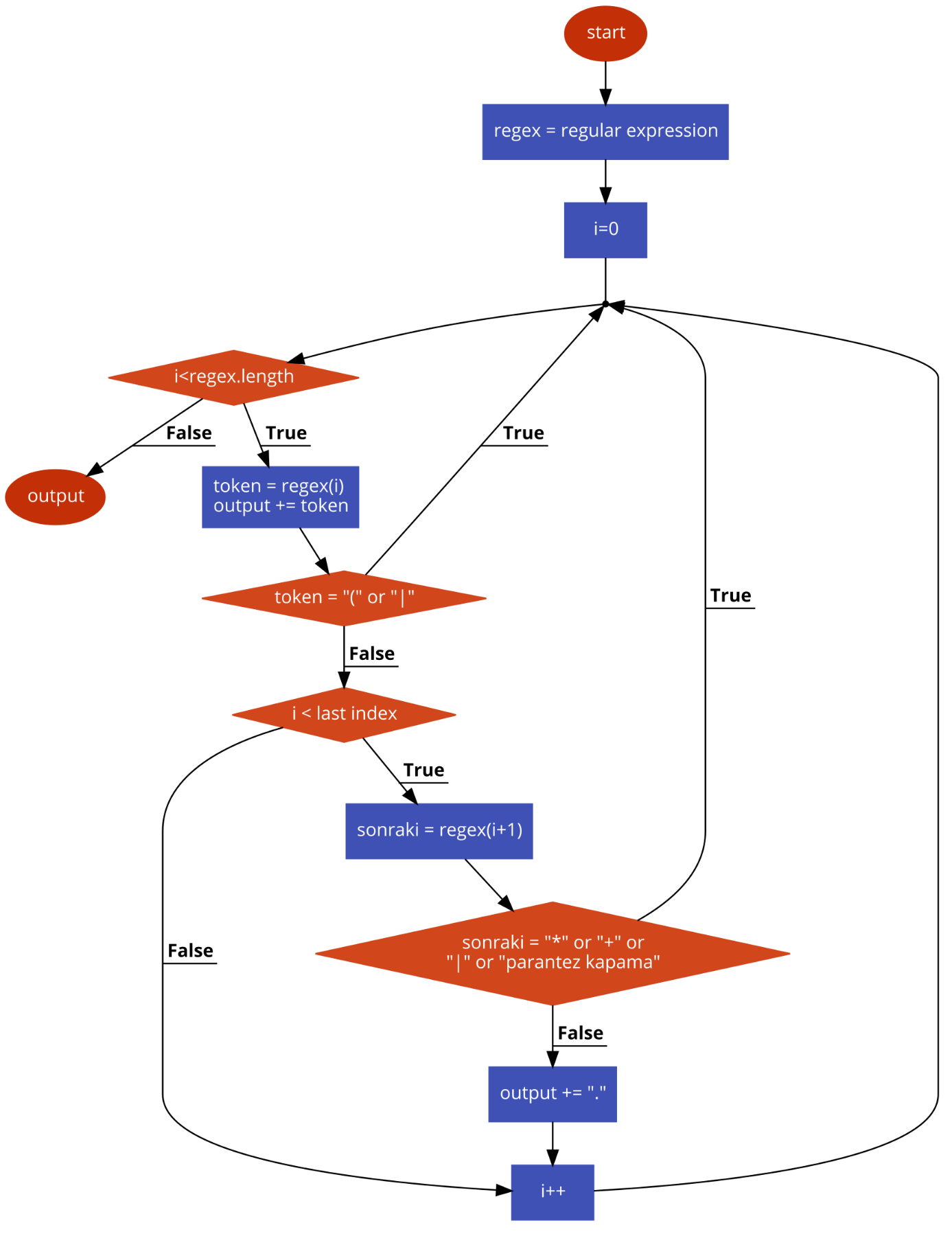
Regular Expression Engine

**Concatenation Operatörü Ekleme İşlemi** :



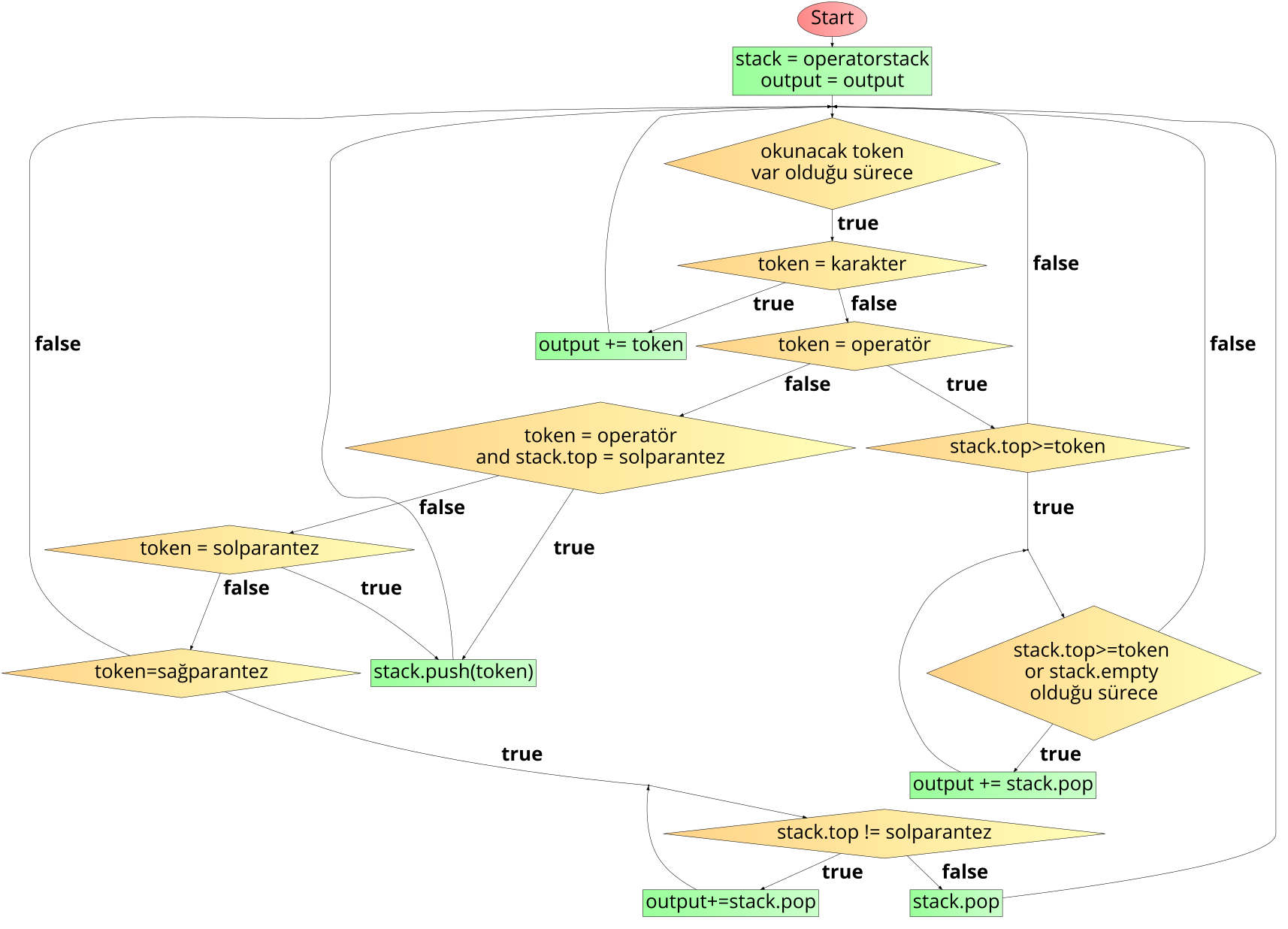
**Concatenation Operatörü Ekleme İşleminin Açıklaması:**

