***HAFTA 2***

1.7 Alfabeler ve Diller

Son bölümde gayri resmi olarak incelediğimiz algoritmalarda belirsiz bırakılan çok şey var. Örneğin, erişilmesi ve değiştirilmesi gereken R ve R\* ilişkilerinin tam olarak nasıl temsil edildiğini ve saklandığını belirtmedik. Hesaplama pratiğinde, bu tür veriler bilgisayarın belleğinde bit dizileri veya bilgisayar tarafından manipüle edilmeye uygun diğer semboller olarak kodlanır. Bu nedenle hesaplama teorisinin matematiksel incelemesi, sembol dizilerinin matematiğini anlamakla başlamalıdır.

Alfabe kavramıyla başlıyoruz: sonlu bir semboller kümesi. Buna örnek olarak, doğal olarak, Roma alfabesi {a, b,...,z} verilebilir. Hesaplama teorisiyle özellikle ilgili bir alfabe de {0, 1} ikili alfabesidir. Aslında, herhangi bir nesne bir alfabede yer alabilir; biçimsel bir bakış açısından, bir alfabe basitçe herhangi bir türden sonlu bir kümedir. Ancak basitlik açısından, sembol olarak yalnızca harfleri, rakamları ve $ veya # gibi diğer yaygın karakterleri kullanıyoruz.

Bir alfabe üzerindeki bir dize, alfabedeki sembollerin sonlu bir dizisidir. Diğer dizileri yazdığımız gibi, dizeleri parantez ve virgüllerle yazmak yerine, sembolleri yan yana getiririz. Böylece karpuz {a,b,...,z} alfabesi üzerinde bir dizidir ve 0111011 {0, 1} üzerinde bir dizidir. Ayrıca, doğal izomorfizmi kullanarak, sadece bir sembolden oluşan bir dizgeyi sembolün kendisiyle tanımlarız; böylece a sembolü a dizgesiyle aynıdır. Bir dizgede hiç sembol olmayabilir; bu durumda boş dizge olarak adlandırılır ve e ile gösterilir.Dizgeleri belirtmek için genellikle u, v, x, y, z ve Yunan harflerini kullanırız; örneğin, abc dizgesine ad olarak w kullanabiliriz. Elbette, karışıklığı önlemek için, dizgilerin isimleri olarak da kullandığımız harfleri sembol olarak kullanmaktan kaçınmak iyi bir uygulamadır. Bir ∑ alfabesi üzerinde boş dizge de dahil olmak üzere tüm dizgelerin kümesi ∑\* ile gösterilir.

Bir dizenin uzunluğu, bir dizi olarak uzunluğudur; dolayısıyla acrd dizesinin uzunluğu 4'tür. Bir dizenin uzunluğunu |w| ile gösteririz; dolayısıyla |101| = 3 ve lel = 0 olur. Alternatif olarak (yani, doğal bir izomorfizm yoluyla) bir dize w ∈ ∑\* bir fonksiyon w olarak düşünülebilir: {1,..., |w|} →∑ ; w(j) değeri, burada 1 ≤ j≤|wl, w'nin j. pozisyonundaki semboldür. Örneğin, eğer w = akordeon ise, o zaman w(3) = w(2) = c ve w(1) = a. Bu alternatif bakış açısı olası bir karışıklığa yol açar. Doğal olarak, üçüncü pozisyondaki c sembolü ikinci pozisyondaki c sembolüyle aynıdır. Ancak, bir dizedeki farklı pozisyonlardaki aynı sembolleri ayırt etmemiz gerekirse, bunlara sembolün farklı oluşumları olarak atıfta bulunacağız. Yani, w(j) = 𝛔 ise, 𝛔 ∈ ∑ sembolü w ∈ ∑\* dizisinin j'inci basamağında yer alır.

