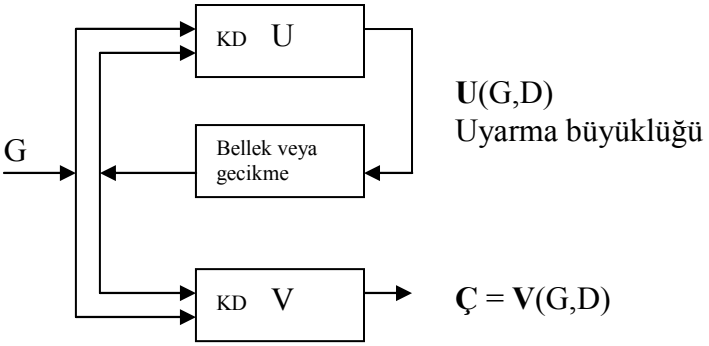
G kümesinin eleman sayısı $|\boldsymbol{G}| = 2^n$'dir. | . | yerine Card(.) tanımı da kullanılır.

FSM = {***G, D, Ç, U, V***} şeklinde tanımlanan makinalara sonlu durum makinası adı verilir. İki farklı ana yapıları bulunur:

A. Mealy Modeli:

$\boldsymbol{G}\, X\, \boldsymbol{D} \rightarrow \boldsymbol{Ç}$

Çıkış, giriş ve durumların bir fonksiyonudur. (KD = kombinezonsal devre)

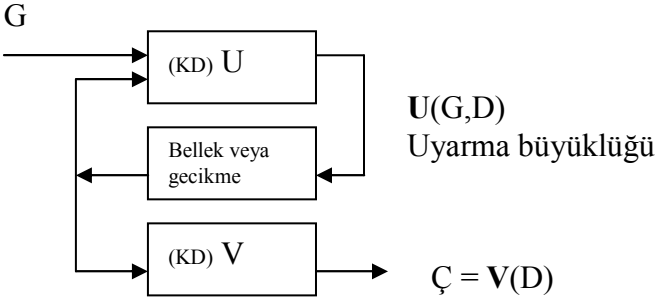


Şekil 1.1 Mealy modeli

B. Moore Modeli:

$\boldsymbol{D} \rightarrow \boldsymbol{Ç}$

Durumlar sistemin çıkışlarıdır. (KD = kombinezonsal devre)



Şekil 1.2 Moore modeli

Sonlu durumlu makina bir sayısal devre ile gerçekleştirilirse, giriş ve çıkış fonksiyonları her iki modelde de kombinezonsal devrelerle yerine getirilir. Ardışıl senkron devrelerde yeni durumların belirlenmesinde durum saklama (bellek) elemanı olarak kullanılan ikililerin (Flip-flop, FF) tanım bağıntıları etkin olur. Asenkron devrelerde ise bellek elemanı saf gecikmedir. Senkron devrelerde, Q kullanılan FF'lerin tanım bağıntılar vektörü olmak üzere:

$\mathbf{D}(t^+) = \mathbf{Q}(\mathbf{D}(t), \mathbf{U}(\mathbf{D}(t), \mathbf{G}(t)))$ şeklinde verilebilir. Eğer FF'ler D tipi iseler ya da saf bellek özelliği gösteriyorlarsa $\mathbf{D}(t^+) = \mathbf{U}(\mathbf{D}(t), \mathbf{G}(t))$ elde edilir. D tipi FF yerine saf gecikme kullanılırsa asenkron devre için de senkron devre tanımları kullanılabilir. Q fonksiyonu kullanılan FF tipine bağlı olarak aşağıdaki formlardan biri olabilir:

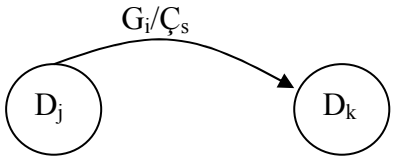
$Q^+ = JQ' + K'Q$, JK tipi FF bağıntısı. J ve K giriş büyüklükleridir.
$Q^+ = T \oplus Q$, T tipi FF bağıntısı. T giriş büyüklüğüdür.
$Q^+ = D_g$, D tipi FF bağıntısı. D_g giriş büyüklüğüdür.
$Q^+ = S + R'Q - SR$ tipi FF bağıntısı. S ve R giriş büyüklükleridir.

Sonlu durumlu makinaların aşağıda belirtilen özelliklere sahip oldukları kabul edilir:

1. Giriş, çıkış ve durum kümeleri (alfabeleri) sonludur;
2. Gerekircidir (ing: Determinist). Şimdiki giriş ve durum, sonraki durumu ve çıkışı kesinlikle belirler. Sisteme makina denmesini sağlayan özellik budur. Makine kavramı gerekirciliği içerir. Gerekirci olmayan sistemler makine olamaz. Gerekirci olmayan sistemlerde belirli bir giriş - durum çiftine birden fazla sayıda çıkış ve sonraki durum değeri karşı düşebilir. Bu tip sistemler Determinist Olmayan Otomat bölümünde detaylı olarak işlenecektir.
3. Büyüklük dönüştürücü veya "transducer" özelliği. Bu özellik sayısal ayrık zamanlı sistemler (digital discrete-time systems) için aşağıdaki gibi tanımlanır:
   - Sekans: Sıralı simge dizisi (Bu bağlamda sıra zamanı belirler). *n* sekans uzunluğudur.
   - [Λ]: boş katarı içine alan "singleton" yani tek elemanlı küme.
   - G*: giriş sekansı kümesi $G^* = [\Lambda] \cup G \cup G^2 \cup ... \cup G^n \cup ...$
   - Ç*: çıkış sekans kümesi $Ç^* = [\Lambda] \cup Ç \cup Ç^2 \cup ... \cup Ç^n \cup ...$
   - $x \in G^*$ ve $w \in Ç^*$ olmak üzere:
   Belirli bir başlangıç durumunda, belirli bir *x* girişi altında belirli bir *w* çıkışı elde edilecektir. $w = f(x)$ dönüşümünün iki özelliği bulunmaktadır:
   - **Uzunluğun korunması** (length preservation - zamanda sıkıştırmanın olmaması), $|x|$, *x* dizisinin simge sayısı, uzunluğu olmak üzere $|w| = |x| = n; \forall n \in N$
   - **Önek içindeliği** (prefix inclusivity)
   $x = x_1x_2 \wedge f(x) = w = w_1w_2 \wedge |x_1| = |w_1| \Rightarrow f(x_1) = w_1$

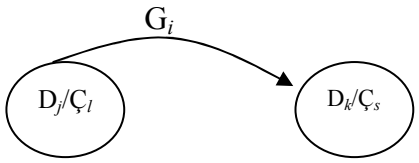
FSM modellemede aşağıda verilen durum tablo ve diyagramları kullanılır:

	G_1	..	G_i	..	G_m
D_1					
..					
D_j			$D_k/Ç_s$		
..					
D_n					



Şekil 1.3 Mealy modeli durum tablo ve diyagramı

	G_1	..	G_i	..	G_m	Ç
D_1						$Ç_1$
..						..
D_j			D_k			$Ç_l$
..						..
D_n						$Ç_p$



Şekil 1.4 Moore modeli durum tablo ve diyagramı

Örnek:

5,10, 25 birimlik metal paraları kabul eden bir makina, 15 birime bir sakız veriyor ve paranın üstünü iade ediyor. Bu makinaya ait Mealy ve Moore modellerini çıkarınız.

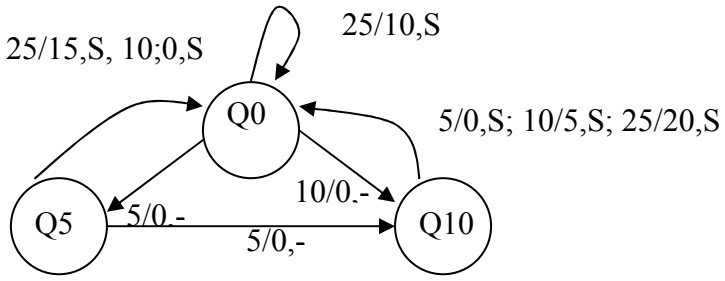
A. Mealy Modeli

A.1. Durum tablosu:

	5	*10*	*25*
Q_0	Q_5/0,-	Q_{10}/0,-	Q_0/10,S
Q_5	Q_{10}/0,-	Q_0/0,S	Q_0/15,S
Q_{10}	Q_0/0,S	Q_0/5,S	Q_0/20,S

$Q_x/i,j$: *x* para durumu, *i* para üstü, *j* ise sakız verince S, vermeyince "-" olan çıkış.

A.2. Durum Diyagramı



B. Moore Modeli

B.1. Durum Tablosu

	5	*10*	*25*	*Çıkış*
Q_0	Q_5	Q_{10}	Q_{25}	0,-
Q_5	Q_{10}	Q_{15}	Q_{30}	0,-
Q_{10}	Q_{15}	Q_{20}	Q_{35}	0,-
Q_{15}	Q_5	Q_{10}	Q_{25}	0,S
Q_{20}	Q_5	Q_{10}	Q_{25}	5,S
Q_{25}	Q_5	Q_{10}	Q_{25}	10,S
Q_{30}	Q_5	Q_{10}	Q_{25}	15,S
Q_{35}	Q_5	Q_{10}	Q_{25}	20,S

Not: Moore modelindeki durum sayısı, en az Mealy modelindeki farklı durum/çıkış sayısı kadar olmalıdır. Durumlar atılan para ile ilişkilendirilmemelidir. Para atılmadığında mevcut durum korunur. Buna göre:

Q_0 başlangıç durumu olmak üzere (istenirse kaldırılabilecek bir durumdur, bir daha bu duruma girilmez),

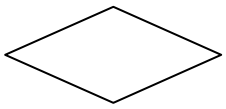
$Q_5/0,- \rightarrow Q_5$
$Q_{10}/0,- \rightarrow Q_{10}$
$Q_0/0,S \rightarrow Q_{15}$
$Q_0/5,S \rightarrow Q_{20}$
$Q_0/10,S \rightarrow Q_{25}$
$Q_0/15,S \rightarrow Q_{30}$
$Q_0/20,S \rightarrow Q_{35}$

B.2. Durum diyagramını siz oluşturunuz.

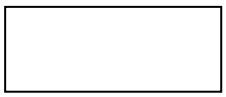
1.2 Algoritmik Durum Makinası modeli ile ardışıl sistem tasarımı

Akış diyagramı ve durum diyagramı ile karşılaştırma yapıldığında her iki diyagram arasında benzerliklerin olduğu görülecektir. Akış diyagramında karar aşamaları (geçişi sağlayan koşullar) açık olarak gösterilirken, durum diyagramında kol değerleri olarak ifade edilir. Ayrıca akış diyagramlarındaki durum ve koşullara bağlı işlemler durum diyagramlarının çıkışları ile ilişkilendirilebilir. Ancak burada Mealy ve Moore modellerine göre melez bir kullanım oluşur. İşlev olarak birbirinin aynısıdır. Akış diyagramlarında kullanılan simgeler aşağıdaki gibidir:

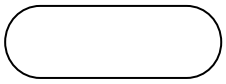
1. Karar Baklavaları



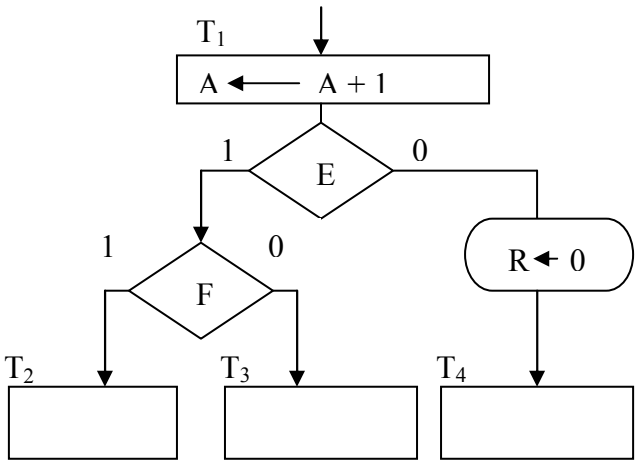
2. Durum ve duruma bağlı işlemler, dikdörtgenler



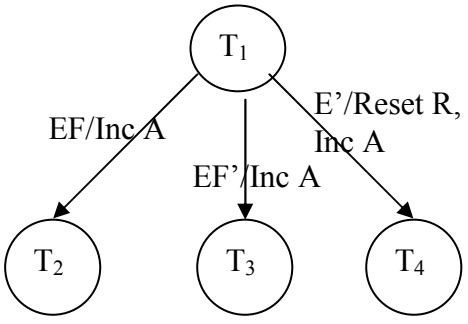
3. Koşullara bağlı işlem, ovaller



<u>Akış diyagramı ile ilgili örnek:</u>

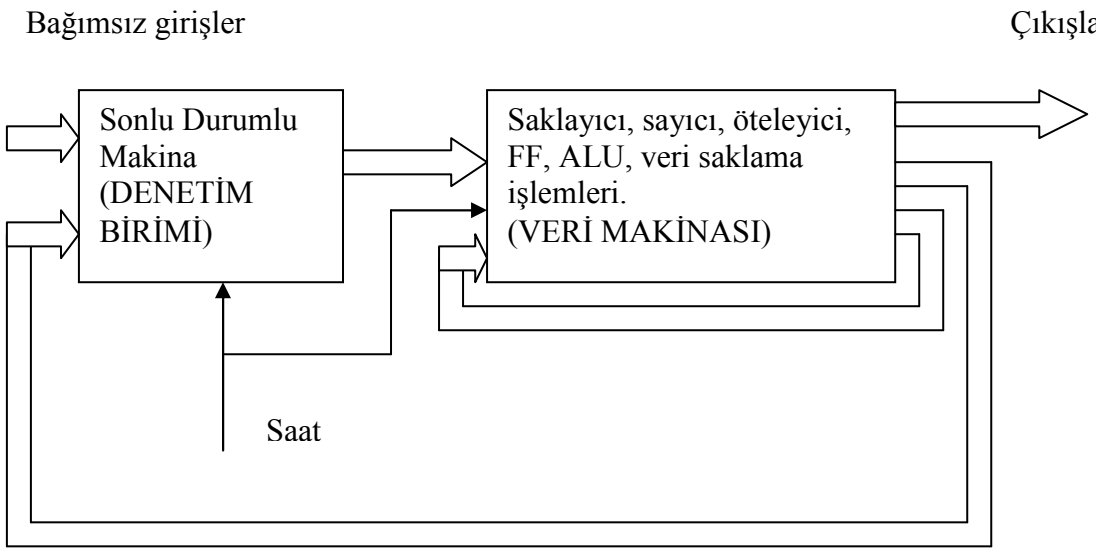


Şekil 1.5 Akış diyagramı



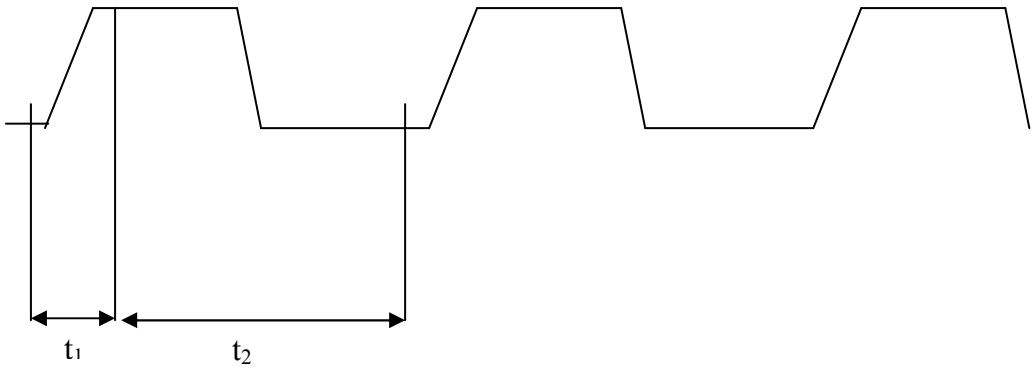
Şekil 1.6 Durum diyagramı

Algoritmik Durum Makinası yönteminin sayısal sistemlere uygulanması:



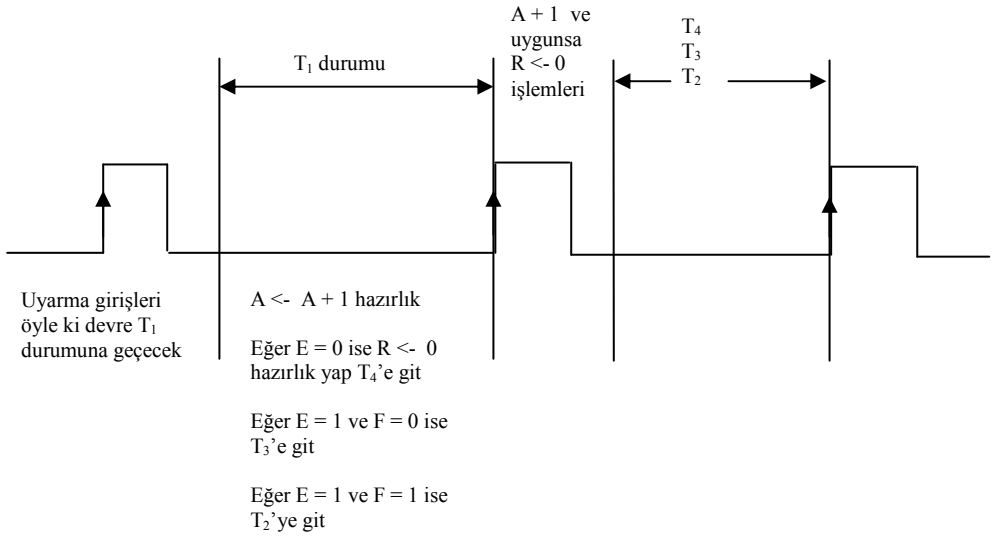
Şekil 1.7 Devre modeli

Şekil 1.7 ile senkron bir ardışıl devre yapısı verilmiştir. Saatin etkin kenarında ya da kayıt süresince giriş, durum ve çıkışlar sabit ve tanımlı olmalıdır. Aşağıdaki şekilde yükselen etkin kenarlı saat işareti uygulanmış bir sistemde, t_1 süresince girişlerin (sistem ve FF girişleri) değişmeden sabit kalması gereklidir. Kalan t_2 zamanı içerisinde işaretler değişebilir. Zaman tanım bölgesinde t_1 ve t_2 sürelerinin gösterimi aşağıdaki gibidir:



Şekil 1.8.A Zamanlama çizelgesi

Buna göre Şekil 1.5'te akış diagramı verilmiş makinaya ilişkin işlemlerin zaman içinde yürütülme sırasını gösterir zamanlama diagramı aşağıda Şekil 1.8.B'de verilmiştir:

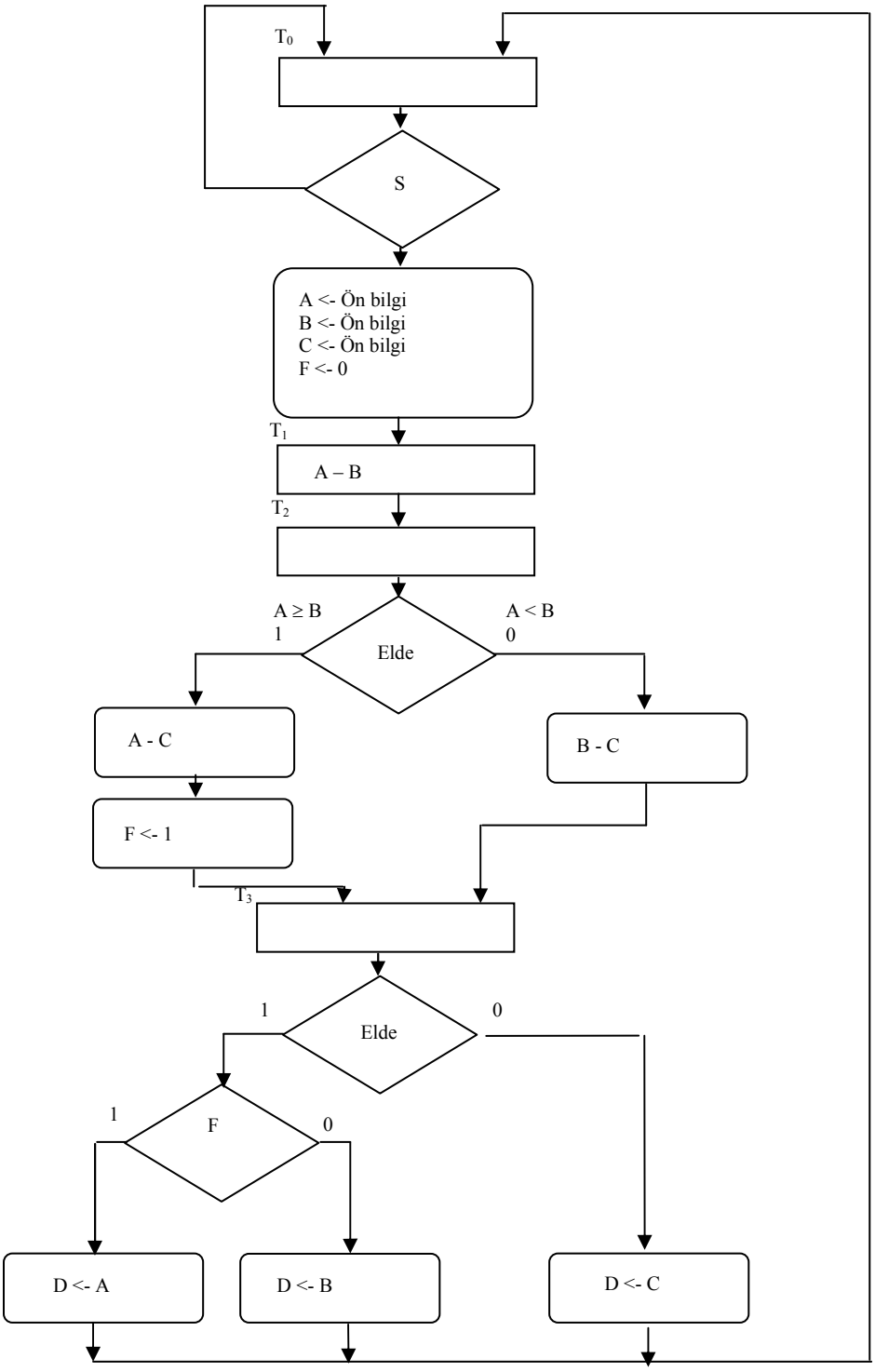


Şekil 1.8.B. Şekil 1.5'te verilen makinanın zamanlama grafiği

Örnek:

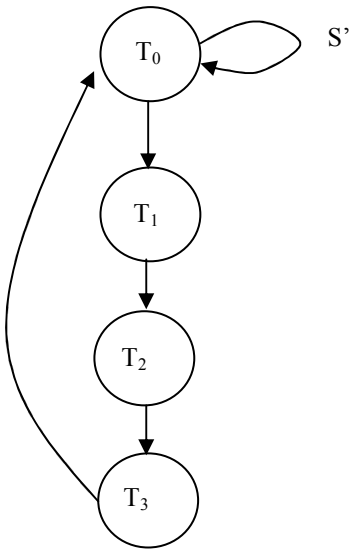
3 adet pozitif sayının en büyüğünü bulan bir algoritmik durum makinası (ASM) tasarlanacaktır. Makine bir giriş işaretinin "1" olması ile çalışmaya başladıktan sonra sekiz bitlik A, B, C saklayıcılarına ilgili sayılar yüklenecektir. Makine çalışmasını tamamlayıp başlangıç durumuna döndüğünde "D" saklayıcısında en büyük sayı yer alacaktır (D = Max(A, B, C)). Bu devrede karşılaştırma işlemlerini yapmak için bir adet kombinezonsal çıkartma devresi kullanılacaktır. Çıkartma sonucu ayrı bir "elde" bayrağında tutulacaktır.

a) Yukarıda tanımlanan devrenin ASM diagramını çiziniz.
b) Veri makinesini tasarlayarak çiziniz. Kullandığınız elemanların giriş işaretlerini belirtiniz.
c) Denetim birimini her duruma bir D FF karşı düşürerek tasarlayıp çiziniz.