İnput olarak girilen Regular Expression’a birleştirme operatörü (“.”) ekleme sebebimiz; NFA fonksiyonumuzda kullandığımız algoritmanın, nerelerde birleştirme işlemi yapması gerektiğini anlamasına yardımcı olmaktır.

Bu algoritma kısaca; kullanıcıdan regular expression’ı alıyor ve bir for döngüsü ile girilen RE’nin her bir indexine tek tek bakıyor. Son indexe gelene kadar; Eğer (union (“|”) operatörü ve parantez açma hariç) herhangi bir inputtan sonra herhangi bir sembol görmüşse output çıktısına “.” operatörünü ekliyor. Son olarak da; inputun birleştirme operatörü eklenmiş halini “output” değişkeni ile döndürüyor.

**Örnek**: a(b|c)d 🡪 a.(b|c).d

**Regular Expression Postfix Dönüştürme İşlemi :**



**Regular expression postfix dönüştürme işleminin açıklaması :**

Birleştirme operatörü eklenmiş regular expression’ı parametre olarak alıp postfix notasyonuna çeviriyor. Daha sonra kullanmak için bir tane output stringi ve bir tane de operatörleri içeren bir stack oluşturuyor. Önce regexin her indexine tek tek bakıyor:

* Eğer index operatör harici bir karakterse output stringine ekliyor.
* Eğer index operatör ise stackin en üstündeki operatörün önceliği elimizdeki operatörden yüksek veya eşit ise operatör stackinin en üstündeki operatörü alıp output stringine ekliyor. Bu işlemi elimizdeki operatörden daha küçük önceliğe sahip bir operatör önüne gelene kadar ya da operatör stacki boşalana kadar yapıyor.
* Eğer index bir operatörse ve operatör stackinin en üstündeki karakter “parantez açma” karakteri ise operatörü direkt olarak operatör stackinin içine atıyor.
* Eğer index “parantez açma” karakteri ise direkt olarak operatör stackinin içine atıyor.
* Eğer index “parantez kapama” ise; parantez açma sembolünü görene kadar operatör stackinin elemanlarını sırayla output stringine ekliyor. Parantez açma sembolünü gördüğünde ise output stackinden parantez açma sembolünü kaldırıyor ve algoritmaya devam ediyor.

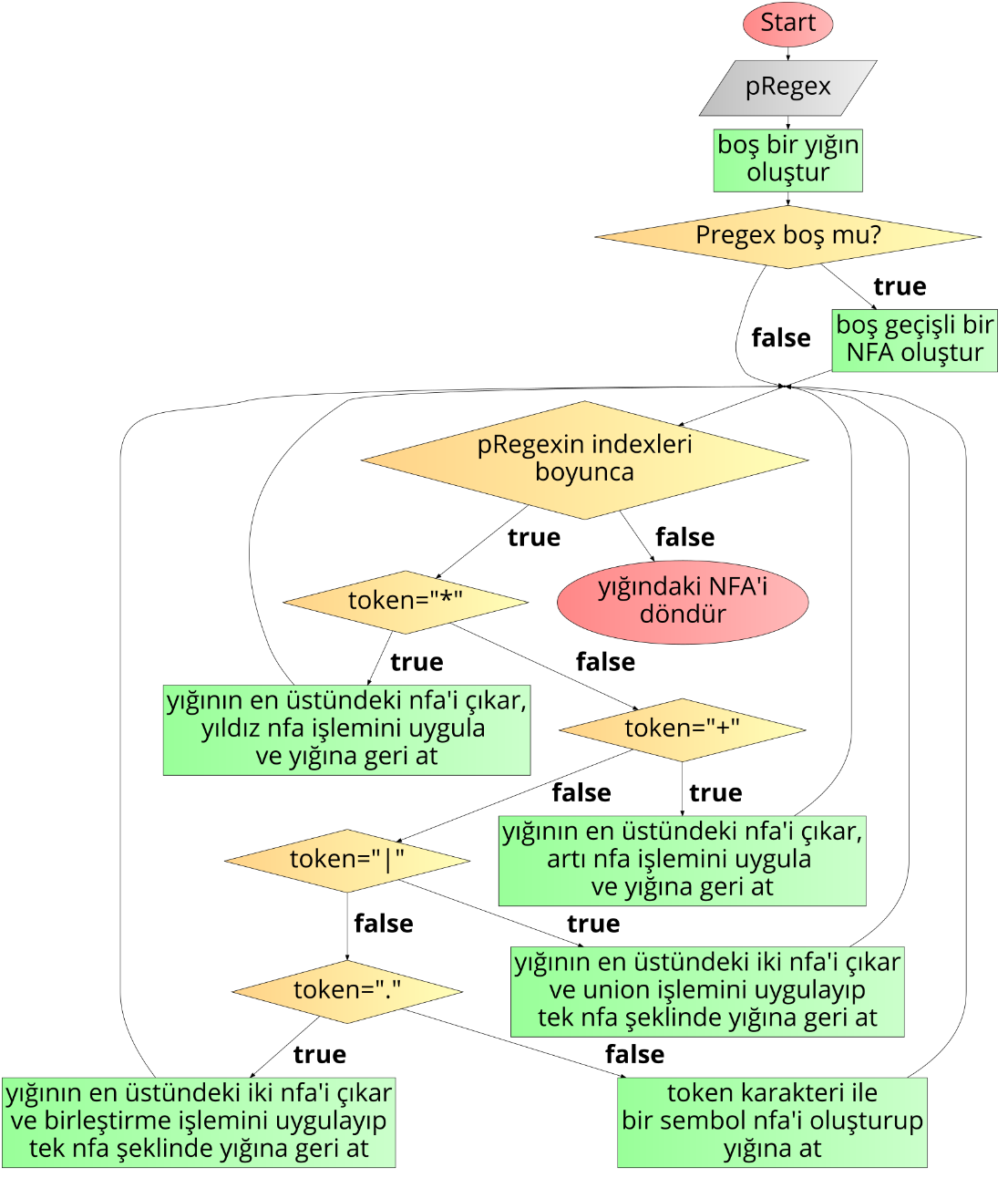
Operatör stackinin içindekileri sırasıyla output stringine ekliyor.