Aynı alfabe üzerindeki iki dizgi, birleştirme işlemi ile üçüncü bir dizgi oluşturmak üzere birleştirilebilir. x ve y dizgilerinin birleştirilmesi, x o y veya sadece xy olarak yazılır, x dizgisini y dizgisinin takip etmesidir; resmi olarak, w = x o y ancak ve ancak |w| = |x| + lyl, j = 1,..., lxl için w(j) =x(j) ve j = 1,..., |y| için w(|x|+ j) = y (j) ise. Örneğin, 01 o 001 = 01001 ve beach o boy = beachboy. Elbette, herhangi bir w dizgesi için w o e = e o w = w. Ve birleştirme birleştiricidir: herhangi bir w, x ve y dizgesi için (wx)y = w(xy). Bir v dizgesi, ancak ve ancak w = xvy olacak şekilde x ve y dizgeleri varsa w dizgesinin bir alt dizesidir. Hem x hem de y e olabilir, bu nedenle her dizge kendisinin bir alt dizgesidir; ve x = w ve v = y = e alırsak, e'nin her dizgenin bir alt dizesi olduğunu görürüz. Eğer bazı x'ler için w = xv ise, v w'nin bir sonekidir; eğer bazı y'ler için w = vy ise, v w'nin bir önekidir. Böylece road, roadrunner'ın bir önekidir, abroad'ın bir sonekidir ve hem bunların hem de broader'ın bir alt dizesidir. Bir dizede aynı alt dizenin birden fazla oluşumu olabilir; örneğin, ababab'ın üç ab ve iki abab oluşumu vardır.

Her w dizisi ve her i doğal sayısı için w dizisi şu şekilde tanımlanır:

metin, yazı tipi, beyaz, tipografi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Böylece w^1 =w ve do^2 = dodo olur.

Bu tanım, çok yaygın bir türün ilk örneğidir: tümevarım yoluyla tanım. Tümevarım yoluyla ispatları daha önce görmüştük ve temelde yatan fikir hemen hemen aynıdır. Tanımın bir temel durumu vardır, burada i = 0 için w^i tanımı; daha sonra tanımlanan şey tüm j≤i için belirtildiğinde, j= i + 1 için tanımlanır. Yukarıdaki örnekte, w^i+1, w^i açısından tanımlanmıştır. Tanımdaki herhangi bir durumun temel duruma nasıl geri götürülebileceğini tam olarak görmek için do^2 örneğini ele alalım. Yine tanıma göre (i = 1 ile), (do)^2 = (do)^1 o do. Yine tanıma göre (i = 0 ile) (do)^1 = (do)^0 o do. Şimdi temel durum geçerlidir: (do)^0 = e. Yani (do)^2= (e o do) o do = dodo.

Bir w dizesinin tersi, w^R ile gösterilir, “tersten yazılan” dizedir: örneğin, reverse^R = esrever. Bir dizginin uzunluğu üzerine tümevarım yoluyla resmi bir tanım verilebilir:

(1) Eğer w uzunluğu 0 olan bir dizge ise, w^R = w = e olur.

(2) Eğer w, n + 1 > 0 uzunluğunda bir dizge ise, bazı a ∈ ∑ için w = ua ve w^R = au^R olur.

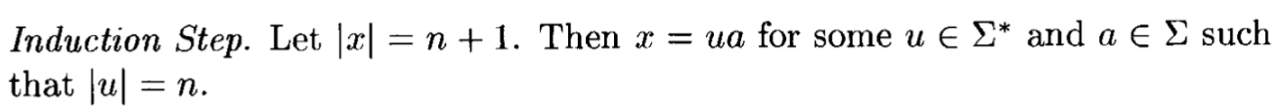
Bu tanımı, tümevarım yoluyla bir ispatın tümevarım yoluyla bir tanıma nasıl bağlı olabileceğini göstermek için kullanalım. Herhangi bir w ve x dizisi için (wx)^R =x^Rw^R Örneğin, (dogcat)^R = (cat)^R (dog)R = tacgod olduğunu göstereceğiz. x'in uzunluğu üzerinde tümevarım yoluyla ilerleyeceğiz.

