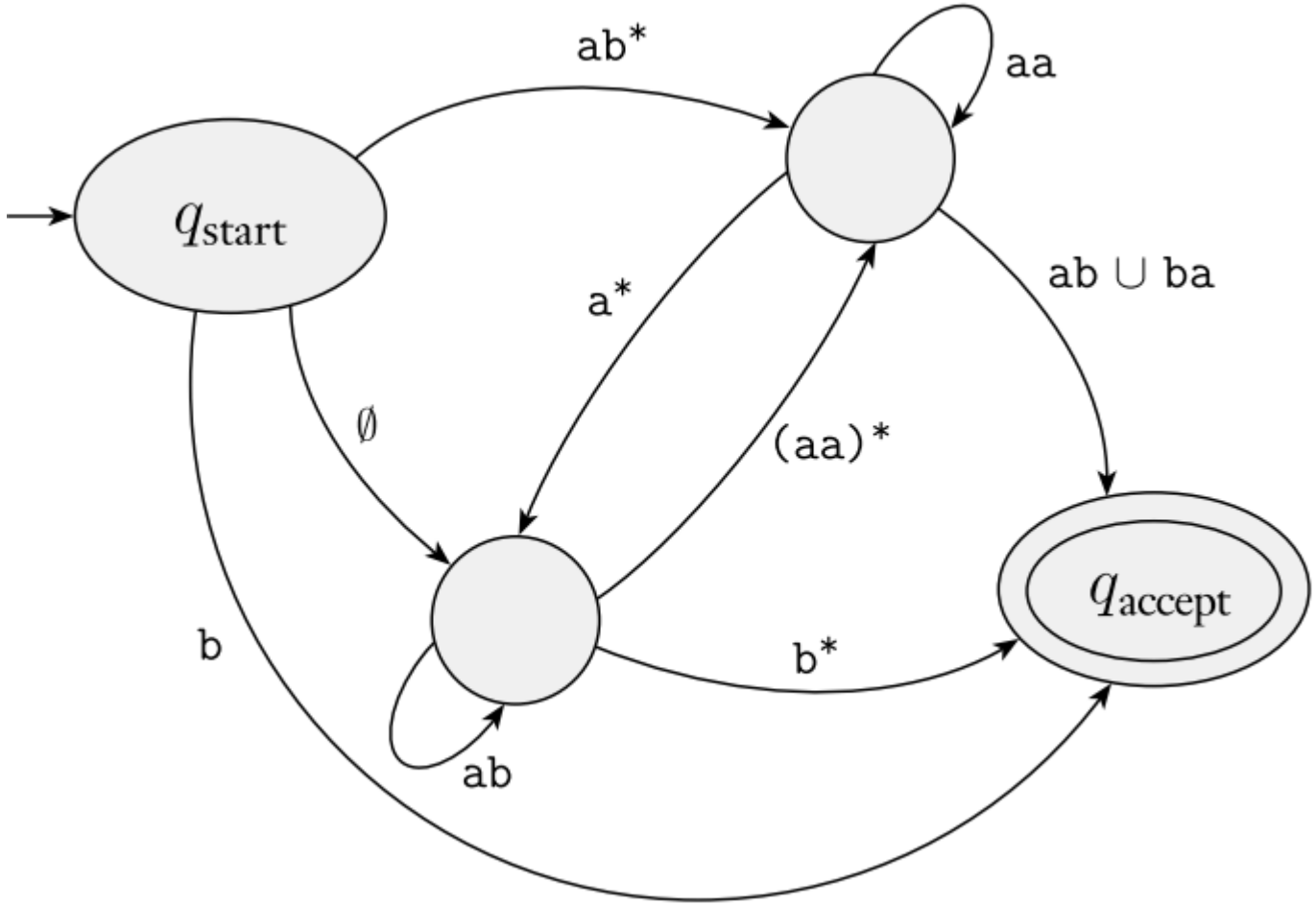
Genelleştirilmiş deterministik olmayan sonlu otomat (GNFA) adı verilen yeni bir sonlu otomat türünü kullanarak bu prosedürü iki parçaya ayırıyoruz. İlk önce DFA'ların GNFA'lara ve ardından GNFA'ların düzenli ifadelere nasıl dönüştürüleceğini gösteriyoruz.