Regexin indexleri bittiğinde ise output stringini döndürüyor.

Postfix dönüşümü yapma sebebimiz kısaca; parantezlerden kurtulma isteğidir. Parantezlerden kurtulduğumuza göre artık Thompson algoritmasını kullanarak bu regular expression’ın NFA’ini oluşturabiliriz.

**Örnek :** a.(b|c).d 🡪 abc|.d.

**Regular Expression’dan NFA oluşturma işlemi : (Pregex = postfix edilmiş RE)**



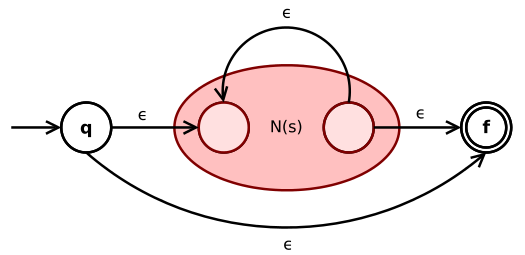
**Regular Expression’dan NFA oluşturma işleminin açıklanması :**

Bu işlem algoritma olarak en düz işlem sayılabilir. RE’den NFA oluşturmak için tek yapmamız gereken gördüğümüz indexteki karaktere göre işlem belirlemek. Fonksiyon için bir tane stack (yığın) kullanıyoruz.

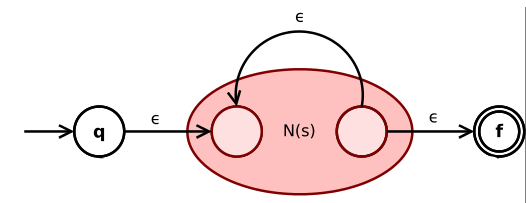
* Eğer girilen RE boş ise boş geçişli bir NFA oluşturuyor.

**( E = boş aktarım)**

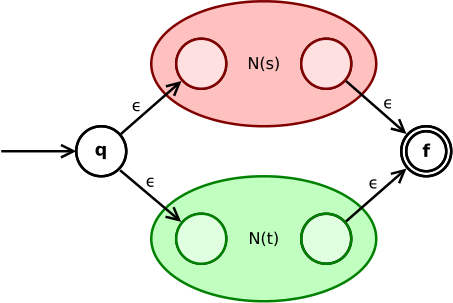
* Eğer indexte “\*” (yıldız, Kleene star,sıfır ya da daha fazla) karakteri var ise yığının en üstündeki nfa’i çıkarıp, yıldız işlemini uygulayan fonksiyonu çağırıyor ve yıldız işlemi uygulanmış nfa’i yığına geri atıyor.

**( E = boş aktarım)**

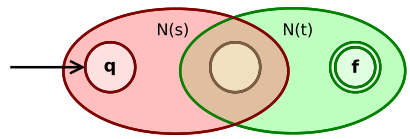
* Eğer indexte “+” (artı,bir ya da daha fazla) karakteri var ise yığının en üstündeki NFA’i çıkarıp artı işlemini uygulayan fonksiyonu çağırıyor ve artı işlemi uygulanmış NFA’i yığına geri atıyor.

**( E = boş aktarım)**

* Eğer indexte “|” (union, OR) karakteri var ise yığının en üstündeki iki tane NFA’i çıkarıp, union işlemini bu iki NFA’e uyguluyor ve oluşan tek NFA’i yığına geri atıyor.

 **( E = boş aktarım)**

* Eğer indexte “.” (Concatenation, Birleştirme) karakteri var ise yığının en üstündeki iki tane NFA’i çıkarıp, birleştirme işlemini bu iki NFA’e uyguluyor ve oluşan tek NFA parçasını yığına geri atıyor.

****

* Eğer indexte operatör harici bir karakter var ise program, bu karakteri, herhangi bir alfabe parçası olarak algılayıp, o parçaya ait başlangıç ve bitiş durumları olan bir NFA oluşturuyor ve oluşturduğu NFA parçasını yığına atıyor.

** ( “a” inputu için)**

Son olarak ise RE’nin indexleri bittiğinde yığında kalan son NFA’i döndürüyor.