metin, yazı tipi, beyaz, makbuz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, yazı tipi, beyaz, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

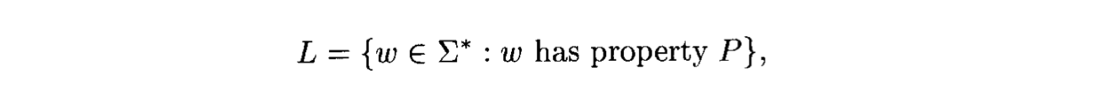


metin, yazı tipi, makbuz, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şimdi tek tek dizgelerin incelenmesinden sonlu ve sonsuz dizge kümelerinin incelenmesine geçiyoruz. Yakında inceleyeceğimiz basit hesaplama makinesi modelleri, birçok farklı dizgiyi ele alma biçimlerindeki düzenlilikler açısından karakterize edilecektir, bu nedenle öncelikle dizgi sınıflarını tanımlamanın ve birleştirmenin genel yollarını anlamak önemlidir.

Bir ∑ alfabesi üzerindeki herhangi bir dizgi kümesi -yani ∑\*'in herhangi bir alt kümesi- bir dil olarak adlandırılacaktır. Böylece ∑\*, ∅ ve ∑ birer dildir. Bir dil basitçe özel bir küme türü olduğundan, sonlu bir dili tüm dizelerini listeleyerek belirtebiliriz. Örneğin, {aba, czr, d, f} {a,b,...,2} üzerinde bir dildir. Bununla birlikte, ilgilenilen çoğu dil sonsuzdur, bu nedenle tüm dizeleri listelemek mümkün değildir. Dikkate alınabilecek diller {0,01, 011, 0111,...}, {w ∈ {0,1}\*: w eşit sayıda 0 ve 1'e sahiptir} ve {w ∈ ∑\*: W=w^R}. Böylece sonsuz dilleri şema ile belirleyeceğiz:



Sonsuz kümeleri belirtmek için kullandığımız genel formu takip eder:

Eğer ∑ sonlu bir alfabe ise, o zaman ∑\* kesinlikle sonsuzdur; fakat sayılabilir sonsuz bir küme midir? Durumun gerçekten de böyle olduğunu görmek zor değildir. Bir f: N → ∑\* bijeksiyonu oluşturmak için, önce alfabenin bazı sıralamalarını sabitleyin, diyelim ki ∑ = {a1,...,an}, burada a1,..., an farklıdır. Daha sonra ∑\* üyeleri aşağıdaki şekilde numaralandırılabilir:

(1) Her k ≥ 0 için, k uzunluğundaki tüm dizgiler k + 1 uzunluğundaki tüm dizgilerden önce sıralanır.

(2) Tam olarak k uzunluğundaki n^k dizgileri leksikografik olarak sıralanır, yani ai1... aik, aj1...ajk'dan önce gelir, bazı m için, 0 ≤ m ≤k - 1, l = 1,...,m için il = jl ve im+1 < jm+1 olması şartıyla.

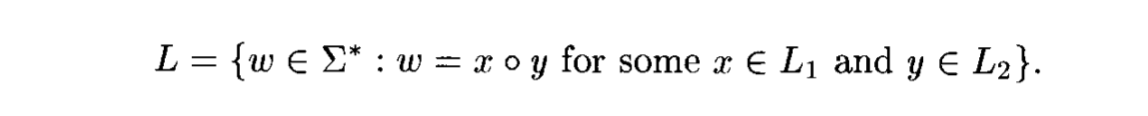
metin, yazı tipi, beyaz, tipografi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Eğer ∑ Roma alfabesi ise ve ∑ = {a1,..., a26) sıralaması olağan {a,...,z} ise, eşit uzunluktaki dizgiler için leksikografik sıralama sözlüklerde kullanılan sıralamadır; ancak, ∑\* içindeki tüm dizgiler için (1) ve (2) ile tanımlanan sıralama, daha kısa dizgileri daha uzun olanlardan önce listeleyerek sözlük sıralamasından farklıdır.