Şekil 1.9 Soruda verilen makinanın akış diagramı

Bu makinanın durum diagramı ise aşağıdaki gibidir:

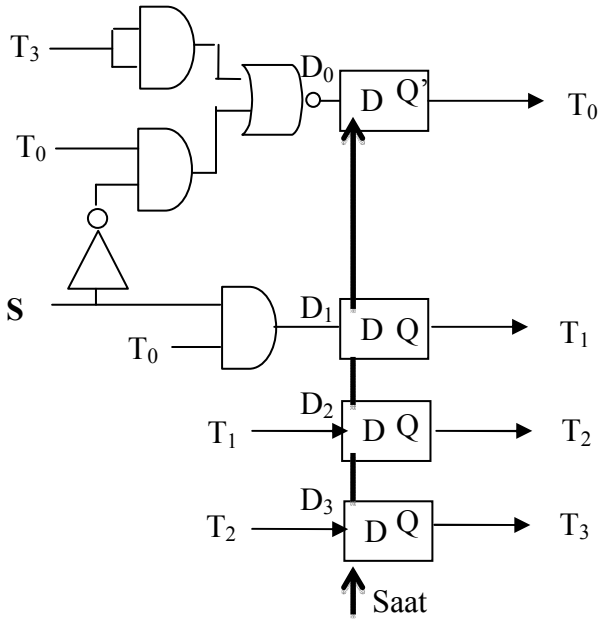


Durum makinasına ilişkin denklemler:

$T_0 = T_3 + s'T_0$
$T_1 = sT_0$
$T_2 = T_1$
$T_3 = T_2$

Şekil 1.10 Durum diagramı

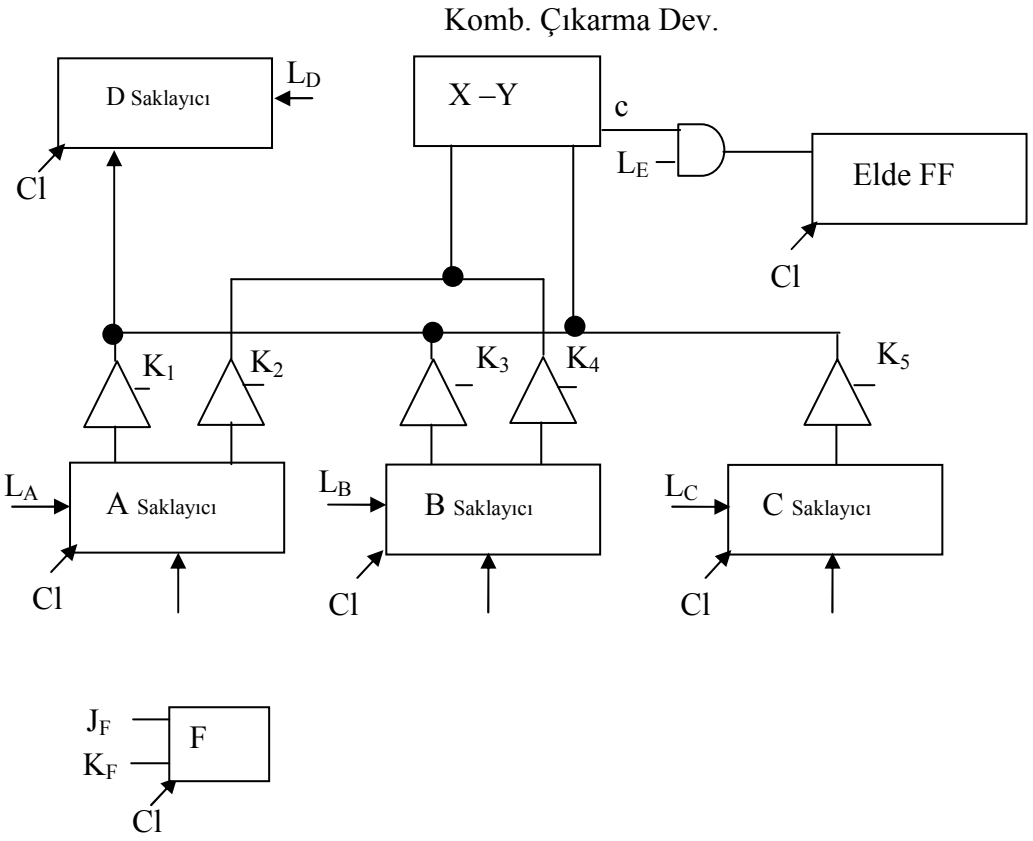
Durum makinasının elektrik devre şeması ise verilen denklemlerden hareketle aşağıdaki gibi bulunur. Burada dikkat edilmesi gereken husus T_0 durumuna ait çıkışın D-FF Q' çıkışından alınmasının gerekliliğidir. Sistem resetlendiğinde bütün FF'ların çıkışları normalde "0" olur. Sistemimizin reset durumu T_0 olduğundan, bu durumun çıkışı reset anında "1" olmalıdır. Q' çıkışı bize bu şartı sağlar. Ancak D-FF girişi de tümlenmelidir.



Bütün FF'lara giden saat ve reset işaretleri şekilde gösterilmemiştir.

Şekil 1.11 Durum makinasının elektrik devre şeması

Saklayıcı, öteleyici ve diğer veri saklama işlemlerine ait blok ise aşağıda verilmiştir:



Şekil 1.12 Saklayıcı, öteleyici ve FF bloğunun elektrik devre şeması. Cl saat işaretidir.

Yukarıdaki devreye ilişkin denklemler ise:

$L_A = L_B = L_C = sT_0$
$L_D = T_3$
$L_E = T_1 + T_2$
$J_F = ET_2$
$K_F = sT_0$
$K_1 = T_3FE$
$K_2 = T_1 + T_2E$
$K_3 = T_1 + T_3F'E$
$K_4 = T_2E'$
$K_5 = T_2 + T_3E'$

şeklinde verilebilir.

1.2 Durum eşdeğerliliği, durum uyuşması ve durum indirgemesi

Tasarlanması öngörülen bir devrenin ya da sistemin işlevlerinin sözle anlatımından giderek ortaya çıkarılan durum tablosunda eşdeğerli ya da birbirleriyle uyuşma gösteren durumlar olabilir. Bu özelliği taşıyan durum sınıflarını ortaya çıkarıp, durum tablosunu daha az durumu olan başka bir tabloya dönüştürme işlemine durum indirgenmesi adı verilir.

1.2.1 Durum eşdeğerliği ve tümüyle tanımlanmış tabloların indirgenmesi:

Tanım: M makinesinin d_i ve d_j durumlarının eşdeğerli olması için gerek ve yeter koşul, her iki durum için makinaya uygulanan herhangi bir giriş dizisinin aynı çıkış dizisini oluşturabilmesidir. Bir $d_i \xi d_j$ olarak gösterilir.

ξ bir eşdeğerlik bağıntısıdır, zira bu bağıntı:

$d_i \xi d_i$ (yansıma)
$d_i \xi d_j \Rightarrow d_j \xi d_i$ (bakışlılık)
$d_i \xi d_j$ ve $d_j \xi d_k \Rightarrow d_i \xi d_k$ (geçiş)

özelliklerinin tümünü taşımaktadır. Şu halde M makinesinin durumlar kümesi *D,* ξ bağıntısı ile ayrık eşdeğerlilik sınıflarına bölümlenebilir. Eşdeğerlilik sınıflarının kesişimi boştur. Her eşdeğerlik sınıfı indirgenmiş tablonun bir durumunu oluşturacaktır. Ancak bu bölümleme sonucunda :

- Eşdeğerliliği açıkça beliren,
- Eşdeğerliliği başka durum çiftlerinin eşdeğerliliğini gerektiren durum çiftleri ortaya çıkarılır.

Bu iki kavramın açıklanması aşağıdaki teoremde verilmiştir.

Teorem: Aynı makinenin iki durumunun eşdeğerli olması için gerek ve yeter koşullar, durum tablosunda her giriş için bu iki duruma ilişkin

1. Çıkışların eşit olması ve
2. a) Gelecek durumların doğrudan ya da karşılıklı olarak eşdeğerlik göstermesi,

   2 b) Gelecek durumların eşdeğerliği için doğrudan ya da dolaylı olarak eşdeğerliliği gereken durum çiftleri varsa bunların arasında çıkışta eşdeğerliliği sağlamayan durum çiftinin olmamasıdır.

Gerektirmelerin sağlaması yönlendirilmemiş bağıntı grafından ve gerektirme merdiveni adını alan tablodan yararlanılarak sağlanır.

Örnek:

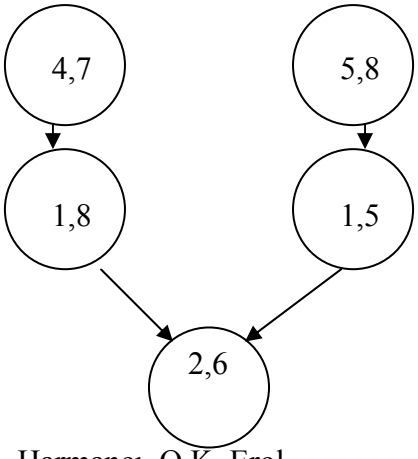
İndirgenmemiş durum tablosu aşağıdaki gibi verilmiş bir sistem olsun:

Q \ I	I_1	I_2	I_3	I_4
Q_1	$Q_1/0$	$Q_2/1$	$Q_5/1$	$Q_1/0$
Q_2	$Q_2/1$	$Q_2/0$	$Q_6/0$	$Q_3/0$
Q_3	$Q_3/1$	$Q_7/0$	$Q_4/0$	$Q_3/0$
Q_4	$Q_1/0$	$Q_7/1$	$Q_4/1$	$Q_3/0$
Q_5	$Q_5/0$	$Q_6/1$	$Q_5/1$	$Q_5/0$
Q_6	$Q_2/1$	$Q_6/0$	$Q_2/0$	$Q_3/0$
Q_7	$Q_8/0$	$Q_7/1$	$Q_4/1$	$Q_3/0$
Q_8	$Q_8/0$	$Q_6/1$	$Q_5/1$	$Q_1/0$

Gerektirme merdiveni oluşturulurken durum numaralama 1'den başlayarak *n*-1'e kadar yukarıda, 2'den başlayarak *n*'ye kadar solda yapılır. Ele alınan durum çiftlerinde öncelikle çıkışlara bakılır. Çıkışlar uyuşmuyorsa o iki durum çifti eşdeğer değildir. Çıkışlar uyuşuyorsa, o zaman uyuşması gereken durum çiftleri (farklı olan durumlar) o göze yazılır. Bu durum çiftleri de kendi aralarında eşdeğerse, o zaman o göze eşdeğer sembolü olan O yazılır. Örnek tablo aşağıdaki gibi oluşturulur:

	1						
2	X	2					
3	X	2,7 4,6 X	3				
4	2,7 4,5 1,3 X	X	X	4			
5	2,6 O	X	X	1,5 6,7 3,5 X	5		
6	X	O	2,3 6,7 2,4 X	X	X	6	
7	2,7 4,5 X	X	X	1,8 O	5,8 4,5 6,7 3,5 X	X	7
8	2,6 O	X	X	1,8 7,6 4,5 X	1,5 O	X	6,7 4,5 1,3 X

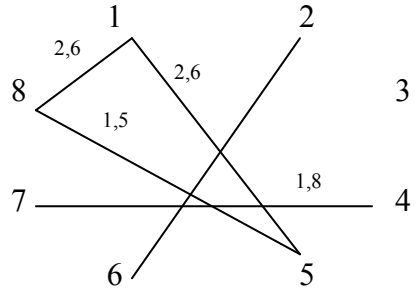
Bu tablodan 2-6 durum çiftinin eşdeğer oldukları açıkça görülmekte, diğer eşdeğer durumlar, başka durumların eşdeğerliliğini gerektirmektedir. Bu gerektirmeler aşağıdaki grafla verilebilir:



Şekil 1.13 Yönlendirilmiş gerektirme grafı

Yönlendirme grafına göre 4 ile 7'nin eşdeğer olması için 1 ile 8'in eşdeğerli olması gereklidir. Benzer şekilde 1-8'in eşdeğerli olması için 2-6'nın eşdeğerli olması gereklidir. 2-6'nın eşdeğerliliği açıktır. Başka bir şart gerekmemektedir. 5-8'in eşdeğerliliği için 1-5'in, ve 1-5'in eşdeğerliliği için yine 2-6'nın eşdeğerliliği gerekmektedir.

Yönlendirilmemiş bağlantı grafını elde etmek için öncelikle tüm durumlar saat yönünde yazılır. Daha sonra eşdeğer olan durum çiftleri bir çizgi ile birleştirilir. Bunlar gerektirme merdivenindeki "O" ile gösterilmiş gözlere ait durum çiftleridir.



Şekil 1.14 Yönlendirilmemiş bağlantı grafı

Şekil 1.14'a göre 1-5-8 tam bağlı bir alt graf oluşturur. Buna "klik" de denir. 3 numaralı durum ait olan küme bir singleton'dur (tek elemanlı küme).

Sonuç olarak 4 adet eşdeğerlik sınıfı ve aşağıdaki indirgenmiş durum tablosu elde edilir:

		I_1	I_2	I_3	I_4
Q_1'	(Q_1, Q_5, Q_8)	Q_1'/0	Q_2'/1	Q_1'/1	Q_1'/0
Q_2'	(Q_2, Q_6)	Q_2'/1	Q_2'/0	Q_2'/0	Q_3'/0
Q_3'	Q_3	Q_3'/1	Q_4'/0	Q_4'/0	Q_3'/0
Q_4'	(Q_4, Q_7)	Q_1'/0	Q_4'/1	Q_4'/1	Q_3'/0

1.2.2 Durum uyuşması, uyuşma bağıntısı ve tümüyle tanımlanmamış durum tabloların indirgenmesi:

Tanım:

Uyuşma bağıntısı: Tümüyle tanımlanmama sözle anlatımın bir sonucu olabileceği gibi, durum kodlaması sonucu oluşan durum değişkenlerinin kullanılmayan kombinezonları olarak da ortaya çıkabilir. Bu özelliği olan tabloları indirgemekte durum eşdeğerliliği yerine durum uyuşması kullanılır. Eşdeğerlilik bağıntısı yerini uyuşma bağıntısına bırakır. Uyuşma bağıntısı yansıma ve bakışlılık özelliği olan, ancak geçiş özelliğini her zaman sağlamayan ikili bir bağıntı şeklidir ve γ ile gösterilecektir.

Örnek:

$d(x,y)$, x ile y arasındaki uzaklığı göstersin. Buna göre:

$R_\gamma = \{(a,b) |\ d(a,b) \le 2, \quad a, b \in N\}$ *olarak tanımlanır.*

$d(1,3) \le 2$ ve $d(3,5) \le 2$ ancak $d(1,5) > 2$ olacağından

$1\ \gamma\ 3$ ve $3\ \gamma\ 5$ yazılabilirken $1\ \gamma\ 5$ yazılamaz. Bu örnek geçiş özelliğinin bulunmadığını gösterir. Ancak 1 ile 2 ve 2 ile 3 arasındaki uzaklıklar söz konusu olsaydı, geçişlilik özelliğinden bahsedilebilirdi. Bu nedenle uyuşma bağıntısı geçiş özelliğini her zaman sağlamaz diyebiliriz.

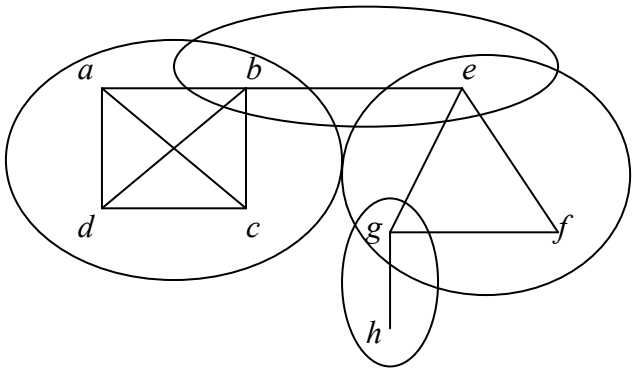
Uyuşanlar sınıfı: Uyuşma bağıntısı tanımlandığı küme üzerinde uyuşanlar sınıflarını oluşturur. Uyuşanlar sınıfının her eleman çifti arasında ikişer ikişer uyuşma vardır. Diğer bir deyişle uyuşanlar sınıfı içinde geçiş özelliği vardır. İki uyuşanlar sınıfının kesişimi boş olmayabilir.

En üst uyuşanlar sınıfı: Bir uyuşanlar sınıfının tanımlandığı küme içinde kendini içine alan daha üst bir sınıf bulunamıyorsa, söz konusu uyuşanlar sınıfına en üst uyuşanlar sınıfı denir. En üst uyuşanlar sınıfı, klik adını da alır. Grafı tam graftır.

Örnek:

$(a, b, c, d) \supseteq (a, b, c) \supseteq (a, b)$ şeklinde yazılan uyuşanlar sınıfında (a, b, c, d) bağıntısı en üst uyuşanlar sınıfı özelliği gösterir.

R γ ={ $a \gamma b$, $a \gamma c$, $a \gamma d$, $b \gamma c$, $b \gamma d$, c γ d, $b \gamma e$, $e \gamma f$, $e \gamma g$, $g \gamma f$, $g \gamma h$} bağıntı kümesi, çiftler kümesi olsun. Bu bağıntı kümelerinde $a \gamma b$ ifadesi a γ a, a γ b, b γ a, b γ b ifadelerini kapsar ve bu dört ifadenin yerine kullanılır. Şekil 1.15 ile küme içindeki uyuşan gruplar gösterilmiştir..



Şekil 1.15 Küme içindeki uyuşan gruplar

{ {a, b, c, d} {b, e} {e, f, g} {g, h} } => Tam örtü

Örtü: Elemanlarının bileşimi, tanımlandığı kümenin tüm elemanlarını içine alan uyuşanlar sınıfları kümesine verilen ad.

Tam örtü: Bir küme üzerinde bir uyuşma bağıntısının oluşturduğu en üst uyuşanlar sınıfları kümesidir. Şekil 1.15'de {b, e} grubunu bir örtü içerisine almayabiliriz. Bu durumda da bütün elemanlar (düğüm noktaları) örtülmüş olur ancak bu tam örtü olmaz.

Uyuşan durumlar: M makinasının d_i ve d_j durumlarının uyuşma göstermesi, her iki durum için makinaya uygulanan herhangi bir giriş sekansı tanımlanmamış çıkışlar dışında aynı çıkış dizisini oluşturabilmesi ile tanımlanır ve $d_i \gamma d_j$ ile gösterilir.

$d_i \gamma d_i$

$d_i \gamma d_j$ => $d_j \gamma d_i$

$d_i \gamma d_j$ ve $d_j \gamma d_k$ ise buradan $d_i \gamma d_k$ yazılamaz.

Örnek:

d_1 = ab∅ef
d_2 = a∅fef
d_3 = acfef

d_1 ile d_2, d_2 ile d_3 uyuşmakta ancak d_1 ile d_3 uyuşmamaktadır.

M makinasının durumlar kümesi *D* üzerinde uyuşması açıkça beliren ya da uyuşması başka durum çiftlerinin uyuşmasını gerektiren uyuşan durum çiftleri ortaya çıkarılır.

Teorem: Aynı makinanın iki durumunun uyuşması için gerek ve yeterli koşullar, her giriş için bu iki duruma ilişkin:

1. Çıkışların uyuşma göstermesi
2. a) Gelecek durumların doğrudan ya da karşılıklı olarak uyuşma göstermesi

   2 b) Gelecek durumların uyuşma göstermesi için doğrudan ya da dolaylı olarak uyuşmaları gereken durum çiftleri varsa, bunların arasında çıkışta uyuşma sağlamayan durum çiftinin olmamasıdır.

Durum indirgemesi: Uyuşan durum sınıflarını ortaya çıkarıp, durum tablosunu daha az durumlu başka bir tablo haline dönüştürmeye durum indirgemesi denir.

Tam örtü ile indirgeme: Uyuşanlar çiftlerinin bulunmasında yönlendirilmemiş uyuşma grafı ve gerektirme merdiveni kullanılır. Uyuşma grafı üzerinden tam örtüyü ortaya çıkarma zor bir iş değildir. Tam örtünün herhangi bir sınıfına yeni bir durum atayıp indirgenmiş makina elde edilebilir, ancak bu makina çoğu kere en iyi indirgenmiş makina olmaz. Optimal çözümü bulmak için kapalı örtü tanımı geliştirilmiştir.

Kapalı örtü: Kapalı örtü öyle bir örtüdür ki, bu örtüden bir uyuşanlar sınıfına ait bir durum çiftinin uyuşması için diğer durum çiftlerinin uyuşması gerekiyorsa, bu durum çiftlerinin her biri kapalı örtünün herhangi bir uyuşanlar sınıfı içinde yer almalıdır. Tam örtü her zaman kapalı bir örtüdür.

Minimal kapalı örtü: En az sayıda uyuşanlar sınıfı ile örtme ve kapalılık özelliklerini taşıyan uyuşanlar sınıfları kümesidir.

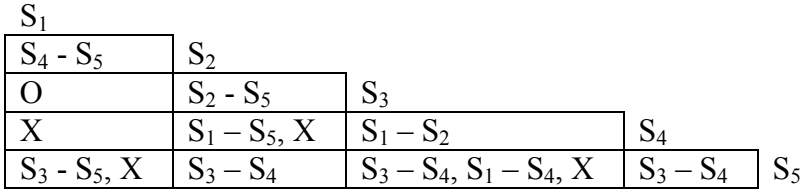
Tam örtü bulunduktan sonra, minimal kapalı örtüye geçmede gerektirme grafından yararlanılır. Ancak bu işin sistemli ve hızlı çözüme götüren bir yolu olduğu söylenemez.