Görüldüğü gibi Thompson algoritmasında herhangi bir işlem yapılacağında her zaman bir tane başlangıç bir tane de bitiş durumu oluyor ve fazladan en fazla 2 tane durum oluşturulabiliyor.

**Örnek :** Yukarıda verdiğimiz “a(b|c)d” örneğini Thompson algoritması ile NFA’e dönüştürelim.

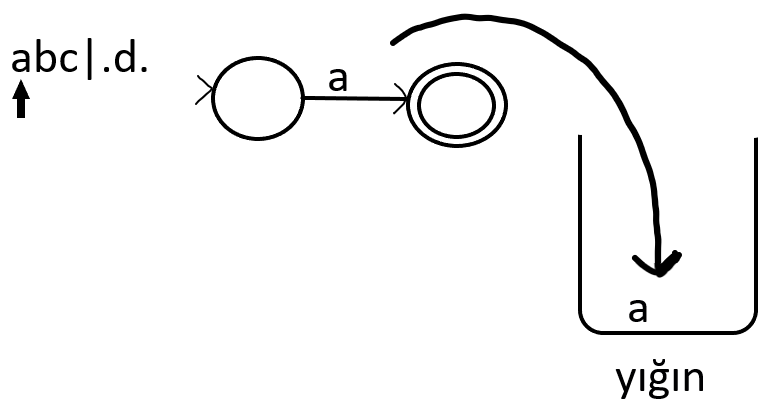
Öncelikle Thompson algoritmasında bu örneği kullanabilmek için parantezlerden kurtulmamız gerekiyor. Parantezlerden kurtulmak için Postfix dönüşümü yapmamız gerekiyor. Postfix dönüşümü için ise birleştirme operatörü olması gereken yerlere gerçekten birleştirme operatörünü eklememiz gerekiyor.

Yani önce a(b|c)d 🡪 a.(b|c).d böyle bir dönüşüm yapıyoruz.

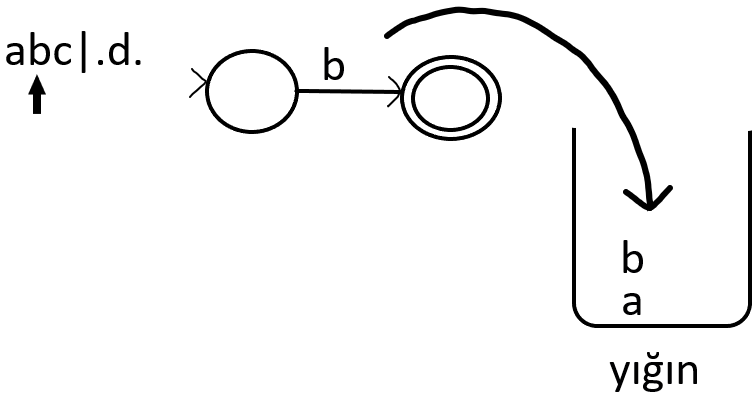
Sonra a.(b|c).d 🡪 abc|.d. bu şekilde postfixe dönüştürüyoruz.

**abc|.d.** bu ifadenin her bir indexine bakarak ona göre NFA oluşturmalıyız.

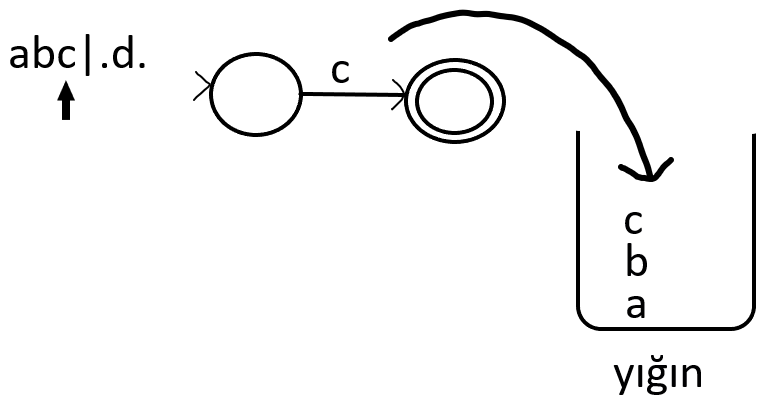
* İlk index “a” karakteri. O zaman bu karaktere uygun NFA’i oluşturup yığına atıyoruz.



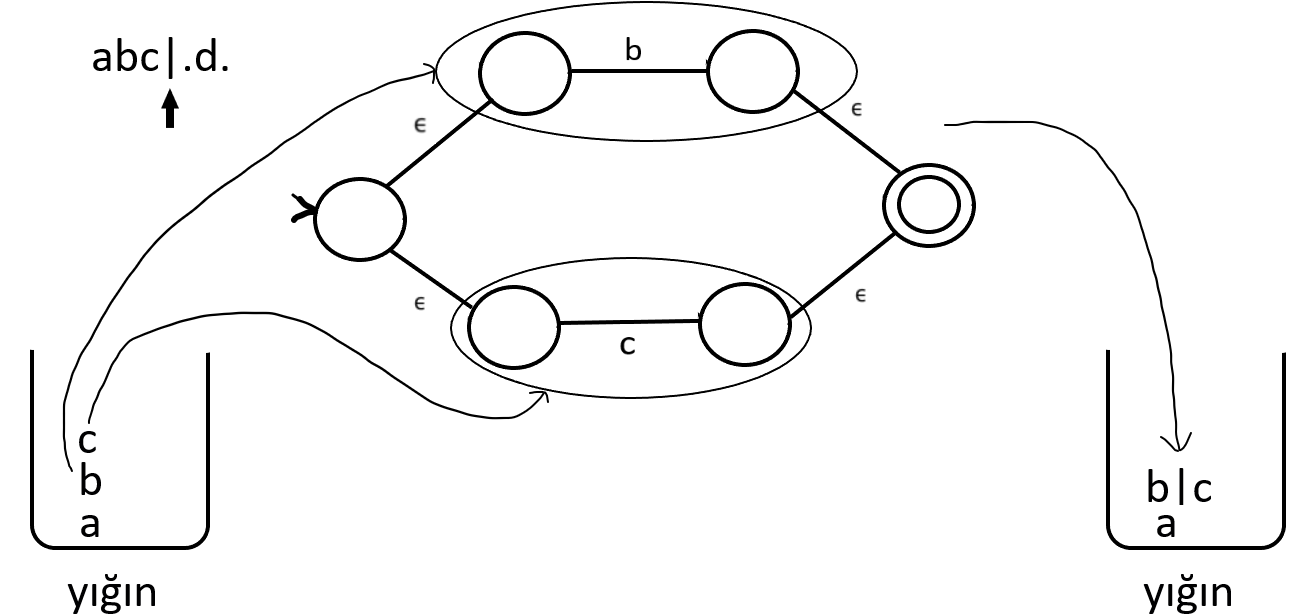
* İkinci index “b” karakteri. Bu karaktere uygun NFA’i oluşturup yığına atıyoruz.