Genelleştirilmiş deterministik olmayan sonlu otomatlar, basitçe deterministik olmayan sonlu otomatlardır; burada geçiş okları, yalnızca alfabenin veya ε'nin üyeleri yerine etiket olarak herhangi bir düzenli ifadeye sahip olabilir. GNFA, sıradan bir NFA'da olduğu gibi her seferinde yalnızca bir sembol olmak zorunda değil, girdiden sembol bloklarını okur. GNFA, girdiden bir sembol bloğu okuyarak iki durumu birbirine bağlayan bir geçiş oku boyunca hareket eder; bu semboller, o ok üzerindeki düzenli ifadeyle tanımlanan bir diziyi oluşturur. Bir GNFA belirleyici değildir ve bu nedenle aynı giriş dizesini işlemek için birkaç farklı yola sahip olabilir. Eğer işlenmesi GNFA'nın girdinin sonunda kabul durumunda olmasına neden olabiliyorsa, girdisini kabul eder. Aşağıdaki şekil bir GNFA örneğini göstermektedir.



Generalized nondeterministic finite automaton

Kolaylık sağlamak amacıyla, GNFA'ların her zaman aşağıdaki koşulları karşılayan özel bir forma sahip olmasını şart koşuyoruz

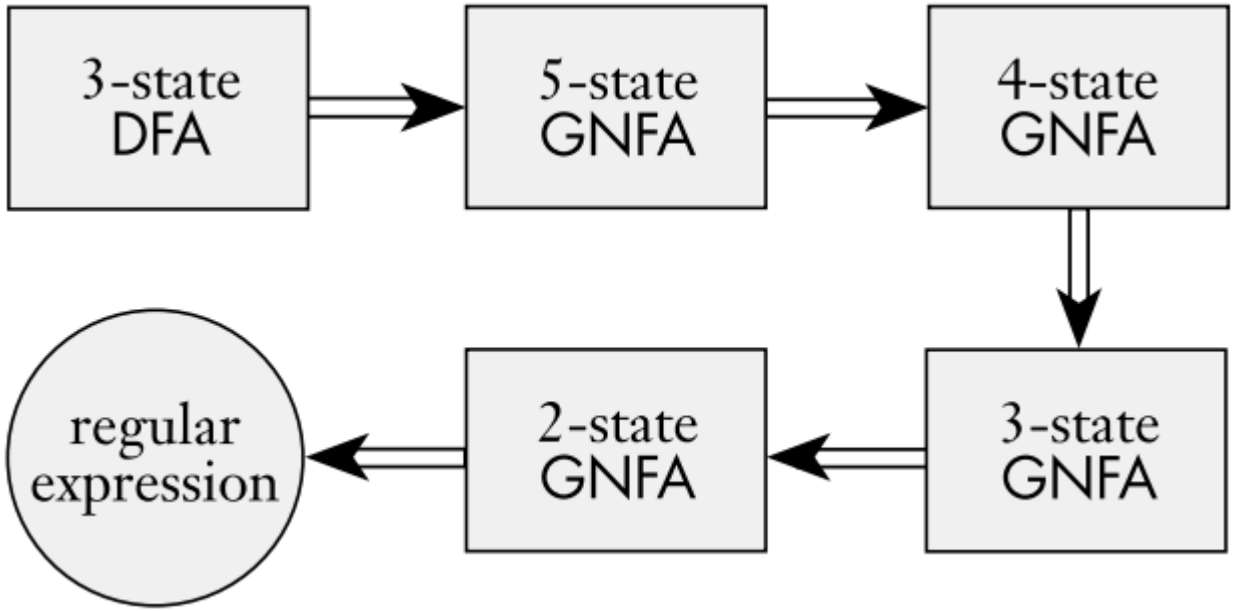
.• Başlangıç durumunda diğer tüm durumlara giden geçiş okları vardır ancak başka herhangi bir durumdan gelen ok yoktur

.• Yalnızca tek bir kabul durumu vardır ve diğer tüm durumlardan oklar gelir, ancak başka herhangi bir duruma giden ok yoktur. Ayrıca kabul durumu başlangıç durumuyla aynı değildir

.• Başlangıç ve kabul durumları dışında, bir ok her durumdan diğer duruma ve her durumdan kendisine gider.

Bir DFA'yı özel formda kolayca GNFA'ya dönüştürebiliriz. Basitçe eski başlangıç durumuna ε oklu yeni bir başlangıç durumu ve eski kabul durumlarından ε oklu yeni bir kabul durumu ekliyoruz. Herhangi bir okun birden fazla etiketi varsa (veya aynı iki durum arasında aynı yönde giden birden fazla ok varsa), her birinin yerini, etiketi önceki etiketlerin birleşimi olan tek bir okla değiştiririz. Son olarak, üzerinde ok bulunmayan durumlar arasına Ø etiketli oklar ekliyoruz. Bu son adım, tanınan dili değiştirmeyecektir çünkü Ø ile etiketlenmiş bir geçiş hiçbir zaman kullanılamaz. Buradan itibaren tüm GNFA'ların özel formda olduğunu varsayıyoruz.