Örnek:

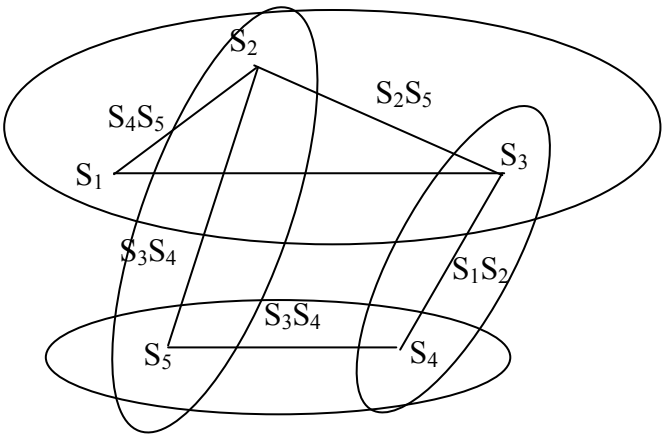
İndirgenmemiş durum tablosu aşağıdaki gibi olan bir sonlu durumlu makina ele alınsın. Bu makinanın durumlarını tam örtü ve kapalı örtü kavramlarını kullanarak indirgeyiniz.

	i_1	i_2	i_3	i_4
S_1	-	-	$S_5/1$	-
S_2	-	$S_5/1$	$S_4/-$	-
S_3	$S_3/0$	$S_2/1$	-	$S_1/0$
S_4	$S_3/0$	$S_1/1$	$S_4/0$	-
S_5	$S_4/0$	-/1	$S_3/-$	$S_4/-$

İndirgenmemiş durum tablosu



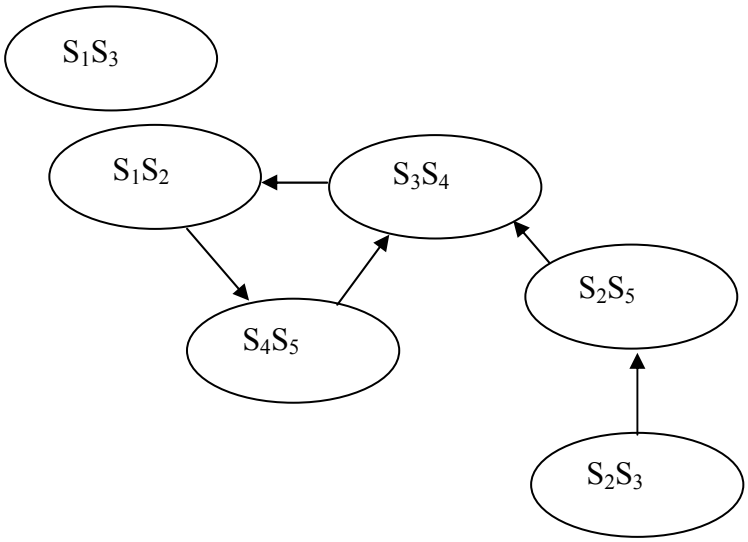
Gerektirme merdiveni



Şekil 1.16 Yönlendirilmemiş bağıntı grafı ve en üst uyuşanlar sınıfları aşağıdaki gibi bulunabilir:

{ (S_1 S_2 S_3), (S_3 S_4), (S_2 S_5), (S_4 S_5) }
a b c d

Buna göre indirgenmiş makinada 4 adet durum bulunur.



Şekil 1.17 Gerektirme grafı

Ancak gerektirme grafında (S_1 S_2), (S_3 S_4), (S_4 S_5)'in kapalılık ve örtme özelliği olan uyuşanlar sınıfları kümesi olduğu görülebilir. Buna göre indirgenmiş makinaların yeni durum tabloları aşağıdaki gibi bulunur:

		i_1	i_2	i_3	i_4
(S_1 S_2)	a	-	c/1	c/1	-
(S_3 S_4)	b	b/0	a/1	b,c/0	a/0
(S_4 S_5)	c	b/0	a/1	b/0	b,c/-

Mealy indirgenmiş durum tablosu

		i_1	i_2	i_3	i_4	Ç
u	(a/1)	-	z	z	-	1
v	(a/0)	-	z	z	-	0
w	(b/0)	w	u	w	v	0
z	(c/1)	w	u	w	w	0

Moore indirgenmiş durum tablosu

BÖLÜM 2

Biçimsel Dillerin Matematiksel Temelleri

2.1 Kümeleri tümevarım yolu ile tanımlama

E tanımlanacak küme olsun:

Taban:
Yapı taşı elemanları kümesi veya taban B ile gösterilsin. Bu kümenin içindeki elemanlardan hareketle diğer elemanlar türetilebiliyorsa, bu elemanlara yapı taşı elemanları denir. Örnek olarak Alfabe bir dil kümesinin yapı taşı elemanları kümesidir.

Kurallar:

Bir kümenin içindeki elemanlardan hareketle yeni elemanları türetmenin kurallarıdır. Bir dil, alfabeden belli kurallar neticesinde türetilmiş kelimeler topluluğudur (kümesidir).
$\Omega = \{ f_1, f_2, ..., f_n \} \wedge \forall f_i \in \Omega \quad f_i: E \text{ x } E \text{ x } E \text{ x } .. \text{ x } E \rightarrow E$
$\forall x_1, x_2, .. , x_p \in E \; f_i(x_1, x_2, ..., x_p) = x \in E$ (X yeni eleman)

Kapanış:
$\Omega(E) = E$
E kümesi, taban elemanları ve sadece kurallar uygulanması sonucu oluşan elemanlardan başka bir elemanı olmayan kümedir.

Tüme varımla gösterilim:
$B_0 = B$
$B_1 = \Omega(B_0) \cup B_0$ (kurallar B_0'a uygulanır.)
$B_2 = \Omega(B_1) \cup B_1$
...
$B_{i+1} = B_i \cup \Omega(B_i) = B_i = E$ (kapanış). i sayısı sonlu olabileceği gibi sonsuza da gidebilir.

Elemanın yüksekliği:
$H(x) = \min\{ i \mid x \in B_i \}$. Elemanın ilk defa kaçıncı sırada elde edildiğini gösterir. Kelimenin uzunluğudur.

Örnek:
Çift sayılar:

i) $0 \in E$
ii) $n \in E \Rightarrow (n + 2) \in E$
iii) Bu kuralın uygulanması ile elde edilenler dışında hiç bir sayı çift sayı değildir.

2.2 Alfabe ve diller

Tanımlar:

Alfabe: Sonlu, boş olmayan simgeler veya karakterler kümesi $\sum$.

Sözcük: $\sum$'nın sonlu bir dizisi veya katarı.

Sözcük uzunluğu: Sözcükteki simge sayısı.

Boş sözcük: Uzunluğu sıfır olan sözcük. Λ veya "e" ile gösterilir.

Sözcük bitiştirme:

$x = a_1a_2...a_n$
$y = b_1b_2...b_m \Rightarrow xy = a_1a_2..a_nb_1b_2...b_m$ ($x\&y$)

Sözcük bitiştirme işleminin etkisiz elemanı Λ' dir.

$x = \Lambda \Rightarrow xy = y$
$y = \Lambda \Rightarrow xy = x$

Boş katar ve birleşme özelliği (asosiyativite) gösteren bitiştirme işlemi (konkatenasyon) ile alfabe üzerinde oluşturulmuş sözcükler kümesi $\sum^*$ bir monoiddir. Bundan sonra aşağıda verilmiş olan kümeler bilgisayar biliminde sıklıkla kullanılır.

Katar Tersi ya da ters katar (ing: Reverse of a string)

Örnek:

$(baba)^R = abab$; $(ana)^R = ana$

Ters katar tanımı (tüme varım ile)

1. $|w| = 0 \Rightarrow w^R = w = \Lambda$; |w|, w sözcüğünün karakter sayısını gösterir.
2. $|w| = n + 1$ ve u^R mevcut olduğunu var sayalım. $|u| = n$, (w ve u) katarlar, $n \in$ R n uzunlukta bir katar verilmiştir ve bunun tersi u^R dur.
   $w = ua \wedge a \in \Sigma \Rightarrow w^R = au^R$

Örnek:

$|w| = 1 \Rightarrow w = \Lambda a$, $w^R = a\Lambda = w|w| = 2 \Rightarrow w = ua$ ($u = b$), $w^R = au^R = au = ab$

$|w| = 3 \Rightarrow w = ua$, $u = cb$, $w^R = au^R = abc$

Teorem:

$(wx)^R = x^R.w^R \wedge |x| = n$, $|w| = m \wedge m \in N$

Tanıt:

$|x| = 0 \Rightarrow x = \Lambda$

$n = |x| = 0$ için $(wx)^R = (w\Lambda)^R = w^R = \Lambda w^R = \Lambda^R w^R = x^R w^R$

Varsayım:

$|x| \leq n \Rightarrow (wx)^R = x^R w^R$

$|x| = n + 1$ için tanıt

$x = ua \wedge |u| = n \wedge a \in \Sigma \wedge x^R = au^R$ (tanım)

$(wx)^R = (w(ua))^R = ((wu)a)^R = a(wu)^R = a(u^R w^R) = au^R w^R = x^R w^R$

Örnek:

$(atlı\ karınca) = (karınca)^R (atlı)^R$

Tanımlar:

Σ^+ *(boş olmayan sözcüklerin kümesi) kümesi:* ($\sum$ alfabesi üzerinde)

i) $a \in \sum \Rightarrow a \in \Sigma^+$ (taban) (Σ'nın her elemanı Σ^+'nın da elemanıdır.)

ii) $(x \in \Sigma^+ \wedge a \in \sum) \Rightarrow ax \in \Sigma^+$ (Σ'nın her elemanı Σ^+'nın da elemanıdır. Bitiştirme işlemi, a bitiştir x)

iii) i) ve ii)'nin sonlu uygulamaları dışında elde edilemeyen hiç bir eleman $\sum^+$' nın elemanı olamaz.

Örnek:

$\Sigma = \{ a, b\} \Rightarrow \Sigma^+ = \{ a, b, aa, ba, ab, bb, aaa, aab, ...\}$. Σ^+ içinde boş katar yoktur. Boş katara sözcük uzayının sıfır elemanı gibi bakabiliriz.

Σ^* *kümesi:* (Σ alfabesi üzerinde)

i) $\Lambda \in \Sigma^*$
ii) $x \in \Sigma^* \wedge a \in \Sigma \Rightarrow ax \in \Sigma^*$
iii) i) ve ii)'nin sonlu uygulamaları dışında elde edilemeyen hiç bir eleman Σ^*'nın elemanı olamaz.

Örnek:

$\Sigma = \{ a, b\} \Rightarrow \Sigma^* = \{ \Lambda, a, b, aa, ba, ab, bb, aaa, aab, ...\}$
$\Sigma = \{0, 1\} \Rightarrow \Sigma^* = \{ \Lambda, 0, 1, 00, 01, 10, 11, 000, 001, ... \}$

$(x \in \Sigma^*) \wedge (\forall n \in N)$ ise x^n'nin tanımı (Bir katarın n'inci kuvvetinin tanımı)

i) $x^0 = \Lambda$ (bitiştirmenin etkisizi olduğu böyle tanımlanmıştır)
ii) $x^{n+1} = x^n.x = (x^n \,\&\, x)$

Örnek 1:

$\Sigma = \{a, b\}$ $x = ab$
$x^0 = \Lambda, x^1 = ab, x^2 = abab, x^3 = ababab$

Örnek 2:

$\{ a^n b^n \mid n \geq 0 \}$ yanda belirtilen kümeyi tanımlar: $\{ \Lambda, ab, aabb, aaabbb, ... \}$

Tanım:

Σ sonlu bir alfabe olsun. Σ üzerinde bir dil, Σ^*' ın bir alt kümesidir. Σ^*'dan bir alt küme seçmenin kuralları ilerideki bölümlerde gramer olarak işlenecektir.

Örnek:

$\{ a^n b^m \mid n, m \in N \}$, $\{ a, b \}$ üzerinde bir dildir. Bu dilde *b*'ler *a*'ların önüne geçemiyor. Bu dilde sade *a*'lar ve *b*'ler olabilir. Örneğin *ba* sözcüğü bu dile ait değildir.

Tanım: (Çarpım Diller)

A ve *B,* Σ üzerinde diller ise
A.B küme çarpımı yeni bir dil oluşturur.
$AB = \{ xy \mid x \in A \wedge y \in B \}$
$AB = \{ z \mid z = xy \wedge x \in A \wedge y \in B \}$

Örnek:

$\Sigma = \{ a, b \}$
$A = \{ \Lambda, a, ab \}$, $B = \{ a, bb \}$ diller olsun.
$AB = \{ a, bb, aa, abb, aba, abbb\}$, burada a terimi ($\Lambda a$)'den gelmektedir.
$BA = \{ a, aa, aab, bb, bba, bbab \}$
İki çarpım dilin elemanlarının hepsinin aynı olmadığı görülmektedir..

Teoremler:

$\varnothing$ boş dili (not: tek elemanı Λ olan dil $\varnothing$ değildir); *A*, *B*, *C*, *D* aynı Σ alfabesinin farklı dillerini göstersin. Buna göre:

1) $A\varnothing = \varnothing A = \varnothing$. ($\varnothing$ boş dildir, boş katar değildir).

2) $A\{\Lambda\} = \{\Lambda\}A = A$
3) $(AB)C = A(BC)$
4) $(A \subset B)\ \Lambda\ (C \subset D) \Rightarrow AC \subset BD$
5) $A(B \cup C) = AB \cup AC$
6) $(B \cup C)A = BA \cup CA$
7) $A(B \cap C) \subset AB \cap AC$
8) $(B \cap C)A \subset BA \cap CA$

Tanıtlar:

1) $A\varnothing = \{xy \mid x \in A \wedge y \in \varnothing\}$, $y \in \varnothing$ önermesi her zaman yanlıştır, buna göre $\forall\ x, y$ için $x \in A$ ve $y \in \varnothing$ önermesini sağlayan xy yoktur. Hiç bir x, y şartı sağlamadığına göre $A\varnothing = \varnothing$. Benzer bir tanıt $\varnothing A = \varnothing$ için de kullanılabilir.

4) Bu teoremin tanıtı doğrudandır. Farz edelim ki $A \subset B\ \Lambda\ C \subset D$, ve z de AC kümesinin rasgele bir elemanı olsun. Bu durumda $z = xy$ olur ki burada $x \in A$ ve $y \in C$. $A \subset B$ ve $C \subset D$ olduğundan, x'in B'ye, y'nin de D'ye ait oldukları ($x \in B, y \in D$) sonucu çıkar. Buna göre $z = xy \in BD$ olur. z, AC'nin rasgele bir elemanı olduğundan $AC \subset BD$ sonucu ispatlanmış olur.

7) $A(B \cap C) \subset AB \cap AC$ teoreminin ispatı: $A(B \cap C)$ ise $\forall z\ (z = xy\ \Lambda\ x \in A \wedge y \in B \cap C)$ yazılabilir. Buradan $x \in A \wedge y \in B \wedge x \in A\ \Lambda\ y \in C$ olur. İlerlersek $xy \in (AB \cap AC)$. Şu halde $\forall z\ (z \in A(B \cap C) \Rightarrow z \in AB \cap AC$
$AB \cap AC \subseteq A(B \cap C)$ her zaman doğru olamayacağına ilişkin karşı örnek:
$A = \{\ a^n \mid n \in N\ \}$
$B = \{\ a^n b^n \mid n \in N\ \}$
$C = \{\ b^n \mid n \in N\ \}$
kümelerini ele alırsak, örneğin $z = a^5 b^2$ değerinde olan bir eleman hem AB, hem de AC'ye aittir, yani $z \in AB \cap AC$'dir. Oysa $z \in A(B \cap C)$ doğru değildir, zira $B \cap C = \{\wedge\}$'dir:
$z = a^5 b^2 = a^3 a^2 b^2$ olsun
$z \in AB \cap AC$
$z \notin A(\ B \cap C\ )$ çünkü $B \cap C = \{\Lambda\}$

Tanım:

A^n: A $\sum$ üzerinde bir dil olsun.

i) $A^0 = \{\ \Lambda\ \}$
ii) $A^{n+1} = A^n A;\ \ \forall n \in N$

Örnek:

$\sum = \{\ a, b\}$, $A = \{\ \Lambda, a, ab\}$
$A^1 = A$
$A^2 = \{\Lambda, a, ab, aa, aab, aba, abab\ \}$

Teoremler:

A ve B, $\sum$ üzerinde tanımlanmış diller ise:

1) $A^m A^n = A^{m+n}$
2) $(A^m)^n = A^{mn}$
3) $A \subset B \Rightarrow A^n \subset B^n$..

Tanıtlar:

Tanıt 1: Bunun için 1.tümevarım yöntemini kullanılabilir.

$n = 1$ için $A^m A^1 = A^{m+1}$ bulunur. Bu eşitlik yukarıdaki tanımdan gelmektedir.

n için doğru kabul eder ve $n + 1$ için doğruluğunu gösterirsek

$A^m A^{n+1} = A^m A^n A^1$. Dil bitiştirmenin asosiyatiflik özelliği uyarınca $A^m(A^n A^1)$ yerine $(A^m A^n)A^1$ yazılabilir. Buradan $A^{m+n}A^1$ bulunur ki bu da tanıtın yapıldığını gösterir.

Tanıt 2: Aynı yöntem kullanılırsa, $n = 1$ için doğruluğun sağlanması açıktır. $A^m = A^m$. n için doğru kabul edilirse, $n + 1$ için:

$(A^m)^{n+1} = (A^m)^n (A^m)^1 = A^{mn+m} = A^{m(n+1)}$ ispat yapılmıştır.

Tanıt 3: $A^2 = A$ x A, $B^2 = B$ x B birer dildir. $A \subseteq B \Rightarrow \forall x \in A \Rightarrow x \in B$

$n = 2$ için:

$A^2 = \{ xy \mid x \in A \ \wedge y \in A\}$

$\forall xy \in A^2 \Rightarrow xy \in B^2 \Rightarrow A^2 \subseteq B^2$

$n = n + 1$ için:

n için doğru kabul edilir ve $n + 1$ için de aynı şekilde tanıtlanır.

Tanım:

Pozitif kapanış:

$$A^+ = \bigcup_{n=1}^{\infty} A^n = A \cup A^2 \cup \ldots$$

Yıldız kapanışı (Kleen Yıldızı veya Kleen Kapanışı):

$$A^* = \bigcup_{n=0}^{\infty} A^n = A^0 \cup A \cup \ldots$$ Boş katarı da içine alan kümedir.

Teorem:

A ve B, Σ üzerinde tanımlı iki dil ve $n \in N$ olsun. Bu durumda aşağıdaki bağıntılar geçerlidir:

1. $A^* = \{\Lambda\} \cup A^+$. Tanımın kendinden gelmektedir.
2. $A^n \subseteq A^*$, $n \geq 0$ için. Aynı şekilde …
3. $A^n \subseteq A^+$, $n \geq 1$ için. Aynı şekilde …
4. $A \subseteq AB^*$. Tanıt: $A^0 \subseteq B^* \Rightarrow A \subseteq AB^*$
5. $A \subseteq B^*A$. İspatı açıktır.
6. $A \subseteq B \Rightarrow A^* \subseteq B^*$. $A \subseteq B \Rightarrow A^n \subseteq B^n$ teoreminden gelmektedir. Bütün n' ler için doğru ise, bütün n'lerin bileşimi için de doğrudur.
7. $A \subseteq B \Rightarrow A^+ \subseteq B^+$
8. $AA^* = A^*A = A^+$
9. $\Lambda \in A \Leftrightarrow A^+ = A^*$
10. $(A^*)^* = A^*A^* = A^*$
11. $(A^*)^+ = (A^+)^* = A^*$
12. $A^*A^+ = A^+A^* = A^+$
13. $(A^*B^*)^* = (A \cup B)^* = (A^* \cup B^*)^*$. Bu teoremi düzenli ifadelerin yalınlaştırılmasında kullanacağız.

Tanıt 8:

$AA^* = A^+ \Rightarrow A(A^0 \cup A^1 \cup \ldots) = A \cup A^2 \cup \ldots = A^+$

Tanıt 9:

$\Lambda \in A \Rightarrow A \cup \{\Lambda\} = A$
$A \cup A^0 = A$

$A^+ = A \cup \{ \overset{\infty}{\underset{n=2}{\cup}} A^n \} = A^0 \cup A \cup \{\ldots\} = A^*$

Tanıt 9 ters yön:

$A \cup A^2 \ldots = A^0 \cup A \cup \ldots$ ise
$A^0 \subseteq A \cup A^2 \cup \ldots$
Bu içindelik iki şekilde yorumlanabilir:

i) $\Lambda \in A \Rightarrow A^0 \subseteq A$

ii) $\Lambda \notin A \Rightarrow \exists\, i,\ \Lambda \in A^i\ ,\ i \in (N^+ - 1)$
HAlbuki $\Lambda \notin A \Rightarrow \forall\, x \in A^i\ (\,|\,x\,| \geq i)$
Oysa $|\,\Lambda\,| = 0 \Rightarrow \Lambda \notin A^i \wedge i \geq 2$. Tezatla karşılaşıldığından ii) şıkkı doğru olamaz.
Buna göre i) şıkkı gösterilmek istenendir.

Tanıt 13:
$(A^*B^*)^* = (A \cup B)^*$

Birinci yön:
$A \subseteq A \cup B \wedge B \subseteq A \cup B$
$A^* \subseteq (A \cup B)^* \wedge B^* \subseteq (A \cup B)^*$
$A^*B^* \subseteq (A \cup B)^*\ (A \cup B)^*$
$A^*B^* \subseteq (A \cup B)^*$
$(A^*B^*)^* \subseteq ((A \cup B)^*)^* = (A \cup B)^*$

İkinci yön:
$A \subseteq A^* \subseteq A^*B^* \wedge B \subseteq B^* \subseteq A^*B^*$. Bu ifadeleri A^* ve B^* içinde Λ olduğu için yazabiliyoruz.

$A \cup B \subseteq A^*B^*$
$\Rightarrow (\mathrm{A} \cup \mathrm{B})^* \subseteq (A^*B^*)^*$.
$(A^*B^{*})^* \subseteq (A \cup B)^*$ ve $(A \cup B)^* \subseteq (A^*B^*)^*$ olduğuna göre eşitlik doğrudur.

Teorem:

A ve B, Σ^* üzerinde tanımlanmış alt kümeler ve $\Lambda \notin A$ olsun
$X = AX \cup B$ denkleminin tek çözümü $X = A^*B$'dir.

Tanıt:

I) $X = A^*B$ bir çözümdür.
$\{\Lambda\}B = B$ özelliğini göz önüne alarak
$A^*B = AA^*B \cup B = A^+B \cup B = (A^+ \cup \{\Lambda\})\ B = A^*B$

II) X, denklemin bir çözümü olsun
$X = A^*B$ olduğunun tanıtlanması için:
$X \supset A^*B$ ve $X \subset A^*B$ ayrı ayrı tanıtlanacaktır.

a) $X \supset A^*B$'nın tanıtı:

İlk aşamada ($X \supset A^nB$) olduğunu 1.tüme varım yöntemi ile tanıtlayalım.
$n = 0$, $A^n = A^0 = \{\Lambda\} \Rightarrow A^0B = B$
Öte yandan $X = AX \cup B \Rightarrow X \supset AX \wedge X \supset B$
$X \supset B$ ve $A^0B = B \Rightarrow X \supset A^0B$
Varsayım: $X \supset A^nB$
$X \supset AX \Rightarrow X \supset A\,(A^nB)$, $X \supset A^{n+1}B$
Şu halde $\forall n$, ($X \supset A^nB$) ve öte yandan
$A^*B = \cup_i A^iB$ olduğundan sonuç olarak:
$X \supset A^*B$ bulunmuş olur.

b) İkinci aşamada $X \subset A^*B$ 2.tümevarım yöntemi kullanılarak tanıtlanacaktır. Bunun için herhangi bir n değerinden daha küçük n'ler için tanıtlanacak ifadenin doğru olduğu kabul edilip n için doğruluğu ispat edilir.

$\forall x(x \in X \Rightarrow x \in A^*B)$
x'in uzunluğu n olsun. Burada n rasgele bir uzunluktur. 2.tümevarım hipotezine göre uzunluğu n'den kısa her zincir için tanıtlanmak istenenin doğru yani
$\forall w\{ |w| < |x| \Rightarrow w \in X \Rightarrow w \in A^*B \}$ varsayımına göre tanıtı yapacağız. Açıklık getirmek için A*B'de en kısa katarın uzunluğu, B'nin en kısa katarının uzunluğuna eşit olduğuna dikkat çekilebilir. Tanıtın birinci kısmından X'in en kısa katarı için bu söylem geçerlidir.