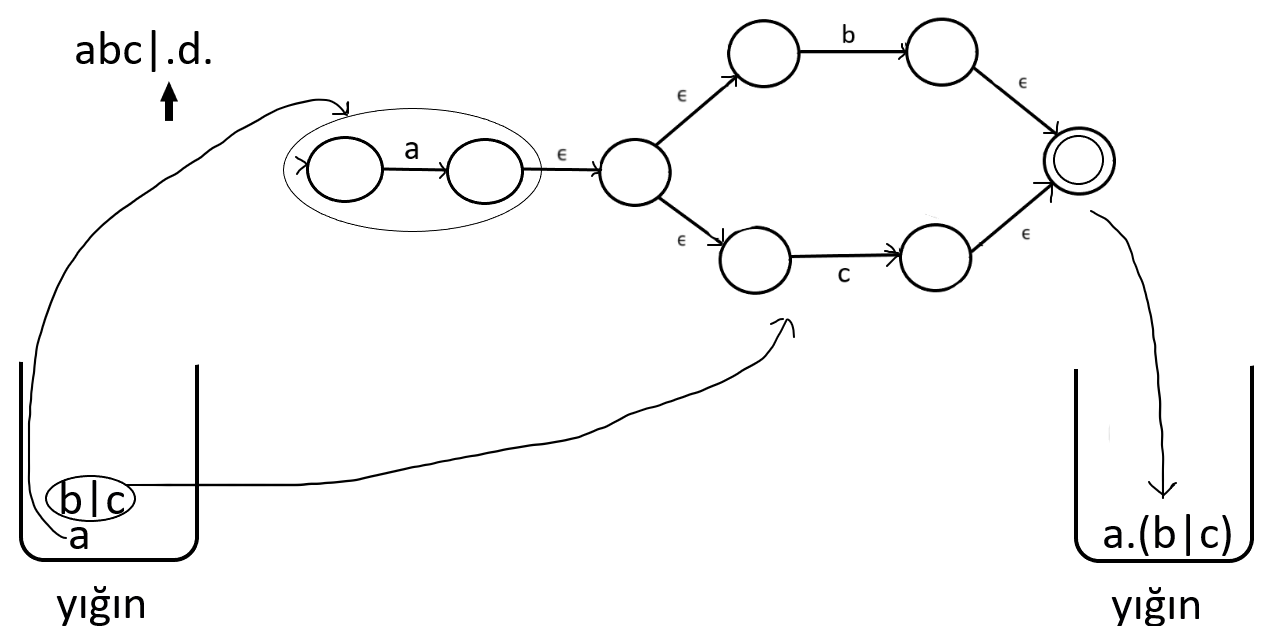
* Üçüncü index “c” karakteri. Bu karaktere uygun NFA’i oluşturup yığına atıyoruz.



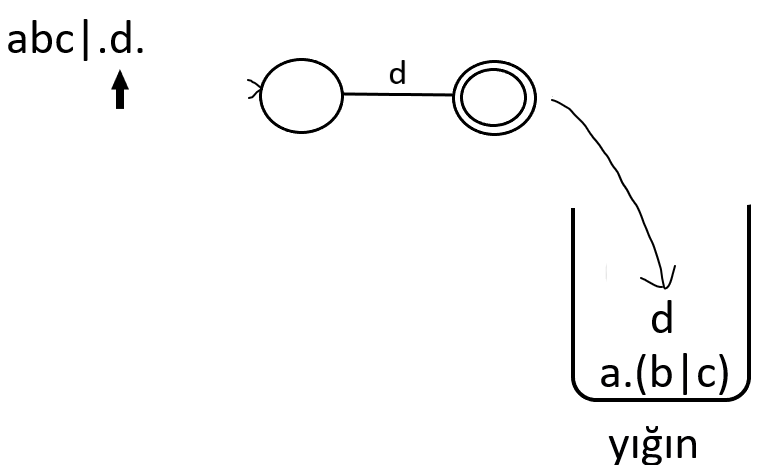
* Dördüncü index “|” karakteri. Yığının üstünden iki tane NFA’i çıkartıp Union işlemi uyguluyoruz ve oluşan tek NFA’i geri yığına atıyoruz.