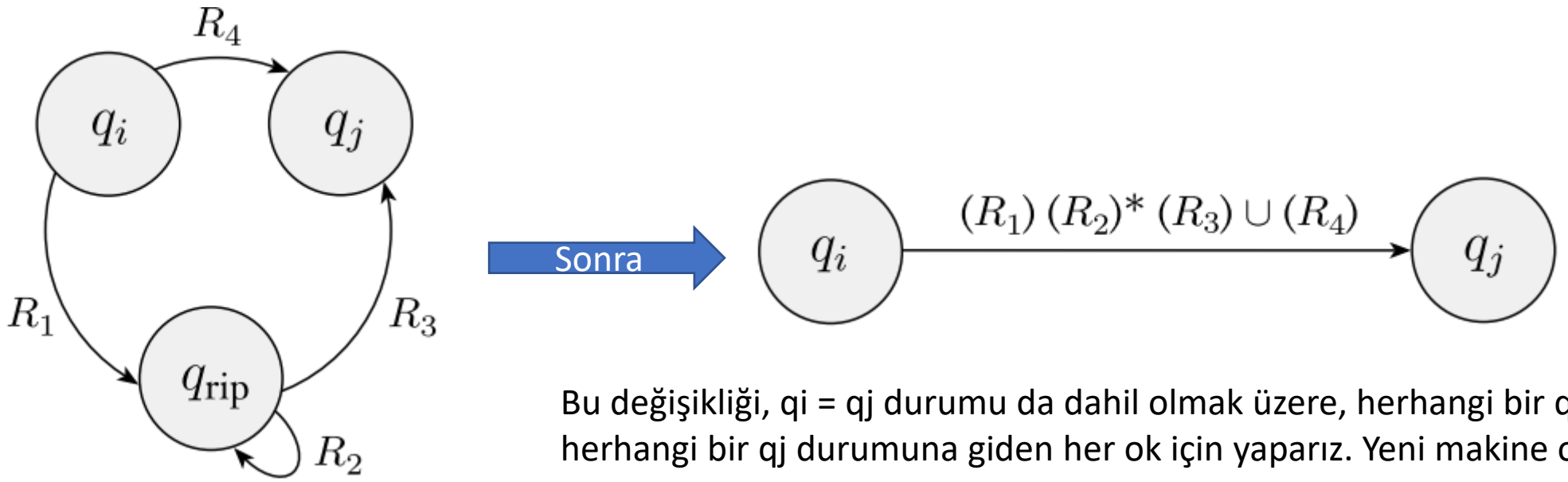
Şimdi bir GNFA'nın normal ifadeye nasıl dönüştürüleceğini gösteriyoruz. GNFA'nın k durumu olduğunu varsayalım. O halde, bir GNFA'nın bir başlangıç ve kabul durumunun olması ve bunların birbirlerinden farklı olması gerektiğinden, $k \geq 2$ olduğunu biliyoruz. Eğer $k > 2$ ise, $k - 1$ durumlu eşdeğer bir GNFA oluştururuz. Bu adım, yeni GNFA'da iki duruma düşene kadar tekrarlanabilir. Eğer $k = 2$ ise, GNFA'nın başlangıç durumundan kabul durumuna giden tek bir oku vardır. Bu okun etiketi eşdeğer düzenli ifadedir. Örneğin üç durumlu bir DFA'nın eşdeğer bir düzenli ifadeye dönüştürülmesinin aşamaları aşağıdaki şekilde gösterilmiştir.



Bir DFA'yı regüler ifadeye dönüştürmenin tipik aşamaları

Önemli adım, k > 2 olduğunda bir eksik duruma sahip eşdeğer bir GNFA oluşturmaktır. Bunu bir durum seçerek, onu makineden sökerek ve geri kalanını onararak aynı dilin hâlâ tanınmasını sağlayarak yaparız. Başlangıç veya kabul etme durumu olmaması koşuluyla herhangi bir durum bunu yapacaktır. k > 2 olduğundan böyle bir durumun var olacağı garantidir. Kaldırılan duruma qrip diyelim.