$(X = AX \cup B \wedge x \in X) \Rightarrow (x \in AX \vee x \in B)$
i) $x \in B \Rightarrow x \in A^*B$
ii) $x \in AX \Rightarrow x = yz \wedge y \in A \wedge z \in X$
$\Lambda \notin A \Rightarrow |y| \neq 0$
$| y | \neq 0 \Rightarrow |z| < |x|$

Eğer | x |, B'nin en küçük katarının uzunluğuna eşitse, | z | < | x | yoktur yani x ∈ B olmadığından ii) şıkkı doğru olamaz. Hipoteze göre:

$|z| < |x| \wedge z \in X \Rightarrow z \in A^*B$

Şu halde:

$x = yz \in AA^*B$
Yani $x \in A^+B \subseteq A^*B$ olur. Bu durumda tanıt bitmiştir.

2.3 Bağıntılar ve kapanış

2.3.1 Bağıntılar:

Bağıntılar bir yapıyı karakterize eder. Önceki bölümde kümeleri incelemiştik. Bu bölümde kümelerin elemanları arasındaki küme içi bağıntılar ile temsil edilen bazı basit yapı formlarını inceleyeceğiz. Bağıntılar bilgisayar biliminin temel kavramlarından biridir. Bileşik bir veri yapısı, dizi, liste, ağaç gibi, bir veri nesneleri kümesini bağıntılar aracılığı ile bir arada göstermek amacıyla kullanılır. Bağıntılar ayrıca program yapısı, algoritma analizi gibi konularda da önemli bir rol oynar.

Küme içi bağıntılar:
$B = A$ kaynak ile hedef aynı
$R_\alpha \subseteq A \times A \Rightarrow M_\alpha$ kare matris

Graf: Koşut bağlı yay bulunmaz. Yönlendirilmiştir. Tek çevre olabilir.

Küme içi bağıntının kuvvetleri:

Tanım: α bağıntısının ($n \in N$) n'inci kuvveti α^n ile gösterilir ve tümevarım ile tanımlanır.

i) $\alpha^0 = \{(x,x) \mid x \in A\}$
ii) $\alpha^{n+1} = \alpha^n\alpha \ \wedge \ n \in N$

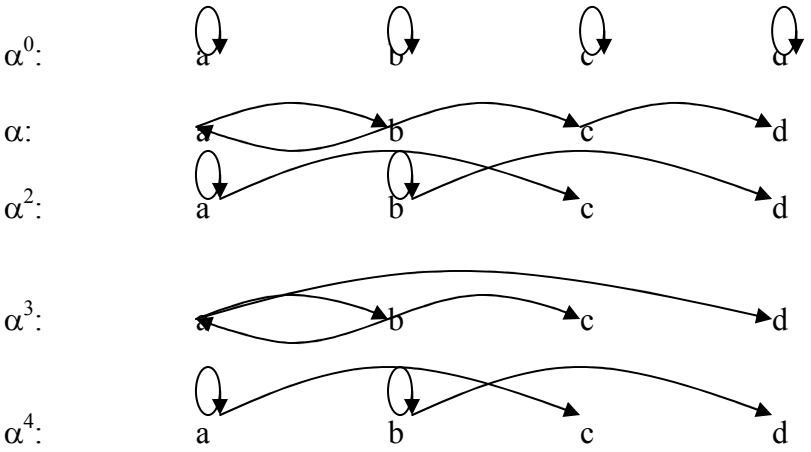
Teorem: (tanıt tüme varım ile öğrenci tarafından yapılacaktır)

$\alpha^m\alpha^n = \alpha^{m+n} \wedge (n, m \in N)$
$(\alpha^m)^n = \alpha^{mn} \wedge (n, m \in N)$
$\alpha^{-p} = (\alpha^{-1})^p = (\alpha^p)^{-1}$

Özellik:
$\langle x, y \rangle \in \alpha^n \Rightarrow$ grafta x ile y bağlı ve aralarında n uzunlukta en az bir dolaşı mevcut.

Örnek:

A = {a, b, c, d} elemanlarından oluşan bir küme olsun. α^n, uzunluğu n olan bağıntıları göstersin. Buna göre:



Şekil 2.1 Bağıntının kuvvetlerinin açık gösterimi

$\alpha^2 = \alpha^4$
$\alpha^4\alpha = \alpha^2\alpha = \alpha^3$
$\alpha^5 = \alpha^3$
$\alpha^6 = \alpha^4 = \alpha^2$
$\alpha^{2n+1} = \alpha^3$
$\alpha^{2n} = \alpha^2$

Şekil 2.2 Bağıntının kuvvetleri grafı

Teorem:

A, n elemanlı bir sonlu küme ise ve α, A üzerinde bir bağıntı ise $[0 - (2^n)^2]$ kapalı aralığında $\alpha^s = \alpha^t$ eşitliğini sağlayan en az bir (s, t) çifti bulunur.

Tanıt:

$\alpha \subseteq A \times A$
Card $(A \times A) = n^2$, Card $P(A \times A) = (2^n)^2$
Şu halde A üzerinde $(2^n)^2$ den fazla birbirinden farklı bağıntı kurulamaz. Buna göre:
$\alpha^0, \alpha^1, \alpha^2, ..., (\alpha^{2n})^2$ bağıntılarının sayısı $(2^n)^2+1$'dir. Şu halde bunlardan en az ikisi birbirinin aynısıdır.

Teorem:

α, A kümesi üzerinde tanımlanmış bir bağıntı ise ve $\alpha^s = \alpha^t$ ($s<t$ ve $p= t-s$ olmak üzere)

a) $\alpha^{s+k} = \alpha^{t+k}$; $\forall k \geq 0$
b) $\alpha^{s+kp+i} = \alpha^{s+i}$; $\forall k > 0 \wedge \forall i\ (0 < i < p)$
c) $\sum = \{\alpha, \alpha^1, \alpha^2, ..., \alpha^{t-1}\}$ olduğuna göre
$\forall q\{\alpha^q \in \sum \wedge q \geq 0 \wedge q \in N\}$

a) ve b) şıklarını tümevarım ile siz tanıtlayınız.
c) şıkkının tanıtı:
$q < t$ ise $\alpha^q \in \sum$ tanım uyarınca
$q \geq t$ ise b şıkkına göre $q = s + kp + i$ ($i < p$) olmak üzere yazılabilir. Buna göre $\alpha^q = \alpha^{s+i}$
$i < p \Rightarrow s + i < t \Rightarrow \alpha^q \in \sum$.

2.3.2 Bağıntılar üzerinde kapanış işlemleri (Closure Operations):

Bir bağıntının kapanışı:
Bir bağıntıdan yeni bağıntı türetme yollarından biri.

Örnek:

Düğümler ve ayrıtlar ile bir graf, bir haberleşme ağını temsil etsin. Bu grafa ilişkin bağıntı lokal bir bağıntıdır. Bu yerel (local) bağıntıdan giderek birbiri ile arasında haberleşme yolu olan düğümleri bulma işlemi ise daha global bir bağıntı şeklidir. Bu bağıntıya 1. bağıntının kapanışı diyoruz.

Tanım:

A üzerinde tanımlanmış bir küme içi bağıntı α'nın yansımalı kapanışı (α') ise aşağıdaki koşullar sağlanır:

i) α' yansımalı bir bağıntıdır.
ii) $\alpha' \supseteq \alpha$
iii) $\alpha'' \supseteq \alpha \wedge \alpha''$ yansımalı bağıntı $\Rightarrow \alpha'' \supseteq \alpha'$

A üzerinde tanımlanmış bir küme içi bağıntı α'nın bakışlı kapanışı (α') ise aşağıdaki koşullar sağlanır:

iv) α' bakışlı bir bağıntıdır.
v) $\alpha' \supseteq \alpha$
vi) $\alpha'' \supseteq \alpha \wedge \alpha''$bakışlı bağıntı $\Rightarrow \alpha'' \supseteq \alpha'$

A üzerinde tanımlanmış bir küme içi bağıntı α'nın geçişli kapanışı (α') ise aşağıdaki koşullar sağlanır:

vii) α' geçişli bir bağıntıdır.
viii) $\alpha' \supseteq \alpha$
ix) $\alpha'' \supseteq \alpha \wedge \alpha''$geçişli bağıntı $\Rightarrow \alpha'' \supseteq \alpha'$

Buna göre (α)'dan (α')'ünü elde etmek için (α)'ya *A*'da gerekli ve yalnız gerekli sıralı çiftleri eklemek yeterlidir.
r(α)'ya yansımalı (reflexive) kapanış bağıntısı,
s(α)'ya bakışlı (symmetric) kapanış bağıntısı,
t(α)'ya da geçişli (transitive) kapanış bağıntısı denir.

Teorem:

α yansımalı ise => $r(\alpha) = \alpha$.
α bakışlı ise => $s(\alpha) = \alpha$.
α geçişli ise => $t(\alpha) = \alpha$.

Tanım:

E herhangi bir *A* kümesinde tanımlanan eşitlik bağıntısı (birim bağıntı) olsun.
$E = \{ \langle x, x \rangle \mid x \in A \}, \ \forall R \subseteq A \times A \ (E = R^0)$

Teorem:

$r(\alpha) = R_\alpha \cup E$ (yönlü grafta her düğüme bir tek çevre eklenebilir)

Örnek:

a) $r(R(<)) = R(\leq)$ (<: dar sıra bağıntısı, ≤: genel sıra bağıntısı olarak isimlendirilir.)
b) $r(E) = E$
c) $r(R(\neq)) = A \times A$ evrensel bağıntı

Not: $r(R(\varnothing)) = E$ Boş bağıntının yansımalı kapayanı eşitliktir.

Teorem:

α bakışlı ise => $\alpha = \alpha^{-1}$

Teorem:

$s(\alpha) = \alpha \cup \alpha^{-1}$
Yönlü grafta her yaya bir ters yay eklemek demektir.

Örnek:

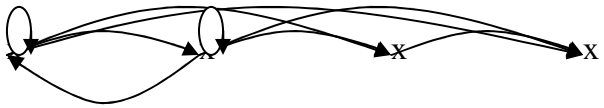
$s(R(<)) = R(\#)$
$s(R(\leq)) = A \times A$
$s(E) = E \wedge s(R(\neq)) = R(\neq)$

Tanım:

Geçişli Kapanış: *D*, α bağıntısının grafı olsun. *t*(α)''nün grafı (*D*')'sü *D*'nin yol olan her iki düğümü arasına bir yay konularak elde edilir.

(*D*)

(*D*')



Teorem:

α, *A*' da bir küme içi ikili bağıntı ise

$t(\alpha) = \bigcup_{i=1}^{\infty} \alpha^i = \alpha \cup \alpha^2 \cup \alpha^3 \cup ...$

Tanıt:

$t(\alpha) \subseteq \bigcup_{i=1}^{\infty} \alpha^i$ ve $\bigcup_{i=1}^{\infty} \alpha^i \subseteq t(\alpha)$ ayrı ayrı tanıtlanmalıdır.

a) Öncelikle:

$\bigcup_{i=1}^{\infty} \alpha^i \subseteq t(\alpha)$ olduğunu, diğer bir deyişle;

$\forall <a, b> \in \alpha^{n+1} => <a, b> \in t(\alpha)$ olduğunu tanıtlayalım.

Tüme varım yöntemi kullanılacaktır.
Tanım gereği $\alpha \subseteq t(\alpha)$
Varsayım gereği: $\alpha^n \subseteq t(\alpha)$
İspatlanacak bölüm:
$\forall <a, b> \in \alpha^{n+1}$ için $\exists c, <a, c> \in \alpha^n \wedge <c, b> \in \alpha$
$<a, c> \in t(\alpha)$ ($<a, c> \in \alpha^n$ ve $\alpha^n \subseteq t(\alpha)$ olduğundan dolayı)
$<c, b> \in t(\alpha)$ ($<c, b> \in \alpha$, $\alpha \subseteq t(\alpha)$ olduğundan dolayı)
$<a, b> \in t(\alpha)$ ($t(\alpha)$'nın geçiş özelliği)
$=> \alpha^{n+1} \subseteq t(\alpha) => \forall i\ (\alpha^i \subseteq t(\alpha))$

$=> \bigcup_{i=1}^{\infty} \alpha^i \subseteq t(\alpha)$

İkinci olarak:

$t(\alpha) \subseteq \bigcup_{i=1}^{\infty} \alpha^i$ olduğunu ispatlayalım

$\bigcup_{i=1}^{\infty} \alpha^i$ geçişli bir bağıntı mıdır? $<a, b>$, $<b, c>$ bu bağıntıdan iki sıralı bir çift olsun.

$<a, b> \in \alpha^s$ ve $(b, c) \in \alpha^t$.

$<a, c> \in \alpha^{s+t}$ olacaktır. $<a, c> \in \bigcup_{i=1}^{\infty} \alpha^i$ dir.

Şu halde $\bigcup_{i=1}^{\infty} \alpha^i$ geçişlidir. Tanıma göre t(α),α'yı içine alan tüm geçişli bağıntıların içinde yer alır. (sırada en küçüğü ya da en önde gelenidir)

$t(\alpha) \subseteq \bigcup_{i=1}^{\infty} \alpha^i$ bulunur.

Örnek:

a) $b = a+1\ \forall a, b \in I$ bağıntısı verilsin. Bu bağıntının geçişli kapanışı R(<) bağıntısıdır.

b) α, A tam sayı kümesi üzerinde R(<) bağıntısı olsun. A kümesinin R(<)'a göre sıralama (sort işlemi) A' da $R(<) = t(\alpha')$ olacak şekilde en küçük (α') bağıntısını bulmak demektir.

Eğer α'nın tanımlandığı A sonlu ise

$t(\alpha) = \bigcup_{i=1}^{(2^n)^2} \alpha^i$ olduğunu biliyoruz.

Teorem: $t(\alpha)$'yı oluşturma yöntemini veren teorem.

α, n elemanlı A kümesi üzerinde tanımlanıyorsa

$t(\alpha) = \bigcup_{i=1}^{n} \alpha^i$

$\forall k\ (k>0 \wedge \alpha^k \subset \bigcup_{i=1}^{n} \alpha^i)$ göstermek yetiyor.

$\forall <x, y> \in \alpha^k \Rightarrow \alpha$' nın D yönlendirilmiş grafında x ile y arasında bir dolaşı mevcuttur anlamını taşır (Dolaşıda düğüm ve ayrıtlardan istenildiği kadar geçilebilir. Yolda her düğümden ancak bir kere geçilebilir. Gezide düğümden istenildiği kadar, ayrıtlardan bir kere geçilebilir.). $<x, y> \in \alpha^k$ ifadesi, x'ten y'ye gitmek için k adım atıldığı anlamına gelir. Bu dolaşıda tüm çevreleri kaldırırsak iki düğüm arasında bir yol elde ederiz.
$<x, y> \in \alpha^i$ burada $0 < i \leq n$ arasında bir değerdir. $i = n$ ancak x ve y düğümlerinin aynı olduğu ve yolun tüm düğümlerden geçtiği durumda olabilir.

Teoremler:

a) α yansımalı ise $s(\alpha)$ ve $t(\alpha)$ da yansımalıdır.
b) α bakışlı ise, $r(\alpha)$ ve $t(\alpha)$ da bakışlıdır.
c) α geçişli ise, $r(\alpha)$ da geçişlidir. Ancak $s(\alpha)$ geçişli olmayabilir.

a) nın tanıtı:

α yansımalı ise $\alpha = E \cup (\alpha')$ ' dır.
$s(\alpha) = E \cup (\alpha') \cup E^{-1} \cup (\alpha')^{-1} = E \cup (\alpha') \cup (\alpha')^{-1} = E \cup r(\alpha')$ yansıma

$t(\alpha) = \bigcup_{i=1}^{n} (E \cup \alpha)^i$ ve $E\alpha = \alpha E = \alpha \wedge E^n = E$ olduğuna göre

$= E \cup (\alpha)' \cup (\alpha')^2 \cup .... \cup (\alpha')^n$ yansımalı bir bağıntıdır.

Teoremler:

a) $rs(\alpha) = sr(\alpha)$
b) $rt(\alpha) = tr(\alpha)$
c) $ts(\alpha) \supset st(\alpha)$

Tanıtlar:

a)

$sr(R) = s(R \cup E) = (R \cup E) \cup (R \cup E)^{-1} = R \cup E \cup R^{-1} \cup E^{-1}$
oysa $E \cup E^{-1} = E \wedge R \cup R^{-1} = s(R)$
$sr(R) = (R \cup R^{-1}) \cup E = s(R) \cup E = \mathrm{rs}(R)$

b)

$tr(R) = t(R \cup E) \wedge rt(R) = t(R) \cup E$
E birim bağıntı $ER = RE \wedge E^n = E$

$$(R \cup E)^n = E \bigcup_{i=1}^{n} R$$

$tr(R) = t(R \cup E) = R \cup E \cup (R \cup E)^2 \cup ... \cup (R \cup E)^n \cup ...$
$= E \cup R \cup R^2 \cup \cup ... \cup R^n \cup ...$
$= E \cup t(R) ...$
$= rt(R)$

c)

$R_1 \supseteq R_2 \Rightarrow s(R_1) \supseteq s(R_2) \wedge t(R_1) \supseteq t(R_2)$
$s(R) \supseteq R$ olduğuna göre
$ts(R) \supseteq t(R) \wedge st(R) \supset st(R)$
$s(R)$ bakışlı ise $t(s(R))$ de bakışlıdır. Şu halde $st(s(R)) = ts(R) \Rightarrow ts(R) \supseteq st(R)$ bulunmuş olur.

Tanım:

α, A üzerinde bir bağıntı ise $\alpha^+ = t(\alpha)$ ve $\alpha^* = tr(\alpha)$ olarak gösterilir. Bağıntıların "+" ve "*" işlemleri formel dillerde, hesap modellerinde ve derleyici tasarımında kullanılır.

Örnek:

$P = \{ P_1, P_2, ..., P_n\}$ bir program kitaplığında program ve alt programlar kümesi olsun. Bu kümede $P_i \Rightarrow P_j$ ikili bağıntısı P_i program P_j'yi çağırır anlamında kullanılsın. Bu durumda $\Rightarrow^+$ bağıntısı bir programın icrası sırasında çağrılan bir tüm alt programları kümesini ortaya çıkarmaya yarar. Örneğin ($P_i \Rightarrow^+ P_j$) P_i'nin icrası sırasında P_j çağrılır demektir.
Oysa $\Rightarrow^*$ bir programın yürütülmesi sırasında herhangi bir anda etkin olacak olan program ya da alt programlar kümesini ortaya çıkarmaya yarar.
$P_i \Rightarrow^* P_j$, P_i'nin icrası sırasında P_j uyarılıyorsa $\forall i$ ($P_i \Rightarrow^* P_i$) her zaman doğrudur ancak $P_i \Rightarrow^+ P_i$ ise $\Rightarrow P_i$ rekürsiftir (kendi kendini çağırıyordur).

2.4 Diller ve gramerler

S: Sözcükler
S^*: Tüm sözcük sekansları (simge katarları) olsun.
$L \subset S^*$ kurallarına göre kurulmuş cümleler kümesi olsun. L arasından anlamlı olanları seçmek önemli bir problem olarak karşımıza çıkar.

Tanım:

Dil: $\sum$ alfabesi ise, $\sum^*$'ın bir alt kümesidir. $\sum^*$, alfabe ile oluşturulabilen tüm sözcüklerin kümesidir. Gramer bunlardan nasıl bir alt küme seçileceğinin kurallarıdır.

Örnek:

Tam sayılar, "+", "-", "x", "/" ile "(", ")" simge kümesi ile aritmetik ifadeler elde edilebilir. Kurallara göre yazılmış tüm aritmetik ifadeler anlamlıdır (sıfıra bölümler hariç).
((2-1)/3)+4x6) anlamlı bir ifadeyi
2+(3/(5-(10/2))) anlamsız bir ifadeyi gösterir (sıfıra bölüm nedeniyle)

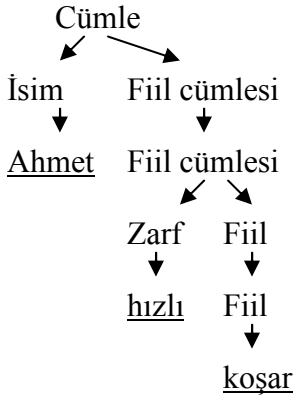
Tanım:

Syntax (sentaks): Cümle yapısının belirlenmesi
Semantics (semantik): Cümlenin anlamının belirlenmesi
İlk yaklaşımda bilgisayar dillerinde yalnızca syntax ile uğraşılır yani "cümlenin yapısını kurma kurallarına" yani gramer (dilbilgisine) ilgi duyulur.
Cümle yapısı grameri (phrase structure grammar) G, bir dörtlü olarak tanımlanır (V, S, v_0, $|\rightarrow$). Burada:
V: sonlu bir küme
S: sözcükler ($S \subset V$) ya da uç (terminal) simgeler kümesi (yazılı bir metinde her sözcük terminaldir).
$N = V - S$ yani $V = S \cup N \wedge S \cap N = \varnothing$ olan, uç olmayan ara simgeler kümesi.
$v_0 \in N$ uç olmayanlar arasından başlangıç simgesi
$|\rightarrow$, V^*'dan küme içi bağıntı ($V^n \rightarrow V^m$'ye). Bu bağıntıya yerini alma (replacement) türünden bir bağıntı deniyor. Burada yerini alma, V^*'dan bir "w" katarının, tek başına, ya da başka bir katarın alt katarı olduğu durumlarda yerine "w"' katarının kurulabileceği anlamı çıkar.
$|\rightarrow$ bağıntı kümesinin her elemanına G' de bir üretim kuralı (production rule) adı verilir. Üretim kuralının sağ ve sol yanlarından söz edilir.
$|\rightarrow$, G'nin bir üretim bağıntısı olarak da tanınır (production relation).
Bu durumda v_0 (start symbol), üretim kurallarının başladığı ilk sözcük, terminal olmayan başlangıç simgedir.
Üretim kurallarından doğrudan türetme (direct derivability) bağıntısı "$\Rightarrow$" ile tanımlanır.
$w \rightarrow w'$ üretim kuralından $x \Rightarrow y \wedge x = l.w.z \wedge y = l.w'.r$ ve nihayet $\Rightarrow^*$ (yansımalı geçişli kapanışı) elde edilecektir. Bu bağıntıya erişilebilirlik (reachability) adı verilir. S^*' dan bir cümlenin sentaks bakımından doğru bir cümle olması $v_0 \Rightarrow^* \sigma$ şeklinde ifade edilir. Bu durumda $\sigma \in L$ denebilir.