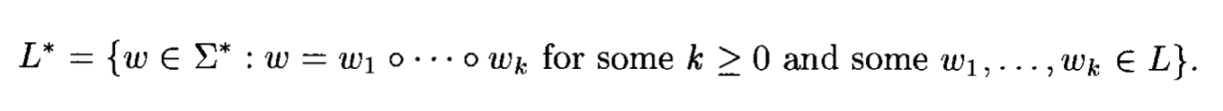
Diller küme olduklarından, birleşim, kesişim ve fark küme işlemleri ile birleştirilebilirler. Belirli bir alfabe ∑ bağlamdan anlaşıldığında, ∑\* - A farkı yerine A. - A'nın tümleyeni yazacağız.

Ayrıca, bazı işlemler sadece diller için anlamlıdır. Bunlardan ilki dillerin birleştirilmesidir. Eğer L1 ve L2 ∑ üzerinde diller ise, bunların birleşimi L = L1 o L2 veya basitçe L = L1L2 olur, burada:



Örneğin, ∑ = {0,1}, L1 = {w ∈ ∑\*: w'de çift sayıda 0 vardır} ve L2 = {w: w bir 0 ile başlar ve geri kalan semboller 1'dir} ise, L1 o L2 = {w: w'de tek sayıda 0 vardır}.

Bir başka dil işlemi de L dilinin Kleene yıldızıdır ve L\* ile gösterilir. L\*, L'den sıfır veya daha fazla dizginin birleştirilmesiyle elde edilen tüm dizgilerin kümesidir. (Sıfır dizginin birleştirilmesi e'dir ve bir dizginin birleştirilmesi dizginin kendisidir). Böylece,



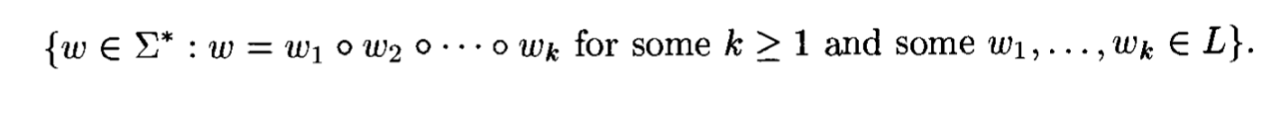
Örneğin, L = {01, 1, 100} ise, 110001110011 ∈ L\*, çünkü 110001110011 = 1 o 100 o 01 o 1 o 100 o 1 o 1 ve bu dizgilerin her biri L'dedir.

∑ üzerindeki tüm dizgelerin kümesini belirtmek için ∑\* kullanımının, sonlu bir dil olarak kabul edilen ∑'nin Kleene yıldızının gösterimiyle tutarlı olduğuna dikkat edin. Yani, L = ∑ olsun ve yukarıdaki tanımı uygulayalım, o zaman ∑\* bazı k ≥ 0 ve bazı w1,..., wk ∈ ∑ için w1 o... o wk şeklinde yazılabilen tüm dizgilerin kümesidir. O halde wi'ler ∑'deki tek tek semboller olduğundan, ∑\* başlangıçta tanımlandığı gibi sembolleri ∑'de olan tüm sonlu dizgelerin kümesidir.

Başka bir uç örnek için, ∅\* = {e} olduğunu gözlemleyin. Yukarıdaki tanımda L = ∅ olsun. k ≥ 0 ve w1,..., wk ∈ L olan tek olası w1 o w2 o ... o wk birleşimi k = 0 olan birleşimdir, yani sıfır dizginin birleşimidir; bu durumda L\*'nin tek üyesi e!