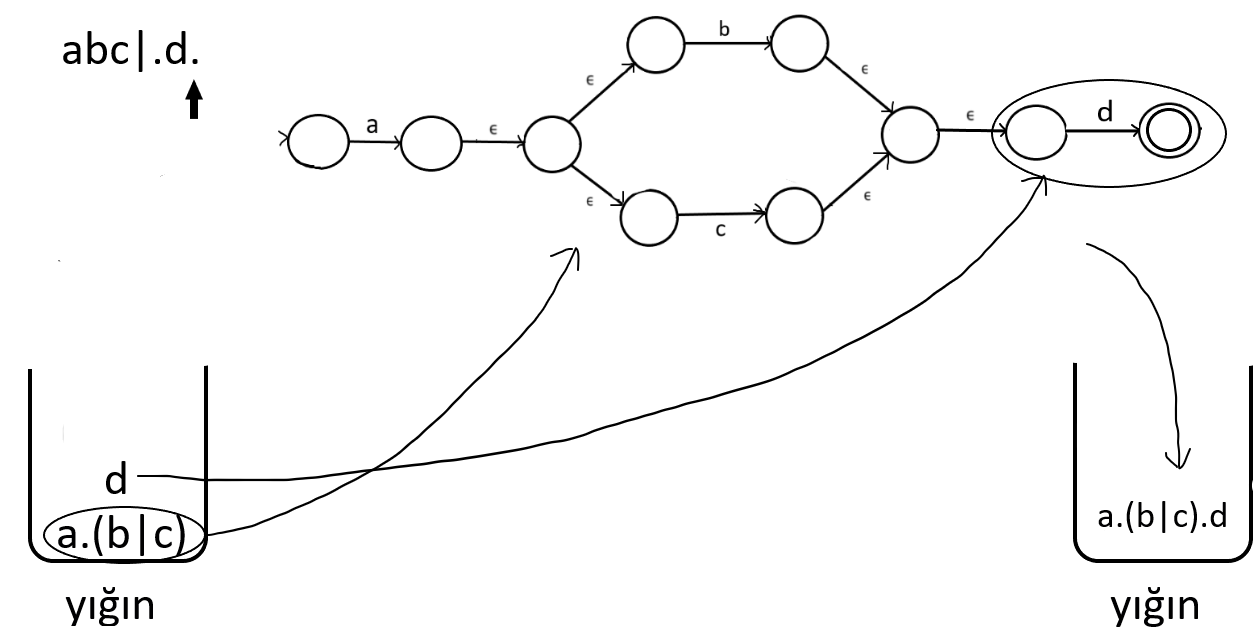
* Beşinci index “.” Karakteri. Yığının üstünden iki tane NFA’i çıkartıp Concatenation işlemi uyguluyoruz ve oluşan tek NFA’i tekrar yığına atıyoruz.



* Altıncı index “d” karakteri. Bu karaktere uygun NFA’i oluşturup yığına atıyoruz.

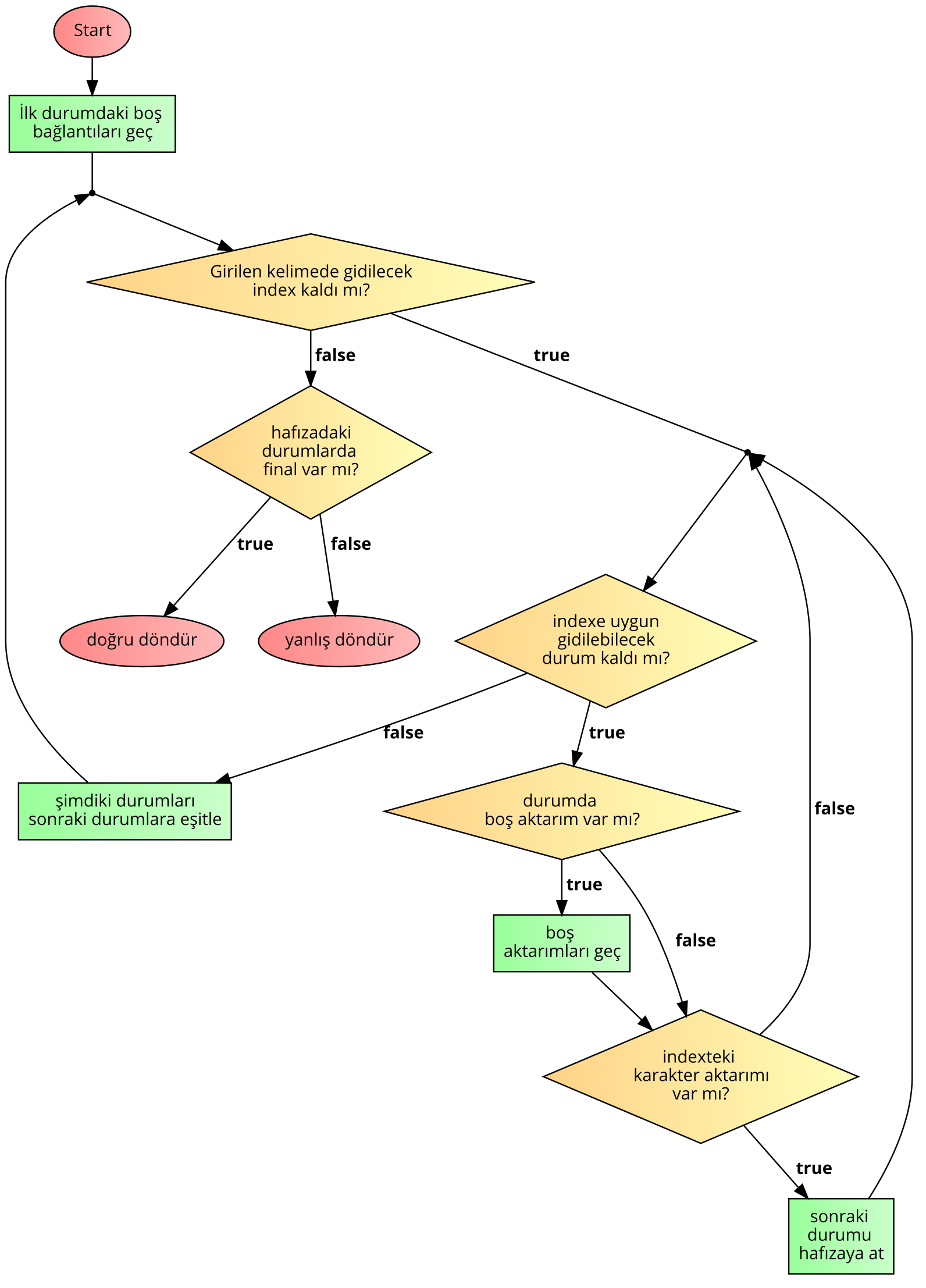


* Yedinci ve son index “.” karakteri. Yığının üstünden iki tane NFA’i çıkartıp Concatenation işlemi uyguluyoruz ve oluşan tek NFA’i tekrar yığına atıyoruz.



En sonda yığında “a.(b|c).d” NFA’ i kalıyor yani en başta NFA’ini oluşturmak istediğimiz RE’nin aynısı. Bu NFA’i metinde pattern arama işlemi için kullanacağız.