Qrip'i kaldırdıktan sonra, kalan okların her birini etiketleyen düzenli ifadeleri değiştirerek makineyi onarıyoruz. Yeni etiketler, kayıp hesaplamaları geri ekleyerek qrip eksikliğini telafi ediyor. Qi durumundan qj durumuna giden yeni etiket, makineyi doğrudan veya qrip aracılığıyla qi'den qj'ye götürecek tüm dizeleri tanımlayan düzenli bir ifadedir. Bu yaklaşımı Şekilde gösteriyoruz.



Bu değişikliği, qi = qj durumu da dahil olmak üzere, herhangi bir qi durumundan herhangi bir qj durumuna giden her ok için yaparız. Yeni makine orijinal dili tanır.

Şimdi bu fikri resmi olarak hayata geçirelim. İlk olarak, ispatı kolaylaştırmak için tanıtılan yeni otomat tipini resmi olarak tanımlıyoruz. Bir GNFA, X biçimindeki geçiş fonksiyonu haricinde, deterministik olmayan sonlu bir otomatla benzerdir.

$$\delta: (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \longrightarrow \mathcal{R}.$$

Bir ok her durumu diğer tüm durumlara bağlar, ancak KABUL DURUMUNDAN gelen veya BAŞLANGIÇ DURUMUNA giden hiçbir ok yoktur.

A ***generalized nondeterministic finite automaton*** is a 5-tuple, $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$, where

1. Q is the finite set of states,
2. Σ is the input alphabet,
3. $\delta: (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \longrightarrow \mathcal{R}$ is the transition function,
4. q_{start} is the start state, and
5. q_{accept} is the accept state.

Bir GNFA, eğer w = w1w2 ⋯ wk ise, Σ∗ cinsinden bir w dizisini kabul eder, burada her bir wi Σ∗'dir ve q0, q1,...,qk durumlarının bir dizisi mevcuttur;

1. q0 = qstart başlangıç durumudur,

2. qk = qaccept, kabul etme durumudur ve

3. her i için wi ∈ L(Ri) var, burada Ri = δ(qi−1, qi); diğer bir deyişle Ri, qi−1'den qi'ye giden okun üzerindeki ifadedir.

M makines, A dili için bir DFA olsun. **Yeni bir başlangıç durumu** ve **yeni bir kabul durumu** ve gerektiği şekilde ek geçiş okları ekleyerek M'yi bir G isimli GNFA'ya dönüştürdük. Bir GNFA'yı giriş olarak alan ve eşdeğer bir regüler ifade döndüren CONVERT(G) prosedürü oluşturmak istiyoruz

Bu prosedür özyinelemeyi kullanır, yani kendini çağırır. Prosedür kendisini yalnızca daha az duruma sahip bir GNFA'yı işlemek için çağırdığından sonsuz bir döngüden kaçınılır. GNFA'nın iki durumunun olduğu durum özyineleme olmadan ele alınır.

CONVERT(G):

1. G'nin durum sayısı k olsun.

2. Eğer k = 2 ise, G bir başlangıç durumu, bir kabul durumu ve bunları birbirine bağlayan ve R düzenli ifadesiyle etiketlenmiş tek bir oktan oluşmalıdır. R ifadesini döndürün.

3. Eğer k > 2 ise, qstart ve qaccept'ten farklı herhangi bir qrip ∈ Q durumunu seçeriz. G''nin GNFA olsun (Q′ , Σ, δ′ , qstart, qaccept),

Q′ = Q − {qrip},
Herhangibir qi ∈ Q′ − {qaccept} ve qj ∈ Q′ − {qstart} için

$$\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4)$$

$$R_1 = \delta(q_i, q_{\text{rip}})$$
$$R_2 = \delta(q_{\text{rip}}, q_{\text{rip}})$$
$$R_3 = \delta(q_{\text{rip}}, q_j)$$
$$R_4 = \delta(q_i, q_j)$$

4. CONVERT(G′)'yi hesaplayın ve bu değeri döndürün.

Temel: İddianın k = 2 durum için doğru olduğunu kanıtlayın. Eğer G'nin yalnızca iki durumu varsa, başlangıç durumundan kabul durumuna giden yalnızca tek bir oka sahip olabilir. Bu oktaki düzenli ifade etiketi, G'nin kabul durumuna geçmesine izin veren tüm dizeleri açıklar. Dolayısıyla bu ifade G'ye eşdeğerdir.