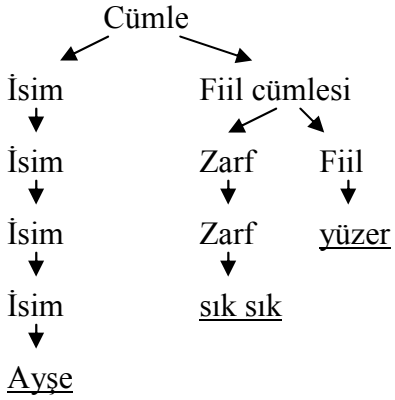
Örnek 1:

S = {Ahmet, Ayşe, koşar, yüzer, hızlı, sık sık, uzun}
v_0 = cümle
N = {cümle, isim, fiil cümlesi, fiil, zarf}
Üretim kuralları:
Cümle $|\rightarrow$ isim fiil cümlesi
İsim $|\rightarrow$ Ahmet

İsim |→ Ayşe
Fiil cümlesi → zarf fiil
Zarf → hızlı
Zarf → sık sık
Zarf → uzun
Fiil → yüzer
Fiil → koşar



Sağdan kanonik türetim ağacı (parse tree)

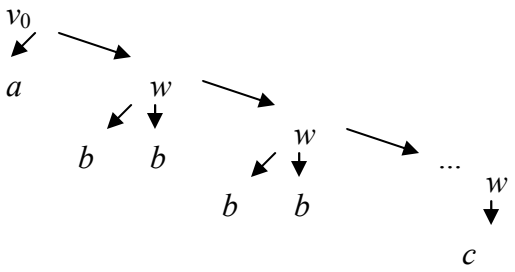


Soldan kanonik türetim ağacı (parse tree)

G gramerinin üretim kurallarını kullanarak elde edilen tüm sentaksı uygun cümleler, *G* gramerinin dili olarak tanımlanacaktır.
$L(G) = \{\sigma \mid v_0 =>^* \sigma\}$

Örnek 2:

$V = \{ v_0, w, a, b, c\}$ $S = \{a, b, c\}$ $|\rightarrow \equiv$
$G = (V, S, v_0, |\rightarrow\}$
$\{v_0 \rightarrow aw \mid w \rightarrow bbw \mid w \rightarrow c\}$



Sonuç:

$L(G) = \{a\}.\{bb\}^*.\{c\}$
$= a(bb)^*c$ (kompakt bir form halinde verilmiş ifade)
$= \{a(bb)^n c \mid n \in N\}$

Örnek 3:

$V = \{v_0, w, a, b, c\}$, $S = \{a, b, c\}$
$|\rightarrow \equiv \{v_0 \rightarrow av_0b \mid v_0b \rightarrow bw \mid abw \rightarrow c\}$
$v_0 => av_0b$
$av_0b => a(av_0b)b => a^n v_0 b^n$
$a^n(v_0b)b^{n-1} => a^n bwb^{n-1}$

$a^{n-1}(abw)b^{n-1} => a^{n-1}cb^{n-1}$

Genelde:
$L(G) = \{ a^m cb^m \mid m \in N\}$

Ağacı yok.

Örnek 4:

$N = \{A, B, C, v_0\}$; $S = \{a, b, c\}$
$v_0 \rightarrow A$
$A \rightarrow aABC$
$A \rightarrow abC$
$CB \rightarrow BC$, $bB \rightarrow bb$, $bC \rightarrow bc$, $cC \rightarrow cc$

v_0
A
$a\underline{A}BC$
$aaABCBC$
$aaa\ b\underline{CB}CBC$
$aaa\ b\underline{BCCB}C$
$aaa\ bb\underline{CB}CC$
$aaa\ bb\underline{BC}CC$
$aaa\ bbb\ CCC$
$aaa\ bbb\ cCC$
$aaa\ bbb\ ccC$
$aaa\ bbb\ ccc$
$a^3\ \ b^3\ \ c^3$
$L(G) = \{a^k b^k c^k \mid k \in N^+\}$

2.5 Dilbilgisi, Chomsky sınıflandırması

$G = (V, S, v_0, \mid\rightarrow)$ ise

Tip 0: Kısıtlama yok. (Turing Makinaları). 3.Örnek.
Tip 1: $w_1 \rightarrow w_2$ de uzunluk $(w_1) \leq$ uzunluk (w_2) (Doğrusal sınırlandırılmış otomata). 4.Örnek
Tip 2: Üretim kuralının sol yanında tek terminal olmayan bir simge bulunur. Sağ yan bir ya da daha fazla sayıda simge içerir. (Yığın yapılı otomata). 1.Örnek.
Tip 3: Sol yan terminal olmayan tek simge, sağda bir kaç terminal sınıfı ya da bir kaç terminal simge ve en sağda tek bir terminal olmayan simge bulunur. (Sonlu durumlu otomata). 2.Örnek

Bu kurallara istenirse ($v_0 \rightarrow \Lambda$) üretim kuralı da eklenir. Böylece boş cümlenin (hiç bir sözcüğü olmayan cümle) dilden olduğu kabul edilir.

(Tip 3) $\subseteq$ (Tip 2) $\subseteq$ (Tip 1) $\subseteq$ (Tip 0)

Ancak Tip 2 ve Tip 3'ün türeme ağaçları bulunabilir. İkinci (ve üçüncü) tipten gramerlere bağlamdan bağımsız gramerler denir (context-free grammar).
Buna karşılık $lwr \rightarrow lw'r$ tipi kuralları olan gramerlere ise bağlama duyarlı "context sensitive" gramer adı verilir.
Tip 3'ten gramerlere ise düzenli gramer adı verilir (regular grammar).
Diller de kendilerini oluşturan gramerlerle aynı tip numarası ile anılırlar. Ancak 2. Tipten bir dili üreten 2.tipten bir gramerden başka farklı tipten de gramerler bulunabilir.

BNF notasyonu: *Backeus – Naur Form* (Backeus:Fortran, Naur: Algol ve Cobol'un oluşturucuları)

a) $|\rightarrow$ üretim kuralı ::=
b) < > terminal olmayan simgeler tek köşeli parantezler arasına
c) tekrarların ayrılması
d) terminal simgeler oldukları gibi

Örnek 1 (tekrar – 2.Tip):
<cümle>::= <isim><fiil cümlesi>
<isim>::= Ahmet | Ayşe
<fiil cümlesi>::=<zarf><fiil>
<zarf>::= hızlı | sık sık | uzun
<fiil>::= yüzer | koşar

Örnek 2 (tekrar – 3.Tip)
$<v_0>$::= *a*<*w*>
<*w*>::= *bb*<*w*> | *c*

Tanım:

a) $w \rightarrow bbw$ üretim kurallarına öz çağırmalı (recursive) kural denir.
b) Öz çağırmalı üretim kuralında non-terminal simge en sağda olursa (sağ tarafın en sağında) "normal" üretim kuralı olarak nitelenir.

Örnek 3 (tekrar – 0.Tip)
$<v_0>$::= *a*$<v_0>$*b*
$<v_0>$*b*::= *b*<*w*>
ab<*w*>::= *c*

Örnek 4 (tekrar – 1.Tip)
$<v_0>$::= <*A*>
<*A*>::= *a*<*A*><*B*><*C*> | *ab* <*C*>
<C><B>::= <B><C>
b<*B*>::= *bb*
b<*C*>::= *bc*
c<*C*>::= *cc*

2.6 Düzenli İfadeler

Σ alfabesi üzerinde düzenli ifade tümevarımla tanımlar:

1) Λ ve Σ'nın her elemanı bir düzenli ifadedir
2) α ve β düzenli ifadeler ise (örnek α = atlı, β = karınca) $\alpha\beta$ (konkatenasyon) da düzenli bir ifadedir.
3) α ve β düzenli ifadeler ise $\alpha \vee \beta$ da düzenli ifadedir.
4) α bir düzenli ifade ise α^* da bir düzenli ifadedir. $\alpha^* = \Lambda \vee \alpha \vee \alpha^2 \vee \ldots$

Düzenli ifadeler ile ona ilişkin diller arasında şu şekilde bir ilişki kurulur:

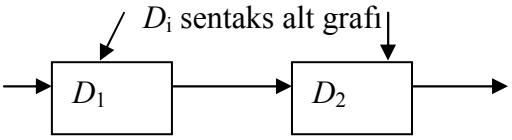
1) $L(\Lambda) = \{\Lambda\}$; $L(a) = \{a\}$; $\forall a \in \Sigma$
2) α ve β düzenli ifadeler ise $L(\alpha\beta) = L(\alpha)L(\beta)$
3) α ve β düzenli ifadeler ise $L(\alpha \vee \beta) = L(\alpha) \cup L(\beta)$
4) α bir düzenli ifade ise $L(\alpha^*) = L^*(\alpha)$

Düzenli ifadeler ve onları temsil eden dil genelde karıştırılır. Dil bir küme olarak tanımlanır.

Düzenli gramerlerin yukarıda tanımlanan şekilde düzenli ifadeler ürettiği tanıtlanır. Düzenli dil, grameri düzenli olan bir dildir. Dil ile düzenli ifade arasında bir izomorfizm vardır. Yapısal denklik demek olan izomorfizm mühendislikte modellemede sıklıkla kullanılır.

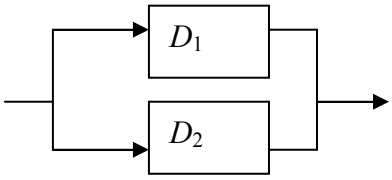
Teorem:

S kümesi üzerinde tanımlanmış bir dil olsun. $L \subseteq S^*$. *L* dilinin düzenli (regular) olması demek *L*'nin düzenli gramer $G = (V, S, v_0, |\rightarrow)$'e uyması gerekir. Yani $L = L(G)$ olmalıdır. Bu dilin bir ifadesi ise düzenli ifadedir.PASCAL dilinde düzenli ifadelerin kullanılan sentaks grafı yardımı ile yazılabilir. Aşağıdaki ifadeler 3.tip diller için geçerlidir.

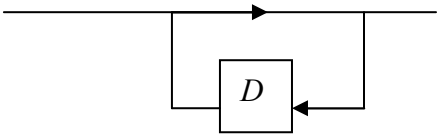


$\alpha_1 \alpha_2$

α_i: D_i sentaks alt grafına ilişkin düzenli ifade



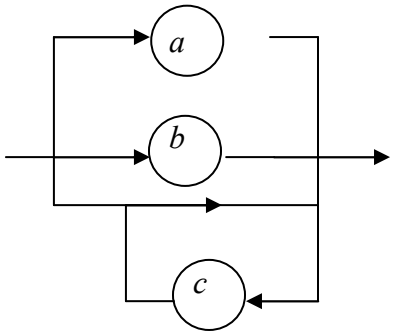
α_1 V α_2



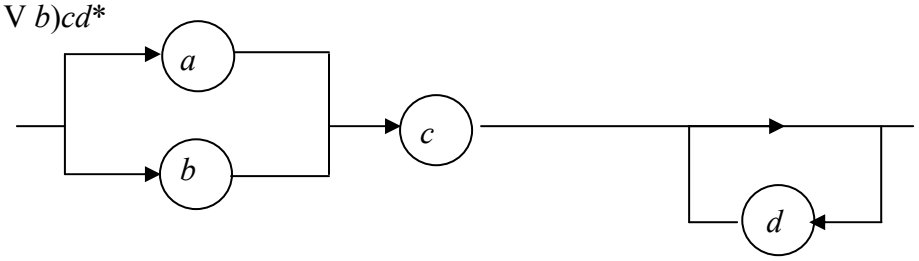
α^*

Örnek:

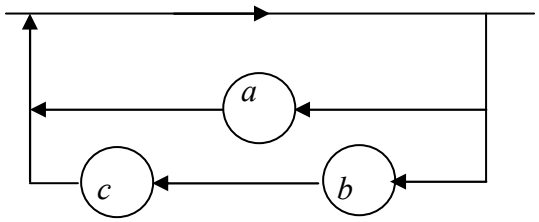
a V *b* V *c**



(*a* V *b*)*cd**



(*a* V *bc*)*



BÖLÜM 3

Otomatlar

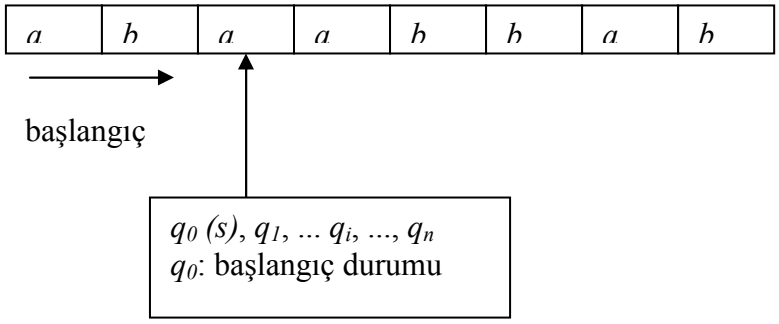
3.1 Determinist sonlu otomat (DFA) ve düzenli ifadelerin tanınması

Sözlük:

Giriş şeridi, durumlar, sonlu denetim, okuma kafası, başlangıç durumu, kabul edilen ya da son durumlar, tanınan ya da kabul edilen dil.

Tanıtım:

Giriş şeridi: Şeridin tüm gözeleri soldan sağa sırayla okunduğunda, son göze okununca otomatın seçeceği durum kabul edilebilen durumlar içinde ise, şeritte yazılı simge katarı otomat tarafından kabul edilen dilin bir katarıdır.



Şekil 3.1 Sonlu durumlu otomat

Tanım:

a) K sonlu durum kümesi $q \in K$

b) $\sum$ giriş alfabesi

c) $s \in K$ başlangıç durumu

d) $F \subseteq K$ kabul edilen ya da son durumlar

> K -> Z = { 0, 1 }; çıkış simgeleri kabul ve red olan Moore modeli. Durumlara 1/0 değeri verilirse, kabul edilenlere 1, edilmeyenlere 0 verilir. En sonda çıkış 1 ise katar kabul edilmiş demektir.

e) δ: K x $\sum$ -> K geçiş fonksiyonu

$M = (K, \sum, \delta, s, F)$

M makinası q durumunda şeritten σ'yı okudu. $\delta (q, \sigma) = q' \in K$ durumuna geçti.

$\delta_\sigma(q) = q' \in K$

$\forall \sigma \in \sum \exists \delta_\sigma: K \rightarrow K \wedge \delta = \{ \delta_\sigma \mid \sigma \in \sum \}$

Tanım:

Konfigürasyon (Konuşlanım)

$(q, w) \in K$ x $\sum^*$

$|-_M$ ikili bağlantı, iki konfigürasyon arasında tanımlanır. (q, w) ile (q', w')'nun $|-_M$ bağıntı kümesi içinde olması

a) $w = \sigma w' \wedge \sigma \in \Sigma$

b) $\delta (q, \sigma) = q'$ koşulları ile tanımlanır.

Buna (q, w)'nin bir adımda (q', w') üretmesi denir. Aslında $|-_M$; K x Σ^+' dan K x Σ^*' a bir fonksiyondur ve (q, Λ) şeklinde bir konfigürasyonun otomatın tüm şeridi tüketmiş olması demektir (okuma yaptıkça w kısalır).

Tanım:

Kabul edilen dil (language recognizer): $|-_M$ bağıntısının yansımalı (reflexive) ve geçişli (transitive) kapanışı (closure), $|-_M$ * bağıntısıdır. Bu durumda $(q, w) |-_M * (q', w')$, (q, w) konfigürasyonundan (q', w') konfigürasyonu üretildi denebilir. Bu bir çok adımda olabilir [ing: (q, w) yields (q', w') after some number of steps].

$w \in \Sigma^*$ katarının bir otomat tarafından kabul edilmesi $q \in F$ olmak üzere, $(s, w) |-_M * (q,\Lambda)$ olması ile tanımlanır. $L(M) = \{ w \in \Sigma^* \mid (s, w) |-_M * (q_i, \Lambda) \wedge q_i \in F \}$ şeklinde de tanımlanır.

Örnek 1:

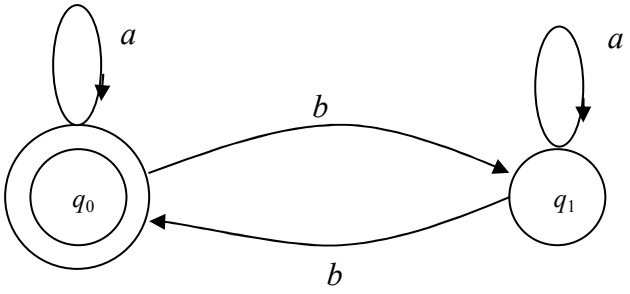
$K = \{ q_0, q_1 \}$

$\Sigma = \{ a, b \}$

$s = q_0$

$F = \{ q_0 \}$ (kabul edilen durumlar)

q	σ	$\delta (q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0



$(q_0, aabba) |-_M (q_0, abba)$
$(q_0, abba) |-_M (q_0, bba)$
$(q_0, bba) |-_M (q_1, ba)$
$(q_1, ba) |-_M (q_0, a)$
$(q_0, a) |-_M (q_0, \Lambda)$

$L(M) = (a \vee ba^*b)^*$ (sezgisel olarak yazdık, sistematiğini sonra vereceğiz)

Gramerini yazmak istersek:

$V = K \cup \Sigma$

$I = \Sigma = \{a, b\}$

$s = q_0 = v_0$

$<q_0> ::= \Lambda \mid a<q_0> \mid b<q_1> \mid a$

$<q_1> ::= b<q_0> \mid a<q_1> \mid b$

Burada verilmiş olan δ kavramı ister durum tablosu, ister durum diyagramı ya da gramer ile açıklanabilir.

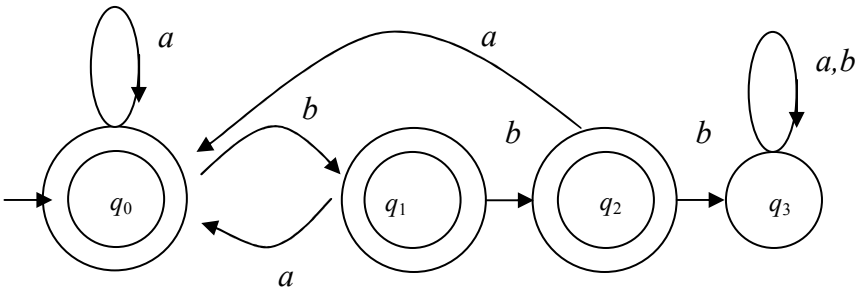
Örnek 2:

M otomatı $L(M)$ dilini kabul etmeli.

$L(M) = \{ w \mid w \in \{ a, b \}^* \wedge w$ katarı peşpeşe 3 b kabul etmemeli $\}$

$K = \{ q_0, q_1, q_2, q_3 \}, \Sigma = \{ a, b \}, \ s = q_0, \ F = \{ q_0, q_1, q_2 \}$

q	σ	δ (q, σ)
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_2
q_2	a	q_0
q_2	b	q_3
q_3	a	q_3
q_3	b	q_3

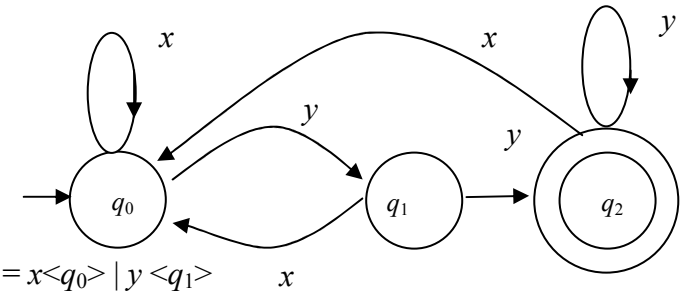


Durum diyagramından da görülebileceği üzere q_3 ölü durumdur. Bu duruma geçince makina kapana kısılmış olur. Bu durumdan çıkış yoktur.

$L(M) = [(\Lambda \vee b \vee bb)a]^*(\Lambda \vee b \vee bb)$

Örnek 3:

$\Sigma = \{x,y\}, K = \{ q_0, q_1, q_2\}, s = q_0 = v_0, F = \{q_2\}$



$<q_0> ::= x<q_0> \mid y <q_1>$ $\quad x$

$<q_1> ::= y<q_2> \mid y \mid x<q_0>$

$<q_2> ::= y \mid y<q_2> \mid x<q_0>$

$L(M) = ((x \text{ V } yx)^*yy^+x)^*(x \text{ V } yx)^*yy^+$

$L(M)=((\ \Lambda \text{ V } y \text{ V } yy^+)x)^*yy^+=(y^*x)^*yy+$

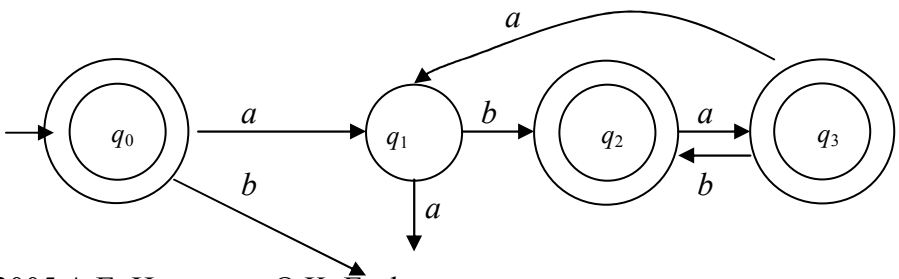
$L(M)=(x \text{ V } yx \text{ V } yy^+x)^*yyy^*$

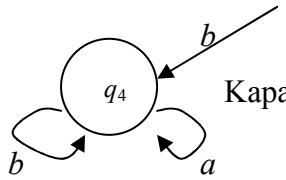
3.2 Determinist olmayan otomat (NFA) ve düzenli ifadelerin tanınması

Bir durumda, belirli bir girişte birden fazla düğüme yol varsa otomata deterministlik özelliğini kaybeder.

Örnek 3.2.1:

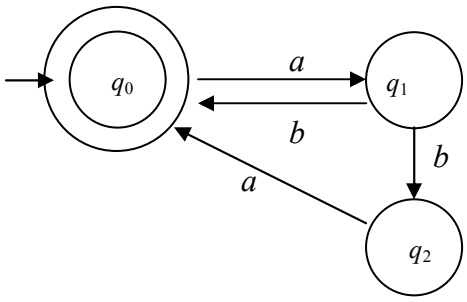
L = (ab ∨ aba)* dilini kabul eden determinist otomat durum diyagramı aşağıda verilmiş olsun:



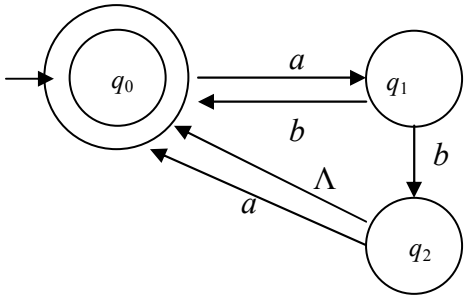


Kapanış durumu

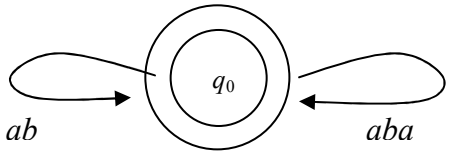
Aynı dili kabul eden determinıst olmayan otomatın üç farklı durum diyagramı aşağıda verilmiş olsun:



ya da



ya da:



Tanım:

Determinist olmayan otomat için sezgisel tanımlama yapılacak olursa, bu otomatın giriş katarını kabul etmesi için başlangıç durumu ile kabul edilir (son) durumların biri arasında katarın sırayla soldan sağa simgeleriyle etiketlenmiş sıralı ayrıtlardan oluşan bir yolun (ya da dolaşının) var olması gerekir.

Determinist olmayan otomatların matematiksel tanımı:

$M = (K, \sum, \Delta, s, F)$

K durumları, $\sum$ alfabeyi, $s \in K$ başlangıç durumunu, $F \subseteq K$ son (kabul edilebilir) durumları ve Δ geçiş bağıntıları sonlu kümesini ifade etsin.

$\Delta \subseteq K \text{ x } \sum^* \text{ x } K \; (q, u, b) \in \Delta \wedge u \in \sum^*)$

Konfigürasyon tanımına göre:

K x Σ^* 'ın bir elemanıdır, bir çifttir (q, w gibi) .