Son bir örnek olarak, eğer L {w ∈ {0, 1}\*: w'de eşit olmayan sayıda 0 ve 1 vardır} dili ise, L\* = {0, 1}\* olduğunu gösterelim. Bunu görmek için, öncelikle herhangi bir L1 ve L2 dili için, eğer L1 ⊆ L2 ise, Kleene yıldızının tanımından da anlaşılacağı üzere L1\* ⊆ L2\* olduğuna dikkat edin. İkinci olarak, {0, 1} ⊆ L, çünkü 0 ve 1'in her biri, bir dizge olarak kabul edildiğinde, eşit olmayan sayıda 0 ve 1'e sahiptir. Dolayısıyla {0, 1}\* ⊆ L\*; ancak L\* ⊆ {0, 1}\* tanım gereği, yani L\* = {0, 1}\*.

LL\* dili için L+ yazıyoruz. Eşdeğer olarak, L+ şu dildir:



L+'nın L'nin birleştirme fonksiyonu altında kapanışı olarak düşünülebileceğine dikkat edin. Yani, L+, L'yi ve L'deki dizelerin birleşimi olan tüm dizeleri içeren en küçük dildir.

1.8 Dillerin Sonlu Gösterimi

Hesaplama teorisindeki temel konulardan biri, dillerin sonlu belirtimlerle temsil edilmesidir. Doğal olarak, herhangi bir sonlu dil, dildeki tüm dizgilerin kapsamlı bir şekilde sayılmasıyla sonlu temsile uygundur. Bu konu ancak sonsuz diller düşünüldüğünde zorlayıcı bir hal alır.

“Bir dilin sonlu temsili” kavramı hakkında biraz daha kesin olalım. Belirtilmesi gereken ilk nokta, böyle bir temsilin kendisinin bir dizge, bazı alfabe ∑ üzerinde sonlu bir sembol dizisi olması gerektiğidir. İkinci olarak, kesinlikle farklı dillerin farklı temsilleri olmasını isteriz, aksi takdirde temsil terimi pek uygun sayılmaz. Ancak bu iki gereklilik zaten sonlu temsil olanaklarının ciddi şekilde sınırlı olduğunu ima eder. Çünkü bir ∑ alfabesi üzerindeki dizelerin ∑\* kümesi sayılabilir şekilde sonsuzdur, dolayısıyla dillerin olası temsillerinin sayısı sayılabilir şekilde sonsuzdur. (Bu, mevcut sembollerin toplam sayısı sayılabilir şekilde sonsuz olduğu sürece, belirli bir alfabe ∑ kullanmak zorunda olmasaydık bile doğru kalırdı). Öte yandan, belirli bir ∑ alfabesi üzerindeki tüm olası dillerin kümesi - yani 2^∑\* - sayılamayacak kadar sonsuzdur, çünkü 2N ve dolayısıyla herhangi bir sayılabilir sonsuz kümenin kuvvet kümesi sayılamayacak kadar sonsuz değildir. Sadece sayılabilir sayıda temsil ve sayılamaz sayıda temsil edilecek şeyle, tüm dilleri sonlu olarak temsil edemeyiz. Bu nedenle, umabileceğimiz en fazla şey, en azından daha ilginç dillerin bazıları için şu ya da bu türden sonlu temsiller bulmaktır.

Bu, hesaplama teorisindeki ilk sonucumuzdur: Dilleri temsil etmek için kullandığımız yöntemler ne kadar güçlü olursa olsun, temsillerin kendileri sonlu olduğu sürece yalnızca sayılabilir sayıda dil temsil edilebilir. Toplamda sayılamayacak kadar çok dil olduğu için, bunların büyük çoğunluğu kaçınılmaz olarak herhangi bir sonlu temsil şeması altında gözden kaçacaktır.

Elbette bu, bu doğrultuda söylememiz gereken son şey değil. Dilleri betimlemenin ve temsil etmenin, her biri bir öncekinin yapamadığı dilleri betimleme kapasitesine sahip olması anlamında bir öncekinden daha güçlü olan çeşitli yollarını tanımlayacağız. Bu hiyerarşi, tüm bu sonlu temsil yöntemlerinin az önce açıklanan nedenlerden dolayı kaçınılmaz olarak kapsam açısından sınırlı olduğu gerçeğiyle çelişmez.