**Metinde pattern arama işlemi:**

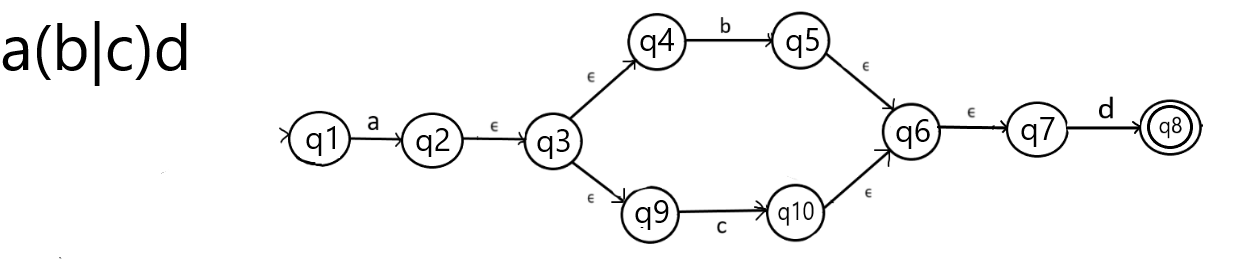
****

**Metinde pattern arama işleminin açıklanması:**

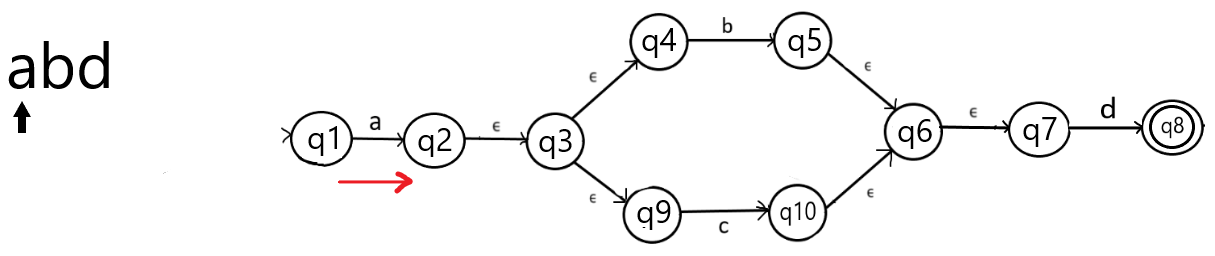
Arama işlemi aynı anda birden fazla durumda bulunabilmeye yarıyor. Bu da programın time complexity’sini büyük oranda azaltıyor.

Fonksiyon önce başlangıç durumunda boş aktarımlar var mı yok mu kontrol ediyor. Eğer varsa o aktarımlardan direkt geçiyor. Sonra girilen kelimedeki her bir karakter için tüm stateleri kontrol ediyor ve gidebildiği son yere kadar gidiyor. Eğer gittiği son yer final durumu ise “true” döndürüyor. Değil ise “false” döndürüyor.

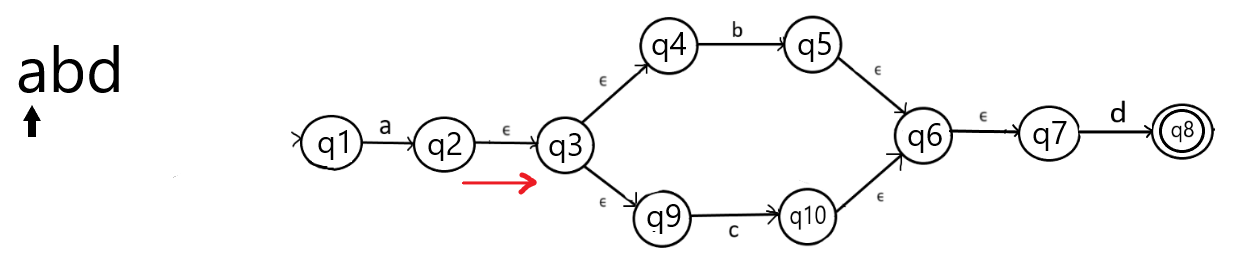
**Örnek:**



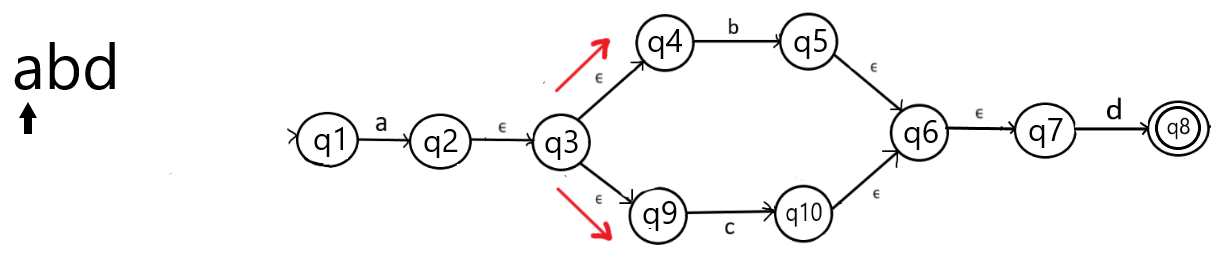
Yukarıda oluşturduğumuz NFA ile “abd” kelimesinde arama yapalım.



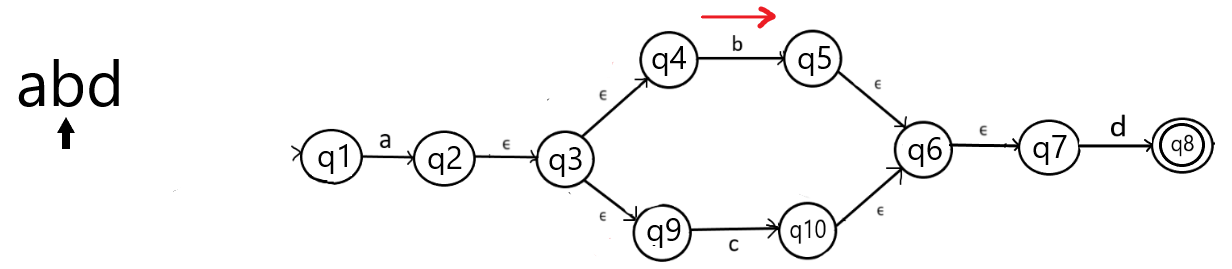
q1’de iken “a” geldiğinde q2 durumuna gidiyor.