$|-_M$ bağıntısı (bir adımda üretim):

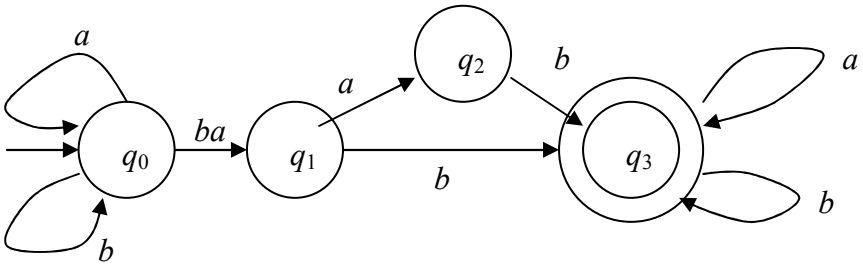
$(q, w) |-_M (q', w') \Rightarrow \exists\, u \in \Sigma^* (w = uw' \wedge (q, u, q') \in \Delta)$ yazılabilir.

$|-_M^*$, $|-_M$' nin yansımalı ve geçişli bir kapanışıdır. Deterministik olmayan otomatın kabul ettiği dil $L(M) = \{w \mid (s, w) \; |-_M^* (q, \Lambda) \wedge q \in F\}$

Örnek 3.2.2:

İçinde *bab*, ya da *baab* alt katarları barındıran bir dile ilişkin deterministik olmayan bir otomat bulun

$K = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{ a, b \}$
$s = q_0$
$F = \{ q_3 \}$
$\Delta = \{(q_0, a, q_0), (q_0, b, q_0), (q_0, ba, q_1), (q_1, b, q_3), (q_1, a, q_2), (q_2, b, q_3), (q_3, a, q_3), (q_3, b, q_3)\}$
$M = (K, \Sigma, \Delta, s, F)$
$< q_0> :: = a<q_0> \mid b<q_0> \mid ba<q_1>$
$<q_1> :: = b <q_3> \mid b \mid a <q_2>$
$<q_2> :: = b<q_3> \mid b$
$<q_3> :: = a \mid b \mid a <q_3> \mid b <q_3>$



Yollardan biri aşağıdaki adım adım üretimlerle oluşturulabilir:

$(q_0, aaabbbaabab) |\rightarrow (q_0, aabbbaabab) |\rightarrow (q_0, abbbaabab) |\rightarrow (q_0, bbbaabab)$
$|\rightarrow (q_0, bbaabab) |\rightarrow (q_0, baabab) |\rightarrow (q_1, abab) |\rightarrow (q_2, bab) |\rightarrow (q_3, ab) |\rightarrow (q_3, b)$
$|\rightarrow (q_3, \Lambda)$

Uyarı:

Deterministik otomata $\Delta \subseteq K$ x Σ^* x K bağıntısı K x $\Sigma \rightarrow K$' ya bir fonksiyon haline dönüşür ve (q, u, q') üçlüsünde $| u | = 1 \wedge \forall\, q \in K \wedge \forall\, u \in \Sigma\, \exists!\, q' \in K$

Lemma:

$M = (K, \Sigma, \Delta, s, F) \wedge q, r \in k \wedge x, y \in \Sigma^*$
$(\exists\, p \in K \wedge (q, x) |-_M^* (p, \Lambda) \wedge (p, y) |-_M^* (r, \Lambda) \Rightarrow (q, xy) |-_M^* (r, \Lambda)$

Tanım:

Düzenli gramer: 3. tipten kuralları olan gramer (ing: regular grammer)
Düzenli dil: Düzenli gramerlerin kabul ettiği dil (ing: regular languages)
Düzenli ifade: $\varnothing$, $\{ \Lambda\}$, $\{ a \mid a \in \Sigma \}$, $A \vee B$, $A.B$, A^* (ing: regular expressions)
Düzenli ifadeler kümesi: düzenli küme (ing: regular set)

Düzenli gramerler deterministik olmayan sonlu otomatlar ile temsil edilebilirler.

a) Terminal olmayan simgelere durumlar iliştirilir.
b) Başlangıç durumu başlangıç simgesi olsun.
c) Kabul edilen durumlar, terminal simgelerle biten kurallara karşı gelir.
d) Eğer Λ kabul ediliyorsa, başlangıç durumu kabul edilen durumlar arasındadır.

Sonlu otomatlar tarafından kabul edilen diller (Düzenli diller) birleşim, bitiştirme ve yıldız işlemlerine kapalıdır.

Teorem:

Her düzenli dil bir sonlu otomat tarafından kabul edilir ve her sonlu otomatın tanımladığı bir düzenli dil vardır (Kleen teoremi)

$M = (K, \Sigma, \Delta, s_0, F) \Leftrightarrow G = (V, S, v_0, |->), L = L(G)$ 3.tipten bir gramer

$L = L(M)$

Eşleştirmeyi yapmak için:

$K = (V - S) \wedge F \subseteq V - S$

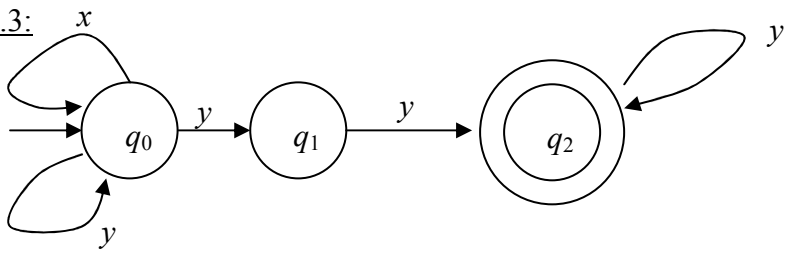
$\Sigma = S$

$s_0 = v_0$

$\Delta = \{(A, w, B) : A |-> wB \in |-> \wedge (A, B \in V - S) \wedge w \in S^*\} \cup \{ (A, w, f_i)$:

$A |-> w \in |-> \wedge A \in V\text{-}\ S \wedge f_i \in F \wedge w \in S^*\}$

Örnek 3.2.3:



$<q_0> ::= x<q_0> \mid y<q_0> \mid y<q_1>$
$<q_1> ::= y \mid y <q_2>$
$<q_2> ::= y<q_2> \mid y$

3.3 DFA ile NFA eşdeğerliği

Teorem:

Her deterministik olmayan otomat için, ona eşdeğerli deterministik bir otomat bulunabilir.

Tanıt için yol gösterme:

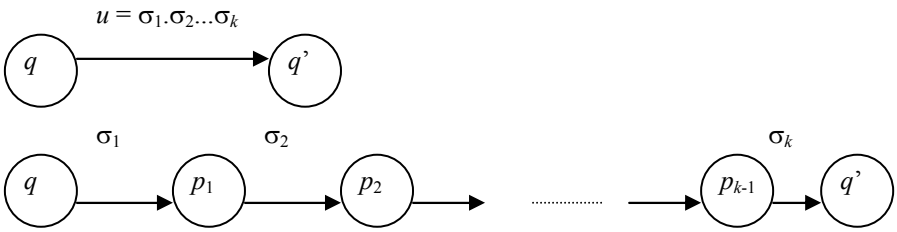
$M = (K, \Sigma, \Delta, s, F)$

Amaç:

a) $(q, u, q') \in \Delta$ arasında $u = \Lambda$' ler ve $| u | > 1$'ler olmamalı
b) Her durumda, her simge için giriş tanımlanmalı
c) Her konfigürasyonda birden fazla geçiş olmamalı.

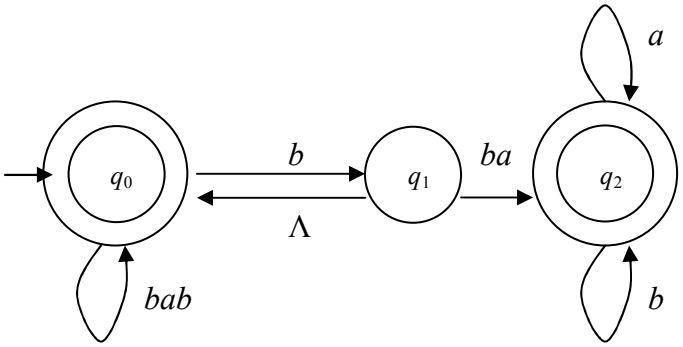
1. aşama:

Δ içinde (q, u, q') olup, buna rağmen $| u | > 1$ olan bağıntıları ortadan kaldırmak için ara durumlar oluşturulur.

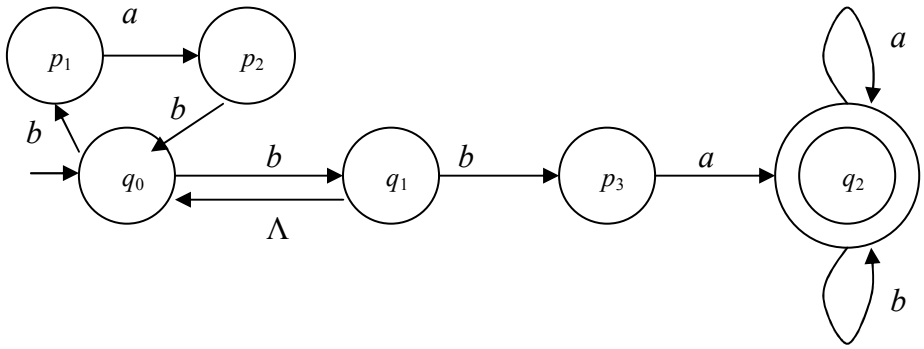


Bu sayede Δ kümesi zenginleşir. (q, u, q') yerine (q, σ_1, p_1), (p_1, $\sigma 2$, p_2), ..., (p_{k-1}, σ_k, q') üçlüleri yeni Δ' oluşturur. Böylece oluşan yeni makinaya:

$M' = (K', \sum, \Delta', s', F'), F' \equiv F$ ve $s' = s$ olarak bakılabilir.



durum diyagramı aşağıdaki diyagram ile özdeştir:



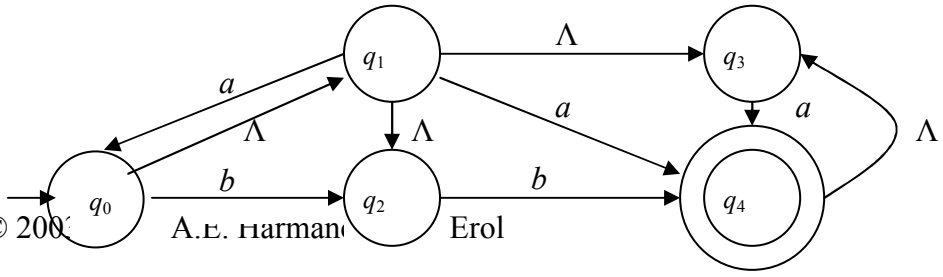
2. Aşama:

Tanım:

$E(q) = \{ p \in K' \mid (q, \Lambda) \vdash_{M'}^* (p, \Lambda)\}$ ya da
$E(q) = \{ p \in K' \mid (q, w) \vdash_{M'}^* (p, w)\}$

Hatırlatma: ($\vdash_{M'}^*$ bir ya da birkaç adımda demektir)

Örnek 3.2.4:



$E(q_0) = \{ q_0, q_1, q_2, q_3 \}$
$E(q_1) = \{ q_1, q_2, q_3 \}$
$E(q_2) = \{ q_2 \}$
$E(q_3) = \{ q_3 \}$
$E(q_4) = \{ q_3, q_4 \}$

Determinist eşdeğer makinanın oluşturulması:
$M'' = (K'', \Sigma, \delta'', F'')$
$K'' = P(K') = 2^{K'}$
$s'' = E(s')$ (s'' = Başlangıçtan $\wedge$ ile gidilebilen durumlar kümesi)
$F'' = \{ Q \subseteq K' \mid Q \cap F' \neq \varnothing \}$

Durumlar: K' nün her alt kümesi bir durumdur.
Final durumlar: K' içinde (F') nün bir elemanı olan her alt kümesi.

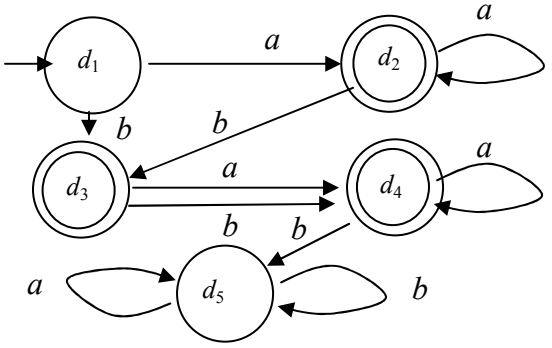
<u>δ'' tanımı:</u>

$\forall Q \subseteq K' \wedge \forall \sigma \in \Sigma$
$\delta''(Q, \sigma) = \cup_p \{ E(p) \mid \forall q \in Q \wedge \forall p \in K' \wedge \forall (q, \sigma, p) \in \Delta' \}$
$= \cup_p E(p) \wedge p \in \{ p \in K' \mid q \in Q \wedge \exists (q, \sigma, p) \in \Delta'' \}$
(Açıklama: $\cup_p$ bütün p'ler için birleşim işlemini gösterir)
Boş katar dışında olabilecek tüm üçlüleri yazarsak:
a ile gidilenler: (q_1, a, q_0), (q_1, a, q_4), (q_3, a, q_4)
b ile gidilenler: (q_0, b, q_2), (q_2, b, q_4)

δ'nın toplam 5 adet elemanı olur. Bunlardan hareketle:

$s'' = E(q_0) = \{ q_0, q_1, q_2, q_3 \}$
$\delta''(s'', a) = E(q_0) \cup E(q_4) = \{ q_0, q_1, q_2, q_3, q_4 \}$
$\delta''(s'', b) = E(q_2) \cup E(q_4) = \{ q_2, q_3, q_4 \}$
$\delta''(\{ q_0, q_1, q_2, q_3, q_4 \}, a\}) = \{ q_0, q_1, q_2, q_3, q_4 \}$
$\delta''(\{ q_0, q_1, q_2, q_3, q_4 \}, b\}) = \{ q_2, q_3, q_4 \}$
$\delta''(\{ q_2, q_3, q_4 \}, a\}) = E(q_4) = \{ q_3, q_4 \}$
$\delta''(\{ q_2, q_3, q_4 \}, b\}) = E(q_4) = \{ q_3, q_4 \}$
$\delta''(\{ q_3, q_4 \}, a\}) = E(q_4) = \{ q_3, q_4 \}$
$\delta''(\{ q_3, q_4 \}, b\}) = \varnothing$
$\delta''(\varnothing, a) = \delta(\varnothing, b) = \varnothing$

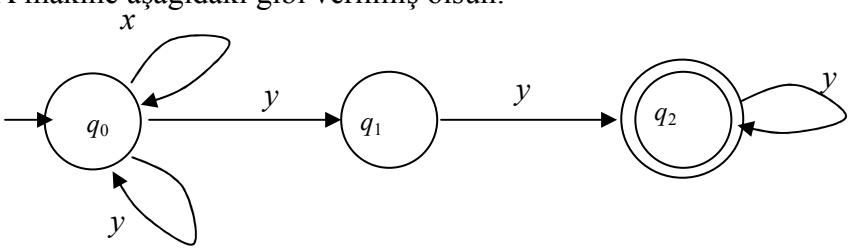
DFA Eşdeğeri makinanın durum diyagramını çizecek olursak:



$d_1 = \{q_0, q_1, q_2, q_3\}$
$d_2 = \{q_0, q_1, q_2, q_3, q_4\}$
$d_3 = \{q_2, q_3, q_4\}$
$d_4 = \{q_3, q_4\}$
$d_5 = \varnothing$

Örnek 3.2.5:

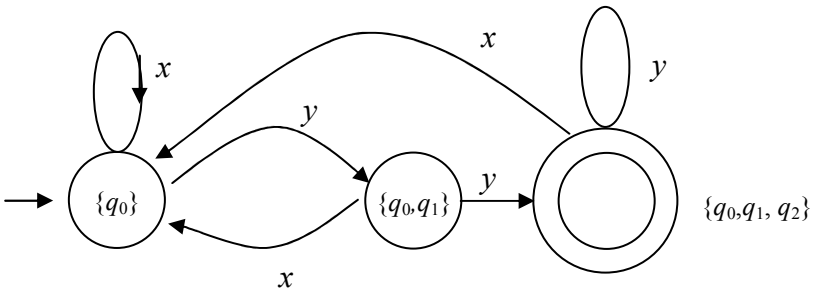
NFA makine aşağıdaki gibi verilmiş olsun:



ve bu makinaya ait dil ise:
$L(M) = (x \vee y)^* yy^+$ olsun. DFA eşdeğeri makinayı bulunuz.

$E(q_0) = \{ q_0 \}$
$E(q_1) = \{ q_1 \}$
$E(q_2) = \{ q_2 \}$

$\Delta' = \{(q_0, x, q_0), (q_0, y, q_0), (q_0, y, q_1), (q_1, y, q_2), (q_2, y, q_2)\}$
$s'' = E(q_0) = \{q_0\}$
$\delta(s'', x) = E(q_0) = \{q_0\}$
$\delta(s'', y) = E(q_0) \cup E(q_1) = \{q_0, q_1\}$
$\delta(\{q_1, q_0\}, x) = E(q_0) = \{q_0\}$
$\delta(\{q_1, q_0\}, y) = E(q_0) \cup E(q_1) \cup E(q_2) = \{q_0, q_1, q_2\}$
$\delta(\{q_0, q_1, q_2\}, x) = E(q_0) = \{q_0\}$
$\delta(\{q_0, q_1, q_2\}, y) = E(q_0) \cup E(q_1) \cup E(q_2) = \{q_0, q_1, q_2\}$



DFA eşdeğeri makine ve buna ait dil (daha önceki örneklerde yapılmıştı):

$L(M) = ((x \vee yx)^* yy^+ x)^* (x \vee yx)^* yy^+$ Buradan yandaki ifade ile yukarıda verilmiş olan NFA dili ifadesinin birbirine denk oldukları görülebilir. Ancak bunu ancak yukarıdaki grafikleri çizdikten sonra anlayabiliriz.

Teorem:

Sonlu otomatlar tarafından kabul edilen düzenli diller sınıfı aşağıdaki işlemlere kapalıdır:

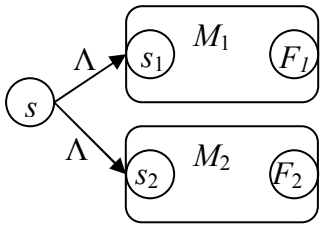
a) Birleşim
b) Bitiştirme
c) Kleen yıldızı

a) Birleşim:

$M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1) \leftarrow L(M_1)$ (Determinist değil)

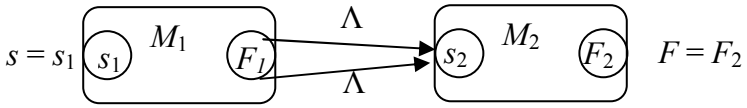
$M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2) \leftarrow L(M_2)$ (Determinist değil)
$M = (K, \Sigma, \Delta, s, F) \leftarrow L(M_1) \cup L(M_2)$ (Determinist değil)

$K = K_1 \cup K_2 \cup \{s\}$
$F = F_1 \cup F_2$
$\Delta = \Delta_1 \cup \Delta_2 \cup \{(s, \Lambda, s_1), (s, \Lambda, s_2)\}$



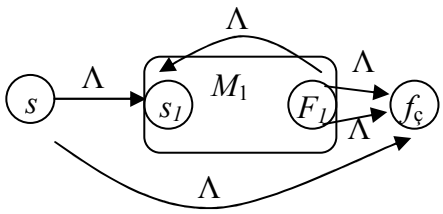
b) Bitiştirme (Determinist değil):

$L(M_1).L(M_2) = L(M)$
$K = K_1 \cup K_2$
$s = s_1$
$F = F_2$
$\Delta = \Delta_1 \cup \Delta_2 \cup (F_1 \text{ x } \{\Lambda\} \text{ x } \{s_2\})$



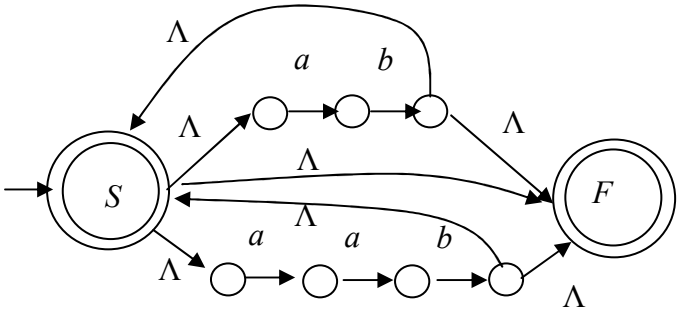
c) Kleen yıldızı:

Başlangıç durumu s
$K = K_1 \cup \{s\}$
$F = \{f_ç\}$
$\Delta = \Delta_1 \cup (F_1 \text{ x } \{\Lambda\} \text{ x } \{s_1\}) \cup (s, \Lambda, s_1) \cup (F_1 \text{ x } \{\Lambda\} \text{ x } F) \cup (s, \Lambda, f_ç)$

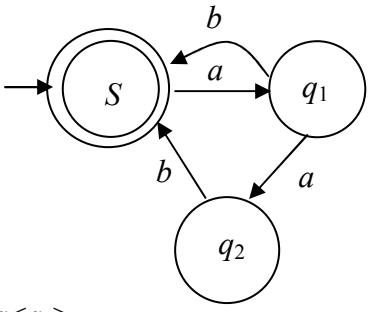


Örnek 3.2.6:

$(ab \vee aab)^*$



Yukarıdaki şekil yerine boş katarla gidilebilen durumlar üst üste bindirilerek aşağıdaki özet şekil ele alınabilir:

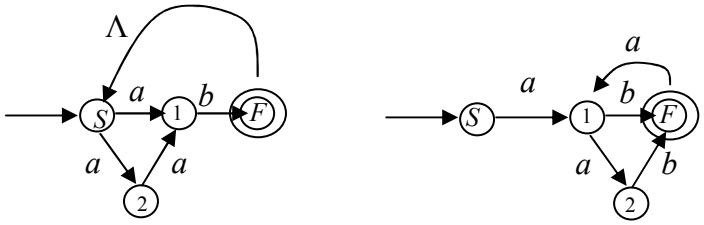


$\langle S \rangle ::= \Lambda \mid a\langle q_1 \rangle$
$\langle q_1 \rangle ::= b\langle s \rangle \mid a\langle q_2 \rangle$
$\langle q_2 \rangle ::= b\langle s \rangle$

Örnek 3.2.7:

$(ab \vee aab)^+$

$= (ab \vee aab)(ab \vee aab)^*$



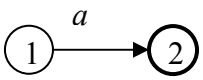
$\langle s \rangle ::= a\langle q_1 \rangle$
$\langle q_1 \rangle ::= b \mid b\langle F \rangle \mid a\langle q_2 \rangle$
$\langle q_2 \rangle ::= b\langle F \rangle \mid b$
$\langle F \rangle ::= a\langle q_1 \rangle$

Düzgün İfadeden Deterministik Otomat Oluşturmak İçin

1. Birinci aşamada düzgün ifadeden NFA oluşturulur.
2. İkinci aşamada NFA'dan DFA oluşturulur.
3. Son aşamada DFA indirgenerek yeni bir DFA oluşturulur (indirgeme merdiveni kullanılabilir).