Ayrıca, incelediğimiz çeşitli temsil yöntemleriyle temsil edilemeyen belirli dilleri sergilemenin yollarını da türetmek isteyeceğiz. Diller dünyasında bu tür temsil edilemeyen çok sayıda örnek olduğunu biliyoruz, ancak garip bir şekilde belki de bir tanesini yakalamak, sergilemek ve belgelemek son derece zor olabilir. Köşegenleştirme argümanları eninde sonunda bize burada yardımcı olacaktır.

Sonlu temsiller çalışmamıza başlamak için, önceki bölümde açıklanan işlemler kullanılarak dillerin nasıl oluşturulabileceğini tanımlayan ifadeleri - sembol dizilerini - ele alıyoruz.

Örnek 1.8.1: L = {w ∈ {0, 1}\*: w iki ya da üç kez 1'e sahiptir, bunlardan birincisi ve ikincisi ardışık değildir} olsun. Bu dil sadece tekil kümeler ve U, o ve \* sembolleri kullanılarak aşağıdaki gibi tanımlanabilir:

metin, yazı tipi, tipografi, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Kabaca konuşmak gerekirse, Örnek 1.8.1'deki L gibi bir ifade düzenli ifade olarak adlandırılır. Yani, bir düzenli ifade, bir dili yalnızca tek semboller ve ø, belki U ve \* sembolleriyle birlikte, muhtemelen parantezlerin yardımıyla tanımlar. Ancak, hakkında konuştuğumuz ifadeleri ve bunları tartışmak için kullandığımız “matematiksel İngilizceyi” düzgün tutmak için, oldukça dikkatli adım atmalıyız. Bu kitapta belirli işlemler ve kümeler için kullanılan U, \* ve ø yerine, tıpkı daha önceki örneklerde kullanılan a, b ve 0 sembolleri gibi, şimdilik tamamen anlamsız olarak kabul edilmesi gereken özel U, \* ve ø sembollerini kullanıyoruz. Aynı şekilde, matematik yapmak için kullandığımız parantezler ( ve) yerine özel semboller (ve) getiriyoruz. Bir ∑\* alfabesi üzerindeki düzenli ifadeler, ∑ U {(,), ø, U,\*} alfabesi üzerinde aşağıdaki gibi elde edilebilen tüm dizelerdir.

(1) ø ve ∑'nin her bir üyesi bir düzenli ifadedir.

(2) Eğer α ve β düzenli ifadeler ise, o zaman (αβ) da öyledir.

(3) Eğer α ve β düzenli ifadeler ise, o zaman (αUβ) da öyledir.

(4) Eğer α bir düzenli ifade ise, α\* da öyledir.

(5) (1)'den (4)'e kadar takip etmediği sürece hiçbir şey düzenli ifade değildir.

Her düzenli ifade, U ve \* sembollerinin küme birleşimi ve Kleene yıldızı olarak ve ifadelerin yan yana gelmesinin birleştirme olarak yorumlanmasına göre bir dili temsil eder. Biçimsel olarak, düzenli ifadeler ve temsil ettikleri diller arasındaki ilişki bir L fonksiyonu ile kurulur, öyle ki α herhangi bir düzenli ifade ise, L(α) α tarafından temsil edilen dildir. Yani, L dizelerden dillere bir fonksiyondur. L fonksiyonu aşağıdaki gibi tanımlanır.

(1) L(ø) = ø ve her α ∈ ∑ için L(α) = {α}.

(2) Eğer α ve β düzenli ifadeler ise, L((αβ)) = L(α)L(β).

(3) Eğer α ve β düzenli ifadeler ise, o zaman L((αUβ)) = L(α) U L(β).

(4) Eğer a bir düzenli ifade ise, o zaman L(α\*) = L(α)\*.