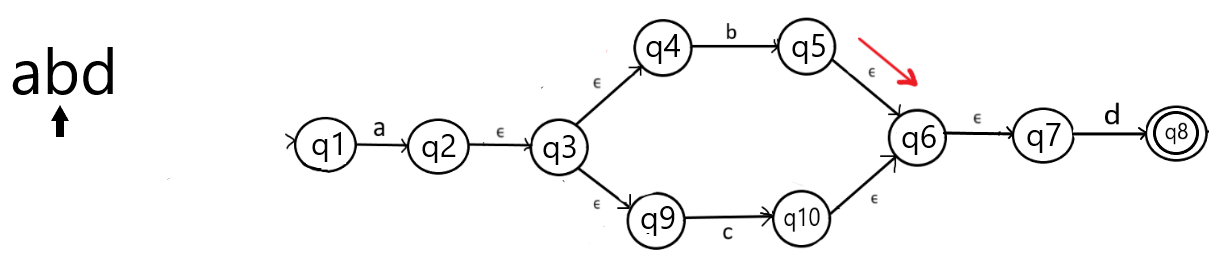
q2’den boş aktarım olduğu için input farketmeden q3’ü depoluyor.



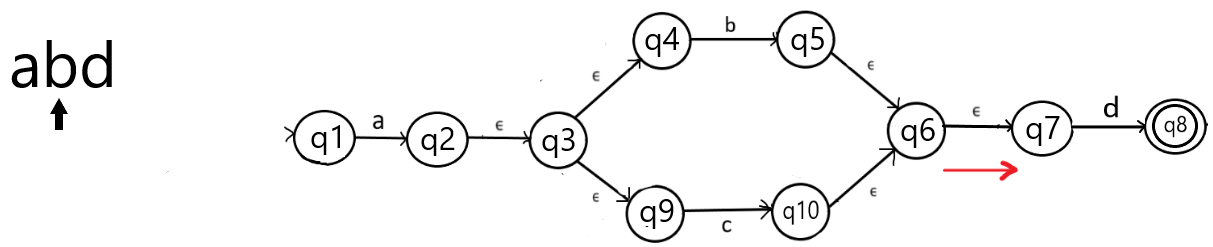
q3 durumdan iki farklı yere boş bağlantı olduğu için ikisine aynı anda gidiyor. Yani hafızaya hem q4’ü hem de q9’u atıyor.



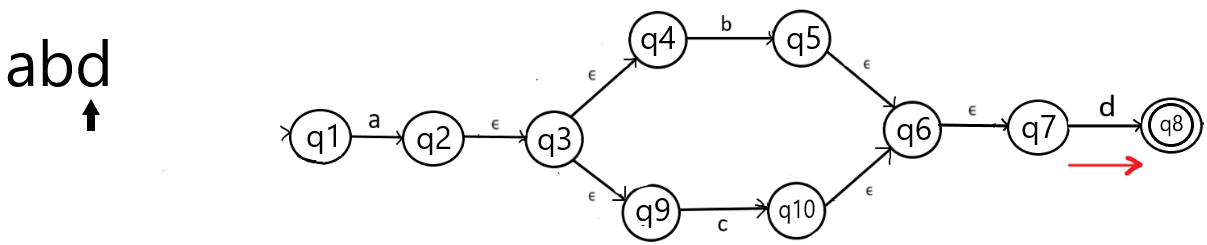
“a” karakterine uygun başka yol kalmadığı için “b” karakterine geçiyor. Hafızada q4 ve q9 olduğu için hem q4’ten hem de q9’dan “b” karakterine sahip bir yol arıyor ve sadece q4’ten buluyor bu yüzden hafızaya q5’i atıyor. Aşağıdaki yol hafızada q9’a kadar kalıyor.



q5’teyken boş aktarım olduğu için hafızaya q6’yı atıyor.

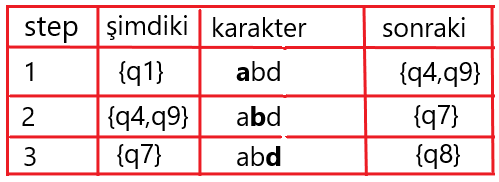


q6’da yine boş aktarım olduğu için q7’yi kaydediyor.



q7 durumundayken ne “b” karakterine ne de boş aktarıma sahip bir yol olmadığı için yeni karaktere yani “d” karakterine geçiyor. q7 durumunda “d” karakteri için bir yol olduğu için hafızaya q8 durumunu ekliyor.

Son olarak bakılacak başka karakter kalmadığı için hafızaya kaydedilen her durumun final durumu olup olmadığı kontrol ediliyor. Eğer final durumuna rastlanırsa “true” döndürülüyor, rastlanmazsa “false” döndürülüyor.



**Kullanıcı Arayüzünün kullanımı:**

Bir tane regular expression girişi yapılması için metin kutusu, bir tane arama yapılacak metin için metin kutusu ve bir tane de işlemi gerçekleştirmesi için “NFA” butonu.

Regular expression girişi ve metin girişi yapıldıktan sonra NFA butonuna tıklandığında metinde bulunan uygun patternler aşağıda sıralanacaktır.

- Aranacak kelimelerin boşluksuz ve satır satır girilmesi zorunludur. Boşluk girildiğinde düzgün çalışmıyor.

- Regex operatörleri olarak sadece "\*","+","|" ve "." operatörleri vardır.

"\*" operatörü işlevi --> sıfır ya da daha fazla

"+" operatörü işlevi --> bir ya da daha fazla

"|" operatörü işlevi --> Union işlemi yapıyor yani mantıksal OR gibi

"." operatörü işlevi --> iki tane regexi birleştiriyor