Birinci aşama:

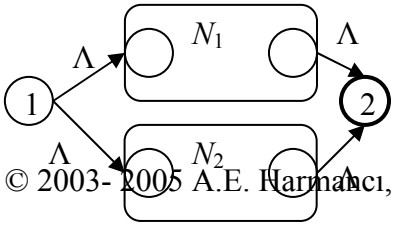
a) Bir simge



Şekil 3.2 Bir simge

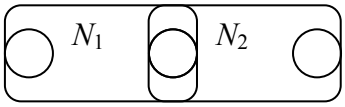
b) $R_1 \vee R_2$, R_1 ve R_2 iki düzgün ifade ise:

$R_1 \rightarrow N_i$ ve $R_2 \rightarrow N_2$; N_1 ve N_2, R_1 ve R_2'nin NFA'ları ise:

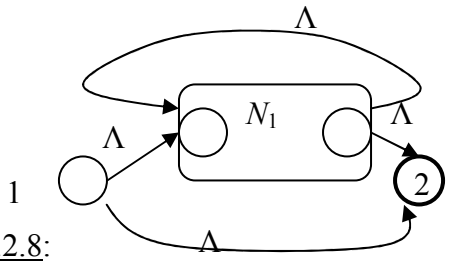


Şekil 3.3

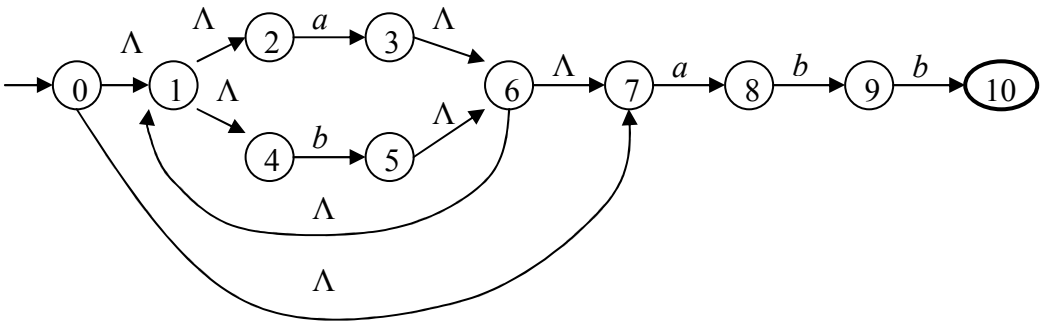
c) R_1R_2 ve (N_1 ve N_2) verilmiş



d) $(R_1)^*$

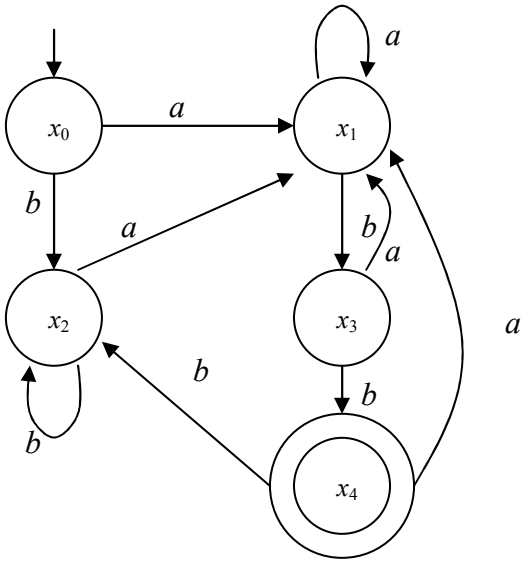


Örnek 3.2.8:
R = (a ∨ b)*abb
Birinci adımda NFA oluşturulur:

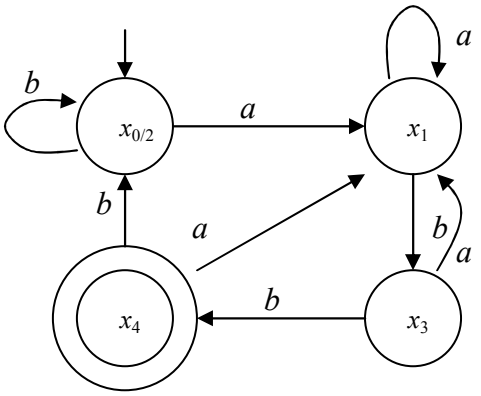


İkinci adımda NFA'dan DFA'ya geçilir:

$E(S') = \{ 0, 1, 2, 4, 7 \} = x_0$
$\delta(x_0, a) = E(3) \cup E(8) = \{ 1, 2, 3, 4, 6, 7, 8 \} = x_1$
$\delta(x_0, b) = E(5) = \{ 1, 2, 4, 5, 6, 7 \} = x_2$
$\delta(x_1, a) = x_1$
$\delta(x_1, b) = E(5) \cup E(9) = \{ 1, 2, 4, 5, 6, 7, 9 \} = x_3$
$\delta(x_2, a) = x_1$
$\delta(x_2, b) = E(5) = x_2$
$\delta(x_3, a) = x_1$
$\delta(x_3, b) = E(5) \cup E(10) = \{ 1, 2, 4, 5, 6, 7, 10 \} = x_4$
$\delta(x_4, a) = x_1$
$\delta(x_4, b) = x_2$



Yukarıdaki DFA'nın indirgenmiş hali aşağıda verilmiştir:



DFA'dan düzenli dil bulmanın sistematik yolu:

$K = \{ q_1, ..., q_n \}$
$s = q_1$
$R(i, j, k)$ burada $i, j \in \{ 1, 2, ..., n\}$; $k \in \{ 1, 2, ..., n + 1 \}$

i numaralı durumdan (düğümden) j numaralı duruma (düğüme) k veya daha yüksek numaralı durumlardan geçmeden elde edilebilecek katarların kümesinin düzgün ifadesi:

$$r(i, j, k) = \{ x \in \Sigma^* \mid (q_i, x) \vdash_M^* (q_j, \Lambda) \wedge (\exists y \in \Sigma^* \wedge (q_i, x) \vdash_M^* (q_l, y) \Rightarrow (l < k) \vee (y = \Lambda \wedge l = j) \vee [(y = x) \wedge l = i])\}$$

Buradan

$k = n + 1$
$R(i, j, n + 1) = \{ x \in \Sigma^* \mid (q_i, x) \vdash_M^* (q_j, \Lambda) \}$ ve
$L(M) = \cup\{R(1, j, n + 1) \mid q_j \in F \}$

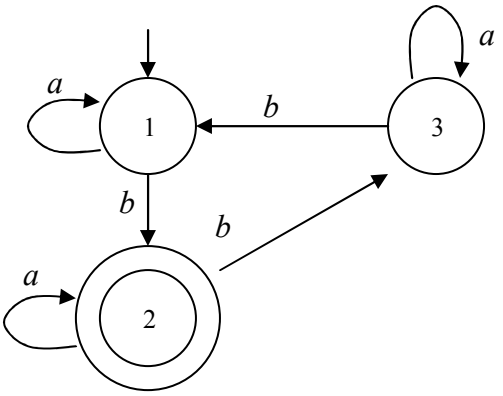
Birinci adım:

$$R(i, j, 1) = \begin{cases} i \neq j \Rightarrow \{ \sigma \in \Sigma \mid \delta(q_i, \sigma) = q_j \} \\ i = j \Rightarrow \{\{\Lambda\} \cup \{ \sigma \in \Sigma \mid (\delta(q_i, \sigma) = q_i)\} \end{cases}$$

İkinci adım:

$R(i, j, k+1) = R(i, j, k) \cup R(i, k, k)\ [R(k, k, k)]^* R(k, j, k)$

Örnek 3.2.9 (Papadimitriou kitabından):



Şekil 3.4

$\{w \mid w$ katarı içinde $3k+1$ adet b olmalı ve $k \in N\}$

$L(M) = a^*ba^*[(\ ba^*\ )^3]^*$

Buna benzer ifadeyi

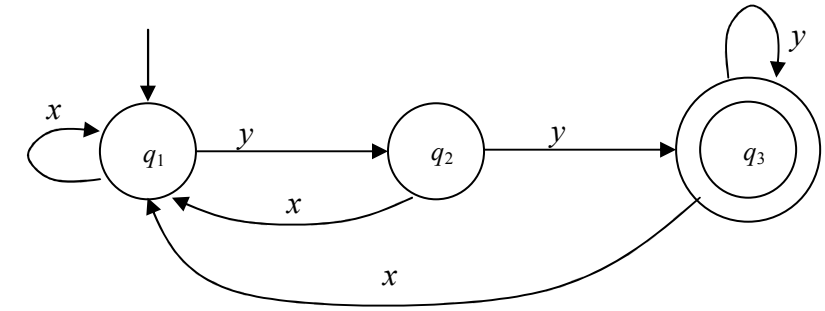
$L(M) = R(\ 1, 2, 4\ )$

$R(1, 2, 4\ ) = R(\ 1, 2, 3\ ) \cup R(1, 3, 3\ )\ [\ R(\ 3, 3, 3\ ))^*\ R(\ 3, 2, 3\ )$

...

$L(M) = a^*ba^* \vee a^*\ ba^*b\ (\ a \vee \Lambda \vee ba^*ba^*b)^*ba^*ba^*$

Örnek 3.2.10:



Şekil 3.5

Genel Formül:

$R(i, j, k+1) = R(\ i, j, k\ ) \cup R(\ i\ ,k, k\ )\ [R(k, k, k)\ ]^*\ R(\ k, j, k\ )$

$R(1, 3, 4) = R(1, 3, 3) \cup R(1, 3, 3)\ R^*(3, 3, 3)\ R(3, 3, 3)$

$R(1, 3, 3) = R(1, 3, 2) \cup R(1, 2, 2)\ R^*(2, 2, 2)\ R(2, 3, 2)$

$R(3, 3, 3) = R(3, 3, 2) \cup R(3, 2, 2)\ R^*(2, 2, 2)\ R(2, 3, 2)$

$R(1, 3, 2) = R(1, 3, 1) \cup R(1, 1, 1)\ R^*(1, 1, 1)\ R(1, 3, 1) = \varnothing$

$R(1, 2, 2) = R(1, 2, 1) \cup R(1, 1, 1)\ R^*(1, 1, 1)\ R(1, 2, 1) = y \vee (\Lambda \vee x)x^*y = x^*y$

$R(2, 2, 2) = R(2, 2, 1) \cup R(2, 1, 1)\ R^*(1, 1, 1)\ R(1, 2, 1) = \Lambda \vee xx^*y = (\Lambda \vee x^+y)$

$R(2, 3, 2) = R(2, 3, 1) \cup R(2, 1, 1)\ R^*(1, 1, 1)\ R(1, 3, 1) = y$
$R(3, 3, 2) = R(3, 3, 1) \cup R(3, 1, 1)\ R^*(1, 1, 1)\ R(1, 3, 1) = \Lambda \vee y$
$R(3, 2, 2) = R(3, 2, 1) \cup R(3, 1, 1)\ R^*(1, 1, 1)\ R(1, 2, 1) = xx^*y = x^+y$

$R(1, 1, 1) = \Lambda \vee x$
$R(1, 2, 1) = y$
$R(1, 3, 1) = \varnothing$
$R(2, 1, 1) = x$
$R(2, 2, 1) = \Lambda$
$R(2, 3, 1) = y$
$R(3, 1, 1) = x$
$R(3, 2, 1) = \varnothing$
$R(3, 3, 1) = \Lambda \vee y$

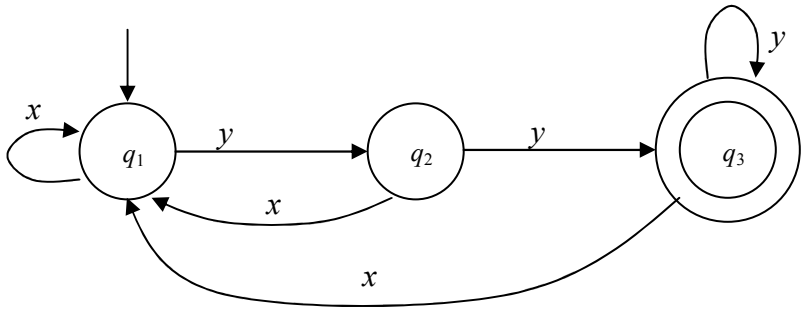
$R(3, 3, 3) = (\Lambda \vee y) \vee (x^+y)(x^+y)^*y = \Lambda \vee y \vee ((x^+y)^+y = \Lambda \vee [\Lambda \vee (x^+y)^+]y$
$= \Lambda \vee (x^+y)^*y$
$R(1, 3, 3) = x^*y(x^+y)^*y$
$R(1, 3, 4) = (x^*y(x^+y)^*y) \vee (x^*y(x^+y)^*y)[(x^+y)^*y)]^*[\Lambda \vee (x^+y)^*y]$
$= x^*y(x^+y)^*y[\Lambda \vee [(x^+y)^*y]^*] = x^*y[(x^+y)^*y]^+$

DFA'dan düzenli dil bulmanın ikinci yolu:
Teorem: $X = XA \cup B \ \wedge \ \Lambda \notin A$ denkleminin tek çözümü $X = BA^*$'dır.
Teoremin düzenli ifadelerle yazılışı:

$x = xa \vee b \ \wedge \ \Lambda \notin A \ \Rightarrow x = ba^*$

Yukarıda verilen teoremin otomattan düzenli ifadenin elde edilmesinde kullanılması

Örnek 3.2.11: Bu otomat, Örnek 3.2.5'te NFA durum diyagramı verilen otomatın DFA şekline dönüştürülmüş ifadesidir.



Durumlara varışlara bakılarak aşağıdaki denklemler yazılır:

$q_1 = q_1x \vee q_2x \vee q_3x \vee \Lambda$
$q_2 = q_1y$ içerisinde "b" terimi olmadığından kullanılamaz.
$q_3 = q_2y \vee q_3y \Rightarrow$ teorem uyarınca $q_3 = q_2yy^* \Rightarrow q_3 = q_1yy^+$

Buna göre;
$q_1 = q_1x \vee q_1yx \vee q_1yy^+x \vee \Lambda$
$q_1 = q_1(x \vee yx \vee yy^+x) \vee \Lambda \Rightarrow q_1 = (x \vee yx \vee yy^+x)^* = (y^*x)^*$

$q_3 = (y^*x)^*yy^+$ ifadesi bulunur. Bunun Örnek 3.2.5'te verilen NFA durum diyagramından hareketle sezgisel olarak bulunan $(x \vee y)^*yy^+$ ifadesi ile aynı olduğu aşağıdaki tanıtlanmıştır:

Tanıt:

$(y^*x)^*yy^+ = (x \vee y)^*yy^+$

Çözüm:
$(x \vee y)^*yy^+ = (x \vee y)^*y^*yy$ şeklinde yazılabilir.
$(y^*x)^*yy^+ = (y^*x)^*y^*yy$ şeklinde yazılabilir.
Yukarıdaki ifadelerden hareketle $(x \vee y)^*y^* = (y^*x)^*y^*$ olduğunu gösterelim. Bunun için
a) $(y^*x)^*y^* \subseteq (x \vee y)^* = (y^*x^*)^*$ ve
b) $(x \vee y)^* \subseteq (y^*x)^*y^*$ oldukları gösterilmelidir.

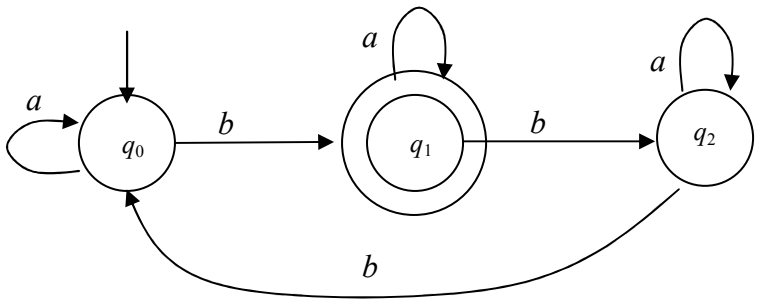
a) şıkkının tanıtı:
$(y^*x)^* \subseteq (y^*x^*)^*$
$(y^*x)y^* \subseteq (y^*x^*)^*y^* = (x^*y^*)^*y^*$
$(x^*y^*)^* = \Lambda \vee x^*y^* \vee (x^*y^*)^2 \vee .... \vee (x^*y^*)^n \vee ...$
$(x^*y^*)^*y^* = y^* \vee x^*y^*y^* \vee (x^*y^*)^2y^* \vee ... \vee (x^*y^*)^ny^* \vee ....$
$= \Lambda \vee y^+ \vee x^*y^*y^* \vee .... \vee (x^*y^*)^{n-1}x^*y^*y^* \vee ...$
$= \Lambda \vee y^+ \vee x^*y^* \vee .... \vee (x^*y^*)^n \vee ...$
$= \Lambda \vee x^*y^* \vee .... \vee (x^*y^*)^n \vee ... = (x^*y^*)^*$

b) şıkkının tanıtı:
$(x \vee y)^* = \Lambda \vee (x \vee y) \vee (x \vee y)^2 \vee (x \vee y)^3 \vee ....$
Tümavarım yöntemi ile tanıtlayalım.
$(x \vee y)^0 = \Lambda \subseteq (y^*x)^*y^*$ açık olarak görülmektedir.
$(x \vee y)^1 \subseteq (y^*x)^*y^*$ açık olarak görülmektedir. Zira $x \subseteq (y^*x)^*y^* \wedge y \subseteq (y^*x)^*y^*$
...
$(x \vee y)^n \subseteq (y^*x)^*y^*$ olduğunu kabul edelim;
$(x \vee y)^n(x \vee y) = (x \vee y)^n x \vee (x \vee y)^n y \subseteq (y^*x)^*y^*$ olduğu gösterilmeli.
i) $(x \vee y)^n x \subseteq (y^*x)^*y^*x = (y^*x)^+ \subseteq (y^*x)^* \subseteq (y^*x)^*y^*$
ii) $(x \vee y)^n y \subseteq (y^*x)^*y^*y = (y^*x)^*y^+ \subseteq (y^*x)^*y^*$

NOT:
$X = XA \cup \{\Lambda\}$ çözümü $X = A^*$
$X = XA$ denkleminin çözümü yok zira $B = \varnothing$

Örnek 3.2.12:
w giriş katarı içinde $3k+1$ adet b sembolü varsa kabul durumuna giden aksi halde bunu kabul etmeyen otomatı bulunuz ve buna ait düzenli ifadeyi çıkarınız. $\Sigma = \{ a, b \}$.

Otomatın şeması aşağıdaki gibi olur:



Aranan bir çözüm $a^*ba^*[(ba^*)^3]^*$

$q_0 = q_0 \vee q_2 b \vee \Lambda$
$q_1 = q_0 b \vee q_1 a$
$q_2 = q_1 b \vee q_2 a$

$q_2 = q_2 a \vee q_1 b \Rightarrow q_2 = q_1 b a^*$
$q_1 = q_1 a \vee q_0 b \Rightarrow q_1 = q_0 b a^*$

Bu ikisinden:

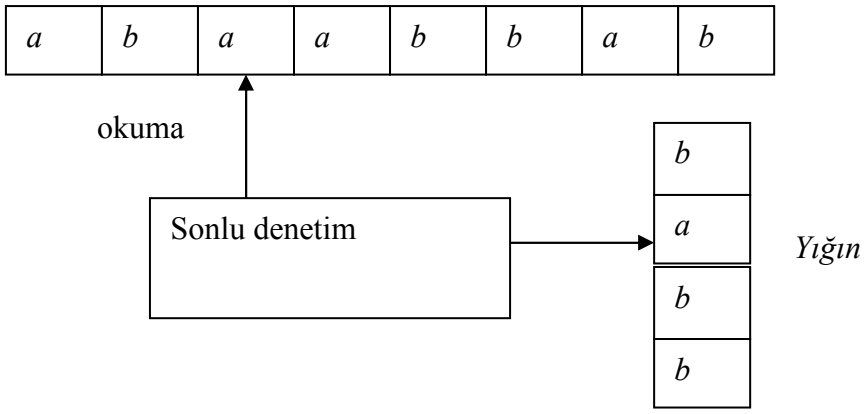
$q_2 = q_0(ba^*)(ba^*)$ elde edilir. Devam edilirse...

$q_0 = q_0 a \vee q_0(ba^*)^2 b \vee \Lambda \Rightarrow q_0 = q_0(a \vee (ba^*)^2 b) \vee \Lambda = (a \vee (ba^*)^2 b)^*$
$q_1 = (a \vee (ba^*)^2 b)^* b \vee q_1 a$
$q_1 = (a \vee (ba^*)^2 b)^* ba^*) = (a \vee (ba^*)^2 b)^*(ba^*)$ bulunur.

BÖLÜM 4

Yığın yapılı otomat (PDA) ve bağlamdan bağımsız dillerin tanımlanması

Bağlamdan bağımsız tüm dilleri tanıyan sonlu otomat mevcut değildir. Örnek olarak ($ww^R \mid w \in \Sigma^*$). Bu dili tanıyan otomatın belleğinin olması gereklidir. Bağlamdan bağımsız her dil yığın yapılı otomat tarafından tanınır.



Şekil 4.1 Yığın yapılı otomat

Bu otomat deterministik değildir. Giriş şeridi sadece okunurken yığına hem bilgi yazılır, hem de yığından bilgi okunur.

Tanım:

$M = (K, \Sigma, \Gamma, \Delta, s, F)$
K: sonlu durum kümesi
Σ: giriş alfabesi
Γ: yığın alfabesi
$s \in K$ başlangıç durumu
$F \subseteq$ k son durumlar kümesi
Δ geçiş bağıntısı

$\Delta \subseteq (K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$

Örnek:

$(w \subset w^R \mid w \in \{ a, b \}^*)$
$M = (K, \Sigma, \Gamma, \Delta, s, F)$
$K = \{ s, f \}, \Sigma = \{ a, b, c \}, \Gamma = \{ a, b \}, F = \{ f \}$
$\Delta = \{[(s, a, \Lambda), (s, a)], [(s, b, \Lambda), (s, b)], [(s, c, \Lambda), (f, \Lambda)], [(f, a, a), (f, \Lambda)], [(f, b, b), (f, \Lambda)] \}$

s	*abb* c *bba*	Λ
s	*bb* c *bba*	a
s	*b* c *bba*	*ba*
s	c *bba*	*bba*
f	*bba*	*bba*
f	*ba*	*ba*
f	*a*	*a*
f	Λ	Λ

$Q = (V, \Sigma, R, S)$

$V = \{S, a, b, c\}$

$\Sigma = \{ a, b, c \}$

$<S> ::= a<S>a \mid b<S>b \mid c$

Tanım:

Push: Bir simgeyi yığına ekleme $((p, u, \Lambda), (q, a))$
Pop: Bir simgeyi yığından çıkarma $((p, u, a), (q, a))$
Konfigürasyon: $K \times \Sigma^* \times \Gamma^*$ elemanı. Örnek (q, xyz, abc) a: yığın tepesi, c: yığın dibi
Ani tanıtım (instantaneous description) veya bir adımda üretme (yields in one step)

$((p, u, \beta)(q, \gamma)) \in \Delta$ olsun ve $\forall x \in \Sigma^* \wedge \forall\ \alpha \in \Gamma^*$
$(p, ux, \beta\alpha) \vdash_M (q, x, \gamma\alpha)$
Burada u, giriş şeridinden; β, yığından okunur; γ, yığına yazılır.