İfade 1, tek bir sembolden oluşan her düzenli ifade α için L(α)'yı tanımlar; daha sonra (2)'den (4)'e kadar, daha küçük uzunluktaki bir veya iki düzenli ifade α' için L(α') cinsinden belirli uzunluktaki düzenli ifadeler için L(α)'yı tanımlar. Böylece her düzenli ifade bu şekilde bir dil ile ilişkilendirilir.

metin, yazı tipi, ekran görüntüsü, cebir içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Örnek 1.8.3: (c\* (aU(bc\*))\*) hangi dili temsil eder? Bu düzenli ifade, ac alt dizesine sahip olmayan {a,b,c) üzerindeki tüm dizelerin kümesini temsil eder. Açıkçası, L((c\*(aU(bc\*))\*) içindeki hiçbir dizgi ac alt dizgisini içeremez, çünkü böyle bir dizgide a'nın her geçişi ya dizginin sonundadır ya da a'nın başka bir geçişi tarafından takip edilir ya da b'nin bir geçişi tarafından takip edilir.

Öte yandan, w dizesi ac alt dizesi içermeyen bir dize olsun. O zaman w sıfır veya daha fazla c ile başlar. Eğer bunlar çıkarılırsa, sonuç ac alt dizesi olmayan ve c ile başlamayan bir dizedir. Böyle herhangi bir dizge L((aU(bc\*))'dedir; çünkü soldan sağa, a'lar, b'ler ve c'lerden oluşan bir dizi olarak okunabilir, c'lerin herhangi bir bloğu b'lerin hemen ardından gelir (a'ların ardından gelmez ve dizenin başında değildir). Bu nedenle w ∈ L((c\*(aU(bc\*))\*).

Örnek 1.8.4: (0\*U(((0\*(1U(11)))((00\*)(1U(11)))\*)0\*)) {0, 1} üzerinde 111 alt dizesine sahip olmayan tüm dizelerin kümesini temsil eder.

Bir düzenli ifade ile temsil edilebilen her dil, sonsuz sayıda düzenli ifade ile temsil edilebilir. Örneğin, α ve (αUØ) her zaman aynı dili temsil eder; ((αUβ)Uγ) ve (αU(βUγ)) da öyle.

Küme birleştirme ve birleştirme çağrışımsal işlemler olduğundan - yani, tüm L1, L2, L3 için (L1 U L2) U L3 = L1 U (L2 U L3) olduğundan ve birleştirme için de aynı şey geçerli olduğundan - normalde düzenli ifadelerdeki ekstra (ve) sembollerini atlarız; örneğin, aUbUc “resmi olarak” düzenli ifade olmasa da düzenli ifade olarak ele alırız. Başka bir örnek için, Örnek 1.8.4'ün düzenli ifadesi 0\*U0\*(1U11)(00\*(1U11))\*0\* şeklinde yeniden yazılabilir.

Dahası, düzenli ifadelerin ve temsil ettikleri dillerin biçimsel ve açık bir şekilde tanımlanabileceğini gösterdiğimize göre, karışıklığa yol açmayacaksa, düzenli ifadeler ile diller hakkında konuşurken kullandığımız “matematiksel İngilizce” arasındaki ayrımı bulanıklaştırmaktan çekinmeyiz.

Böylece bir noktada a\*b\*'nin belli sayıda a'yı belli sayıda b'nin takip etmesinden oluşan tüm dizgelerin kümesi olduğunu söyleyebiliriz - tam olarak {a}\* o {b}\* yazmamız gerekirdi. Başka bir noktada, a\*b\*'nin bu kümeyi temsil eden düzenli bir ifade olduğunu söyleyebiliriz; bu durumda, kesin olmak gerekirse, (a\*b\*) yazmamız gerekirdi.