Tümevarım adımı: İddianın k - 1 durum için doğru olduğunu varsayalım ve bunu iddianın k durum için doğru olduğunu kanıtlamak için varsayım olarak kullanın. Öncelikle G ve G''nin aynı dili tanıdığını gösteriyoruz. G'nin w girdisini kabul ettiğini varsayalım. Daha sonra hesaplamanın kabul eden bir dalında G bir dizi durum girer:

$$q_{\text{start}}, q_1, q_2, q_3, \ldots, q_{\text{accept}}.$$

Eğer bunlardan hiçbiri kaldırılan durum qrip değilse, açıkça G' de w'yi kabul eder. Bunun nedeni, G' oklarını etiketleyen yeni düzenli ifadelerin her birinin, bir birliğin parçası olarak eski düzenli ifadeyi içermesidir.

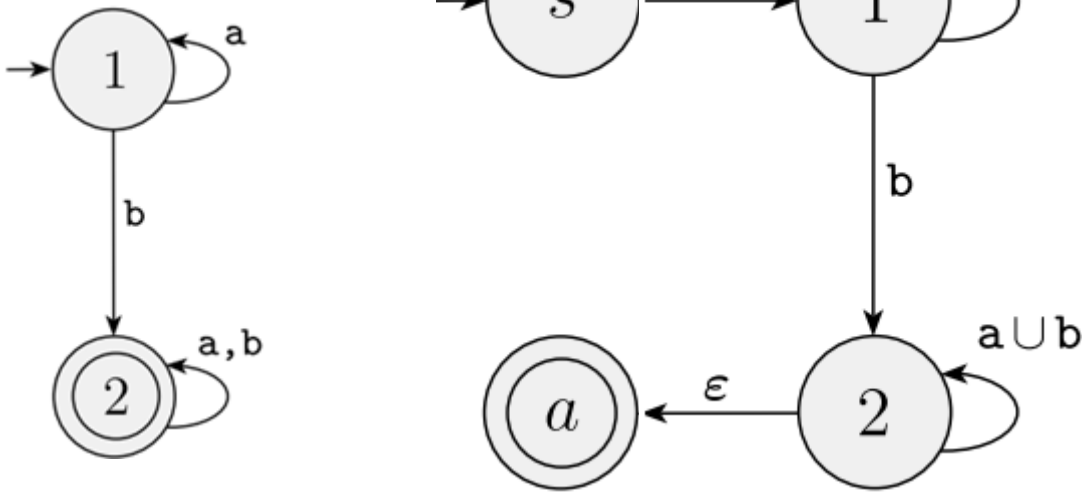
Eğer qrip ortaya çıkarsa, ardışık qrip durumlarının her bir çalışmasının kaldırılması, G' için kabul edilebilir bir hesaplama oluşturur. Bir koşuyu paranteze alan qi ve qj durumları, aralarındaki ok üzerinde, G'deki qrip aracılığıyla qi'yi qj'ye götüren tüm dizileri tanımlayan yeni bir düzenli ifadeye sahiptir. Dolayısıyla G', w'yi kabul eder.

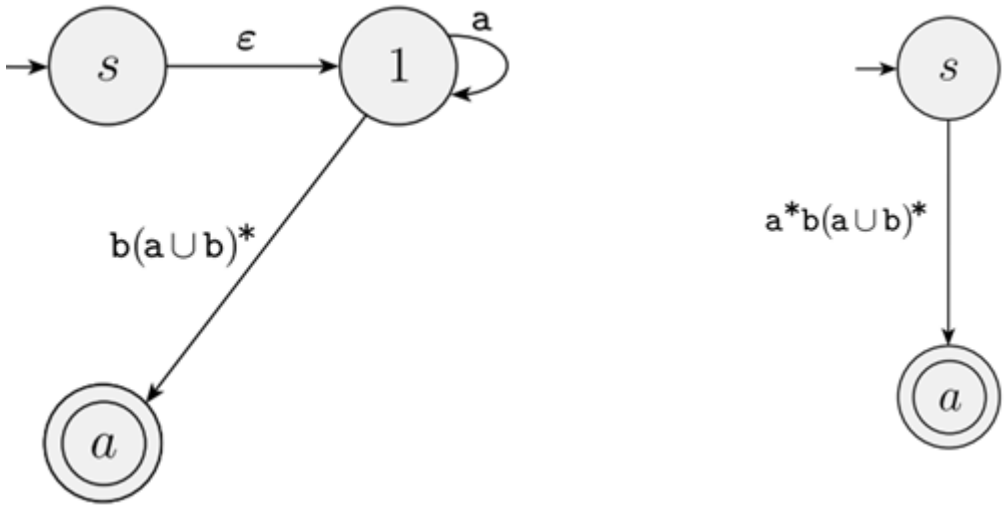
Tersine, G'nin w girdisini kabul ettiğini varsayalım. G'deki qi ve qj durumları arasındaki her ok, qi'yi doğrudan veya qrip yoluyla G'deki qj'ye götüren dizilerin koleksiyonunu tanımladığından, G'nin w'yi de kabul etmesi gerekir. Dolayısıyla G ve G' eşdeğerdir.