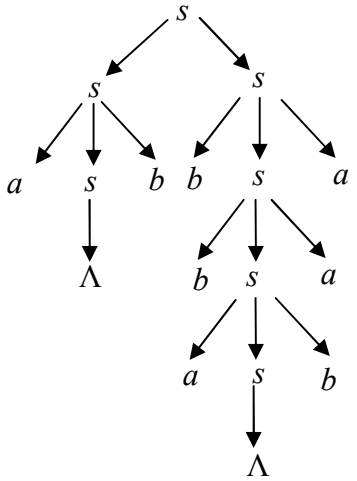
$\vdash_M$ işleminin yansımalı ve geçişli kapanışı $\vdash_M^*$ olsun. M otomatının w katarını kabul etmesi, $w \in \Sigma^*$ ve s başlangıç durumunu

$(s, w, \Lambda) \vdash_M^* (p, \Lambda, \Lambda)$ ve $p \in F$
$C_0 = (s, w, \Lambda)$ ve $C_n = (p, k, \Lambda)$ ile $C_0 \vdash_M C_1 \vdash_M \ldots. \vdash_M C_{n-1} \vdash_M C_n$
Bu işleme M otomatının yaptığı "hesaplama", (ing: computation) denir. Bu hesaplama n adımlıdır.
$L(M)$, M otomatı tarafından kabul edilen katarlar kümesidir.
$L(M) = \{ w \mid (s, w, \Lambda) \vdash_M^* (p, \Lambda, \Lambda) \wedge p \in F\}$

Örnek:

$w \in \{ \{a, b\}^* \mid \#(a) = \#(b) \}$
$M = (K, \Sigma, \Gamma, \Delta, s, F)$
$\Delta = \{ [(s, \Lambda, \Lambda), (q, c)], [(q, a, c), (q, ac)], [(q, a, a), (q, aa)]$
$[(q, a, b), (q, \Lambda)], [(q, b, c), (q, bc)], [(q, b, b), (q, bb)]$
$[(q, b, a), (q, \Lambda)]\ [(q, \Lambda, c), (f, \Lambda)]$

s	*abbbabaa*	Λ
q	*abbbabac*	*c*
q	*bbbabaa*	*ac*
q	*bbabaa*	*c*
q	*babaa*	*bc*
q	*abaa*	*bbc*
q	*baa*	*bc*
q	*aa*	*bbc*
q	*a*	*bc*
q	Λ	*c*
f	Λ	Λ

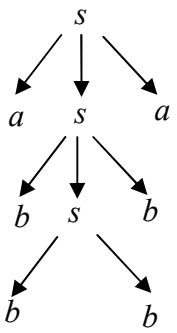


$Q = (V, \Sigma, R, s)$
$V = \{s, a, b\}$
$s = \{a, b\}$
$<s> ::= a<s>b \mid b<s>a \mid <s><s> \mid \Lambda$

Örnek:

$w \in \{ xx^R \mid x \in [\ a, b\ ]^* \}$
$M = (\ K, \Sigma, \Gamma, \Delta, s, F)$
$K = \{\ s, f\ \}, \Sigma\ = \Gamma = \{\ a, b\}, F = \{f\}$
$\Delta = \{\ ([s, a, \Lambda), (s, a)], [(s, b, \Lambda), (s, b)], [(s, \Lambda, \Lambda), (f, \Lambda)], [(f, a, a), (f, \Lambda)], [(f, b, b), (f, \Lambda)]\}$

s	*abbbba*	Λ
s	*bbbba*	*a*
s	*bbba*	*ba*
s	*bba*	*bba*
f	*bba*	*bba*
f	*ba*	*ba*
f	*a*	*a*
f	Λ	Λ



$Q = (V, \Sigma, R)$
$V = \{s, a, b\}$
$\Sigma = \{a, b\}$
$\langle S \rangle ::= a\langle S \rangle a \mid b\langle S \rangle b \mid aa \mid bb$

Örnek:

Her NFA bir PDA olarak tanımlanabilir. Bu durumda $\Gamma = \varnothing$ olur.
$\Delta' = \{(p, u, \Lambda), (q, \Lambda)) \mid (p, u, q) \in \Delta\}$
$M_{NFA} = (K, \Sigma, \Delta, s, F)$
$M'_{PDA} = (K, \Sigma, \Gamma, \Delta', s, F)$

Determinist PDA:

1) $\forall\ k \in K \wedge \forall\ \gamma \in \Gamma$ eğer $\delta(k, \Lambda, \gamma) \neq \varnothing \Rightarrow \delta(k, \sigma, \gamma) = \varnothing; \forall\ \sigma \in \Sigma$
2) $a \in \Sigma \cup \{\Lambda\}$ ise $\forall k$, $\forall \gamma$ ve $\forall a$ $\text{Card}(\delta(k, a, \gamma)) \leq 1$

Açıklama:

Her hangi bir konfigürasyondan tek adımda üretilebilen konfigürasyon sayısı tek ise, giriş "Λ" olduğunda (k, Λ, γ)'dan tek adımda tek bir üretim yapılabiliyorsa, aynı koşullarda giriş başka bir değer aldığında üretim olmamaktadır, ve $\delta(k, a, \gamma) = \varnothing$ olur.

Sonlu otomatlarda, kabul edilen diller açısından determinist ve determinist olmayan otomatlar arasında eşdeğerlilik bulunabilmekteydi, ancak PDA'lar için bu geçerli değildir.

ww^R determinist olmayan bir PDA tarafından kabul edildiği halde, bu dili kabul eden bir determinist PDA yoktur.

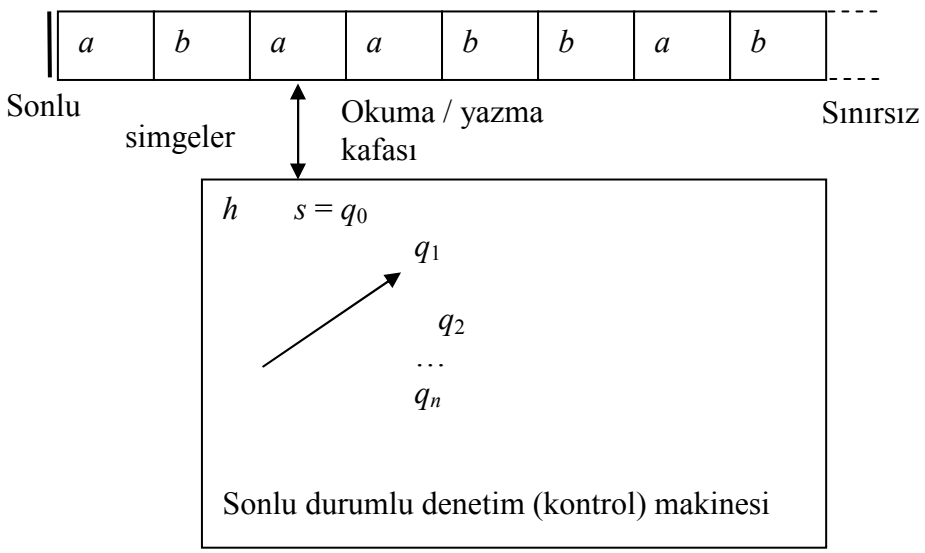
BÖLÜM 5

Turing Makinesi ve hesaplama kuramlarına giriş

Klasik otomat (DFA, NFA, PDA) ya da dil tanıyıcı otomat, çoğu basit dilleri tanımaya bile yetememektedir (Örnek: $a^n b^n c^n$: $n \geq 0$)

Daha genel bir dil tanıyıcı ve aynı zamanda, zincirden zincire fonksiyonel dönüşüm yapan makineler araştırılmıştır. Sonuç: Turing Makinesi. Her algoritma ya da hesaplama yordamının Turing Makinesi ile modellebileceği savı ortaya atılmıştır ve bu sav Church'ün tezi olarak adlandırılmıştır. İspat edilemese de tersi de gösterilememiştir. Bu nedenle bu bir konjektürdür.

Turing Makinesi tanım ve tanıtımı:



Şekil 5.1 Turing Makinesi

Okuma / yazma kafası önünde bulunduğu simgeyi okur, denetim makinesinin durumuna göre yerine ya yeni bir simge yazar, ya sağa (R), ya da sola (L) bir adım hareket eder ve denetim makinesi yeni bir duruma geçer. Kafa şeritteki en soldaki simgeden daha sola hareket etmek isterse makineyi takılma (ing: hang) durumuna sokar. Kafa sağa doğru istenildiği kadar hareket ettirilebilir.

Özel simgeler:

h: hesap sonu durumu (halt, giriş, çıkış ve durum alfabesinde yer almaz)
#: boşluk simgesi
L, R sol ve sağa hareketli simgeler (giriş, çıkış ve durum alfabesinde yer almaz)
Giriş sözcüğü genelde şeridin sol ucuna yazılır.

Tanım:

$<K, \Sigma, \delta, s>$
K durumları belirtir. (Genelde $h \notin K$'dır)
Σ giriş ve çıkış alfabesi ($\# \in \Sigma$) ama L, R $\notin \Sigma$
$s \in$ K başlangıç durumu
δ bir geçiş fonksiyonudur. $K \times \Sigma \rightarrow K \cup \{ h \} \times (\Sigma \cup \{L, R\})$

$q \in K \wedge a \in \Sigma \wedge \delta(q,a) = (p, b)$
q -> p
i) (Şeritten okunur) a -> $b \in \Sigma$ (a'nın yerine şeride yazılır)
ii) b = L (Kafa sola)
iii) b = R (Kafa sağa)

Turing makinası ile günümüzün bilgisayarları karşılaştırılacak olursa, teyp şeridinin bilgisayarın belleği olan RAM'e, Turing makinası durum tablosunun bilgisayarın programına, sonlu durumlu denetiminin de çevre birimleri çıkarılmış mikroişlemciye benzediği görülür. Burada "programın" yani durum geçiş tablosunun okunurluğunu arttırmak ve gereken satır miktarını azaltmak için, yukarıda verilen tanımdan özde farklı olmamak kaydı ile Turing makinası için durum geçiş tablosunun her satırı aşağıdaki gibi de yazılabilmektedir:

(q, σ, q', σ', HM)

Burada:
q ile makinanın o anki durumu,
q' ile gidilecek durumu,
σ ile şeritten okunan bilgi,
σ' ile şeride yazılacak bilgi (σ ile aynı olursa şeride yazma işlemi yapılmamış gibi olur)
HM ile kafanın hareketi ifade edilmektedir.
Burada HM ∈ {-1: sola hareket, 1: sağa hareket, 0: hareket yok}

Örnek 1:

Soldan sağa hareketle boşluğa kadar tüm simgelerini silip yerine boşluklar yazan makine:
$K = \{q_0, q_1\}$
$\Sigma = \{a, \#\}$
$s = q_0$

q	σ	$\Delta(q, \sigma)$
q_0	a	$(q_1, \#)$
q_0	#	$(h, \#)$
q_1	a	q_0, a *
q_1	#	q_0, R

Not: (*) satırı gereksizdir, fonksiyon tanımına uyması için konulmuştur.

Aynı makinayı farklı yazım şekliyle vermek istersek

q	σ	$\Delta(q, \sigma)$, HM
q_0	a	q_0, #, 1
q_0	#	h, #, 0

Şekilden görüleceği üzere daha az satır (durum) ile aynı işlem yapılabilmektedir.

Örnek 2:

Sola doğru a simgelerini tanıyan ve rastladığı ilk # simgesinde duran makineyi tanımlayınız.
$K = \{q_0\}$
$\Sigma = \{a, \#\}$
$s = q_0$

q	Σ	$\Delta(q, \sigma)$
$q0$	a	(q_0, L)
q_0	#	$(h, \#)$

Bir konfigürasyon aşağıdaki kümenin elemanıdır:

$K \cup \{h\}$ x Σ^* x Σ x $(\Sigma^*(\Sigma - \{\#\}) \cup \{\Lambda\})$

Örnekler: (Teyp kafası altı çizili harfin üzerinde bulunmaktadır)

(q, aba, a, bab) veya $(q, aba\underline{a}bab)$
$(h, \#\#, \#, \#a)$ veya $(h, \#\#\#\underline{\#}a)$
(q, Λ, a, aba) veya $(q, \Lambda\underline{a}aba)$ veya $(q, aaba)$
$(q, \#a\#, \#, \Lambda)$ veya $(q, \#a\#\underline{\#}\Lambda)$ veya $(q, \#a\#\underline{\#})$
Bu düzenleşim tanımına uymaz $(q, baa, a, bc\#)$ veya $(q, baaabc\#)$

Tanım:

Turing Makinesi bir adımda üretir (yields in one step)
$(q_1, w_1, a_1, u_1) \vdash_M (q_2, w_2, a_2, u_2)$
Burada $\delta(q_1, a_1) = (q_2, b)$ ve $b \in \Sigma \cup \{L, R\}$

1) $b \in \Sigma$, $w_2 = w_1$, $u_2 = u_1$, $a_2 = b$ (a_1 yerine a_2 yazması)
2) $b = L$, $w_1 = w_2a_2$; ($a_1 = \#$ ve $u_1 = \Lambda \Rightarrow u_2 = \Lambda$) veya ($a_1 \neq \#$ veya $u_1 \neq \Lambda \Rightarrow u_2 = a_1u_1$)
   Sola hareket, kafa boşluk üzerinde ise sağ boş olduğunda boşluk silinir, yoksa olduğu gibi bırakılır.
3) $b = R$, $w_2 = w_1a_1$; ($u_1 = a_2u_2$) veya ($u_1 = \Lambda \Rightarrow u_2 = \Lambda$ ve $a_2 = \#$)
   Sağa hareket, sağda boş sözcük varsa yerine boşluk yazılır.

Örnek 3:

$w, u \in \Sigma^*$; $a, b \in \Sigma$; (u, # ile bitemez)
$\delta(q_1, a) = (q_2, b) \Rightarrow (q_1, w\underline{a}u) \vdash_M (q_2, w\underline{b}u)$
$\delta(q_1, a) = (q_2, L) \Rightarrow$ i) $(q_1, wb\underline{a}u) \vdash_M (q_2, w\underline{b}au)$
ii) $(q_1, wb\underline{\#}) \vdash_M (q, w\underline{b})$
$\delta(q, a) = (q_2, R) \Rightarrow$ i) $(q_1, w\underline{a}bu) \vdash_M (q_2, wa\underline{b}u)$
ii) $(q_1, w\underline{a}) \vdash_M (q_2, wa\underline{\#})$

Tanım:

n uzunlukta bir hesaplama (ing: A computation of length n or a n steps computation)
$C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_n$

Örnek 4 (Örnek 1'in devamı):

$(q_0, \underline{a}aaa) \vdash_M (q_1, \underline{\#}aaa) \vdash_M (q_0, \#\underline{a}aa) \vdash_M (q_1, \#\#aa) \vdash_M (q_0, \#\#\underline{a}a)$
$\vdash_M (q_1, \#\#\#a) \vdash_M (q_0, \#\#\#\underline{a}) \vdash_M (q_1, \#\#\#\underline{\#}) \vdash_M (q_0, \#\#\#\#\underline{\#})$
$(h, \#\#\#\#\underline{\#})$

Tanım:

Turing Computable Function: (karakter zincirleri üzerinde dönüşüm yapabilme özelliği)

$M = (K, \Sigma, \delta, s)$

Σ_0 giriş alfabesi, Σ_1 çıkış alfabesi olmak üzere;
$f(w) = u \wedge w \in \Sigma_0^* \subseteq \Sigma^* \wedge u \in \Sigma_1^* \subseteq \Sigma^*$ ise
ve ayrıca:

$(s, \#w\underline{\#}) \vdash_M^* (h\#u\underline{\#}), \quad \# \notin \Sigma_0 \wedge \# \notin \Sigma_1$

Örnek 5: Katar eviren makine (*a* yerine *b*, *b* yerine *a* yazan otomat)

$K = \{q_0, q_1, q_2\}, \Sigma = \{a, b, \#\}, s = q_0, \delta$

q	σ	$\delta(q, \sigma)$
q_0	a	(q_1, L)
q_0	b	(q_1, L)
q_0	#	(q_1, L)
q_1	a	(q_0, b)
q_1	b	(q_0, a)
q_1	#	(q_2, R)
q_2	a	(q_2, R)
q_2	b	(q_2, R)
q_2	#	$(h, \#)$

$(q_0, \#aab\underline{\#}) \vdash_M (q_1, \#aa\underline{b}) \vdash_M (q_0, \#aa\underline{a}) \vdash_M (q_1, \#a\underline{a}a) \vdash_M (q_0, \#a\underline{b}a) \vdash_M (q_1, \#\underline{a}ba)$

$\vdash_M (q_0, \#\underline{b}ba) \vdash_M (q_1, \underline{\#}bba) \vdash_M (q_2, \#\underline{b}ba) \vdash_M (q_2, \#b\underline{b}a) \vdash_M (q_2, \#bb\underline{a}) \vdash_M (q_2, \#bba\underline{\#})$

$(h, \#bba\underline{\#})$

Örnek 6:

Sayısal örnek $f(n) = n + 1$
n sayısı *n* adet *III….I* simgesi ile verilsin.
$M = (K, \Sigma, \delta, s)$
$K = \{q_0\}$
$\Sigma = \{I, \#\}, s = q_0$

q	σ	$\delta(q, \sigma)$
q_0	I	(h, R)
q_0	#	(q_0, I)

a) $(q_0, \#II\underline{\#}) \vdash_M (q_0, \#III\underline{\#}) \vdash_M (h, \#III\underline{\#})$
b) $(q_0, \#I^n\underline{\#}) \vdash_M^* (h, \#I^{n+1}\underline{\#})$
c) $(q_0, \#\underline{\#}) \vdash_M (q_0, \#\underline{I}) \vdash_M (h, \#I\underline{\#})$

Tanım:

Turing Decidable Machine
Σ_0 alfabe ve $\# \notin \Sigma_0$ ve *L* dili $L \subseteq \Sigma_0^*$
x_L fonksiyonu aşağıdaki gibi hesaplanabiliyorsa:

$$\forall w \in \Sigma_0, \; F_L(w) = \begin{cases} \text{Ⓨ} \Rightarrow w \in L \\ \text{Ⓝ} \Rightarrow w \notin L \end{cases}$$

Örnek:

$\Sigma_0 = \{a\}$; $L = \{ w \in \Sigma_0^* \mid ||w|| \text{ çift sayı}\}$
$M = (K, \Sigma, \delta, s)$ ve $K = \{q_0, ..., q_6\}$
$\Sigma = \{ a, \text{Ⓨ}, \text{Ⓝ}, \# \}$
$s = q_0$

q	σ	$\delta(q, \sigma)$
q_0	#	(q_1, L)

q_1	a	$(q_2, \#)$
q_1	#	(q_4, R)
q_2	#	(q_3, L)
q_3	a	$(q_0, \#)$
q_3	#	(q_6, R)
q_4	#	$(q_5, \textcircled{Y})$
q_5	Y	(h, R)
q_5	N	(h, R)
q_6	#	$(q_5, \textcircled{N})$

İki farklı katar için makineyi çalıştırırsak:
a) $(q_0, \#aa\underline{\#}) \vdash_M (q_1, \#a\underline{a}) \vdash_M (q_2, \#a\underline{\#}) \vdash_M (q_3, \#\underline{a}) \vdash_M (q_0, \#\underline{\#}) \vdash_M (q_1, \underline{\#}) \vdash_M (q_4, \#\underline{\#})$
$\vdash_M (q_5, \#\textcircled{Y}) \vdash_M (h, \#\textcircled{Y}\underline{\#})$

b) $(q_0, \#aaa\underline{\#}) \vdash_M (q_1, \#aa\underline{a}) \vdash_M (q_2, \#aa\underline{\#}) \vdash_M (q_3, \#a\underline{a}) \vdash_M (q_0, \#a\underline{\#}) \vdash_M (q_1, \#\underline{a})$
$\vdash_M (q_2, \#\underline{\#}) \vdash_M (q_3, \underline{\#}) \vdash_M (q_6, \#\underline{\#}) \vdash_M (q_5, \#\underline{\textcircled{N}}) \vdash_M (h, \#\textcircled{N}\underline{\#})$

a) $(q_0, \#a^n\underline{\#}) \vdash_M^* (h, \#\textcircled{Y}\underline{\#}) \Rightarrow$ *n* çift sayı
b) $(q_0, \#a^n\underline{\#}) \vdash_M^* (h, \#\textcircled{N}\underline{\#}) \Rightarrow$ *n* tek sayı

Uygulamalar:

1) $\#w\underline{\#} \to \#w\#w\underline{\#}$
2) $\#w\underline{\#} \to \#ww^R\underline{\#}$
3) $\#ww^R\underline{\#} \to \#\textcircled{Y}\underline{\#}$

Makine olsaydı bu işlemleri nasıl yapardı diye düşünerek belli örnekler üzerinde işlemleri oluşturuyuruz. Buradan hareketle durum geçiş tablolarını oluşturulabilir.

#aba# ->*#abaγ* -> *#abaγ* -> ... -> *#abaγ* -> *#abaγ* -> *#αbaγ#* -> .. -> *#αbaγ#* -> *#αbaγa*
-> … -> *#αbaγa* -> *#abaγa* -> *#aβaγa* -> ... -> *#aβaγa#* -> *#aβaγab* -> ... ->
#aβaγab -> *abaγab* -> *#abaγab* -> *#abαγab* -> ... -> *#abαγab#* -> *abαγaba* -> ... -> *#abαγaba*
-> *#abaγaba* -> *#abaγaba* -> *aba#aba* -> ... -> *#aba#aba#*

1a) $\#\#\underline{} \to \#\underline{\gamma} \to \underline{\#}\gamma \to \#\#\underline{} \to \#\#\#\underline{}$ (kural üretirken önce sınır değerler ele alınır)
2) $\#ww^R\underline{\#}$
#abb# -> *#abb* -> *#abβ* -> *#abβ#* -> *abβb* -> *#abβb* -> *#abbb* ->*#abbb* -> *#aβbb* -> *#aβbb* ->
-> *#aβbb* -> *#aβbb#* -> *#aβbbb* -> ... -> *#aβbbb* ->*#abbbb* -> *#αbbbb* -> *#αbbbb* ->
-> *#αbbbb#* -> *#αbbbba* -> *#αbbbbaa* -> ... -> *#αbbbba* -> *#abbbba* ->*#abbbbba* -> *#abbbba* ->
-> … -> *#abbbba* -> *#abb|bba#*
3)
#abb bba# -> *#abbbb* -> … -> *#abbbb* -> *#abbbb* -> *#θbbbb* -> … ->*#θbbbb#* -> *#θbbbb* ->
-> *#θbbb* -> … -> *#θbbb* -> *#θbbb* -> *#θθbb* -> … -> *#θθbb* -> # *θθbb#* -> *#θθb* -> *#θθb* ->
-> *#θθb* -> *#θθb* -> *#θθθ* -> *#θθθ*Ⓨ*#* -> *#θθθ* -> *#θθ* -> *#θ* -> *#* -> *#*Ⓨ-> *#*Ⓨ*#*
…
-> *#θθbbb* -> *#θθbb* -> *#θθθb#* -> *#θθθb* -> *#θθθ* -> *#θθθ#* (makine burada çıkışın *N*
olacağını anlar)-> *θθθ#* -> *#θθθ* ->
-> … -> *#θ* -> *#* -> *#*Ⓝ*#*
…
#ab# -> *#ab* -> *#a* -> *#a* -> *#a* (makine burada çıkışın *N* olacağını anlar)-> *#a#* -> *#a* -> *#* ->
#Ⓝ-> *#*Ⓝ*#*

#aaab# -> *#aaab* -> *#aaa* -> … -> *#aaa* -> *#aaa* (makine burada çıkışın *N* olacağını anlar)->
#aaa# -> *#aaa* -> *#aa* -> *#a* -> *#* ->
-> #Ⓝ-> #Ⓝ#