Bir alfabe ∑ üzerindeki düzenli diller sınıfı, ∑ üzerindeki bazı düzenli ifadeler α için L = L(α) olacak şekilde tüm L dillerinden oluşacak şekilde tanımlanır. Yani, düzenli diller düzenli ifadelerle tanımlanabilen tüm dillerdir. Alternatif olarak, düzenli diller kapanışlar açısından da düşünülebilir. ∑ üzerindeki düzenli diller sınıfı, tam olarak diller kümesinin kapanışıdır

yazı tipi, beyaz, hat sanatı, kaligrafi, tipografi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Düzenli ifadelerin bazı önemsiz ve ilginç dilleri tanımladığını zaten görmüştük. Ne yazık ki, başka yollarla çok basit tanımları olan bazı dilleri düzenli ifadelerle tanımlayamıyoruz. Örneğin, {0^n1^n: n ≥ 0}'ın düzenli olmadığı Bölüm 2'de gösterilecektir. Şüphesiz dillerin sonlu gösterimine ilişkin herhangi bir teori en azından bu gibi basit dilleri barındırmak zorunda olacaktır. Bu nedenle düzenli ifadeler genel olarak yetersiz bir belirtim yöntemidir.

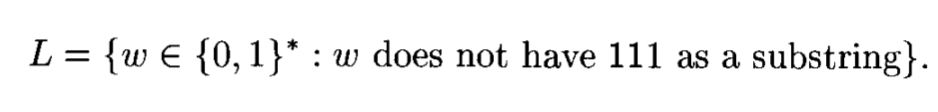
Dilleri sonlu olarak belirlemek için genel bir yöntem arayışında, genel şemamıza geri dönebiliriz:

yazı tipi, metin, beyaz, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Ancak hangi P özelliklerini zorunlu kılmalıyız? Örneğin, önceki özellikleri, “w eşit sayıda 1‘in takip ettiği bir dizi 0’dan oluşur” ve “w'de hiç 111 yoktur” gibi bariz adaylar yapan nedir? Okuyucu doğru cevap üzerinde düşünebilir; ancak şimdilik algoritmik özelliklere ve sadece bunlara izin verelim. Yani, dizgelerin bir P özelliğinin bir dilin belirtimi olarak kabul edilebilmesi için, verilen bir dizgenin dile ait olup olmadığına karar veren bir algoritma olmalıdır.

Bazı L dilleri için “w dizesi L'nin bir üyesi midir?” şeklindeki soruları yanıtlamak üzere özel olarak tasarlanmış bir algoritmaya dil tanıma cihazı denecektir. Örneğin, bir dili tanımak için bir cihaz:



soldan sağa doğru her seferinde bir sembol olmak üzere dizeleri okuyarak şu şekilde çalışabilir:

Sıfırdan başlayan ve girdide her 0 ile karşılaşıldığında sıfıra ayarlanan bir sayım tutun; girdide her 1 ile karşılaşıldığında bir ekleyin; sayım üçe ulaşırsa Hayır yanıtı ile durun ve sayım üçe ulaşmadan tüm dize okunursa Evet yanıtı ile durun.

Bir dili belirtmek için alternatif ve biraz ortogonal bir yöntem, dildeki genel bir örneğin nasıl üretildiğini tanımlamaktır. Örneğin, (e U b U bb)(a U ab U abb)\* gibi bir düzenli ifade, bir dilin üyelerini üretmenin bir yolu olarak görülebilir:

L'nin bir üyesini üretmek için önce ya hiç ya da b ya da bb yazın; sonra a ya da ab ya da abb yazın ve bunu sıfır dahil istediğiniz sayıda yapın; L'nin tüm üyeleri ve sadece üyeleri bu şekilde üretilebilir.

Bu tür dil üreteçleri algoritma değildir, çünkü soruları yanıtlamak için tasarlanmamışlardır ve ne yapacakları konusunda tamamen açık değildirler (a, ab veya abb'den hangisinin yazılacağını nasıl seçeceğiz?) Ancak yine de dilleri temsil etmek için önemli ve yararlı araçlardır. Her ikisi de sonlu dil belirtimlerinin bir türü olan dil tanıma aygıtları ve dil üreteçleri arasındaki ilişki, bu kitabın bir diğer önemli konusudur.

ÖDEV : 1.8.1 – 1.8.2 -1.8.3 – 1.8.5