Tümevarım hipotezi, algoritma kendisini G' girişinde yinelemeli olarak çağırdığında sonucun G'ye eşdeğer bir düzenli ifade olduğunu, çünkü G'nin k - 1 durumuna sahip olduğunu belirtir. Dolayısıyla bu düzenli ifade de G'ye eşdeğerdir ve algoritmanın doğruluğu kanıtlanmıştır.

qstart ve qaccept yerine s ve a adı verilen yeni bir başlangıç durumu ve yeni bir kabul durumu ekleyerek dört durumlu bir GNFA oluşturuyoruz, böylece bunları uygun şekilde çizebiliyoruz.

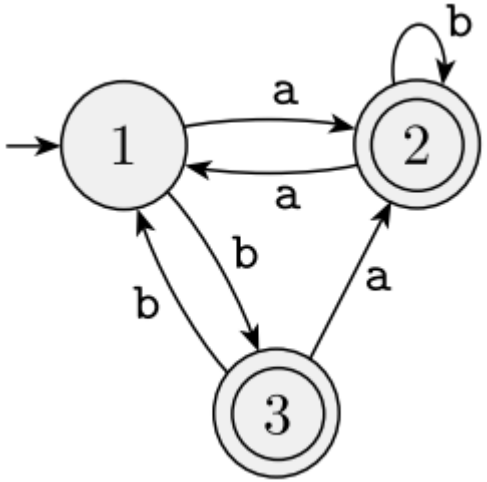
Şeklin karmaşıklığını önlemek için Ø etiketli oklar mevcut olsa bile çizmiyoruz. DFA'nın 2. durumundaki öz döngüdeki a, b etiketini, GNFA'nın karşılık gelen noktasındaki aUb etiketiyle değiştirdiğimizi unutmayın. Bunu yapıyoruz çünkü DFA'nın etiketi biri a için, diğeri b için olmak üzere iki geçişi temsil ederken, GNFA'nın 2'den kendisine giden yalnızca tek bir geçişi olabilir.

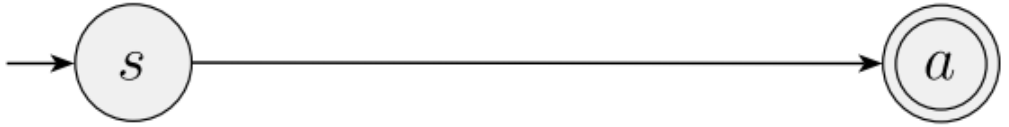




Şekil 1.67(c)'de durum 2'yi kaldırıyoruz ve kalan ok etiketlerini güncelliyoruz. Bu durumda değişen tek etiket 1'den a'ya olan etikettir. (b) şıkkında Ø vardı ama (c) şıkkında b(a ∪ b)∗. Bu sonucu CONVERT prosedürünün 3. adımını takip ederek elde ederiz. qi durumu 1. durumdur, qj durumu a'dır ve qrip 2'dir, dolayısıyla R1 = b, R2 = a ∪ b, R3 = ε ve R4 = Ø. Bu nedenle, 1'den a'ya giden okun üzerindeki yeni etiket (b)(a ∪ b)∗(ε) ∪ Ø'dur. Bu düzenli ifadeyi b(a ∪ b)∗ şeklinde sadeleştiriyoruz. Şekil 1.67(d)'de, (c) kısmından durum 1'i çıkarıyoruz ve aynı prosedürü izliyoruz. Yalnızca başlangıç ve kabul durumları kaldığı için bunları birleştiren ok üzerindeki etiket, orijinal DFA'ya eşdeğer olan normal ifadedir.

Örnek 2: Üçdurumlu bir DFA ile başlıyoruz.





$$(a(aa \cup b)^* ab \cup b)((ba \cup a)(aa \cup b)^* ab \cup bb)^*((ba \cup a)(aa \cup b)^* \cup \varepsilon) \cup a(aa \cup b)^*$$