**T.C. SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ**

logo, simge, sembol, amblem, ticari marka içeren bir resim

Açıklama otomatik olarak oluşturuldu

**TEKNOLOJİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**BIL-305 BİÇİMSEL DİLLER VE OTOMATLAR**

**Doç.Dr.Zeynep GARİP**

**Arş. Gör.Muhammed Yusuf KÜÇÜKKARA**

**Öğrencilerin Adı Soyadı:** Neslihan KARADENİZ/ Kevser Şebnem ÇİLATA/ Zehra DEMİRTAŞ

**Numara**   **:** 22010903063 / B210109016 / 23010903135

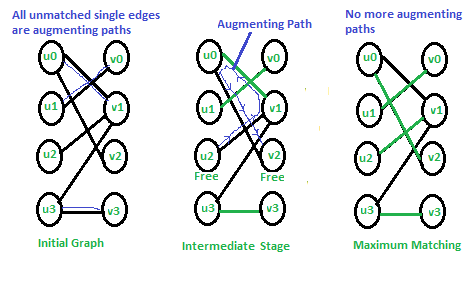
**Proje Konusu :** Dil Karmaşıklığı ve Minimizasyon

**Dil Karmaşıklığı ve Minimizasyon**

Bu algoritma, bir deterministik sonlu otomat (DFA) tanımlayıp onu minimize etmeyi ve minimize edilmiş DFA'nın gramerini, geçişlerini ve yeni durumlarını hem terminalde yazdırmayı hem de bir görsel dosya olarak kaydetmeyi amaçlamaktadır.

Minimizasyon işlemi, DFA'nın işlevselliğini değiştirmeden durum sayısını azaltmayı hedefler. Fazladan veya gereksiz durumlar kaldırılır ve eşdeğer davranış sergileyen durumlar birleştirilir. Bu, DFA'yı daha basit ve optimize hale getirir.

**Hopcroft Algoritması**

Hopcroft algoritması, deterministik sonlu otomatların (DFA) durumlarını minimize etmek için kullanılan etkili algoritmadır. Algoritmanın temel amacı, DFA'nın işlevselliğini değiştirmeden eşdeğer davranış sergileyen durumları birleştirerek gereksiz durumları ortadan kaldırmaktır. Bu işlem, DFA'nın daha az durumda çalışmasını sağlar ve daha verimli bir otomata dönüştürür. Hopcroft algoritması, öncelikle DFA'nın durumlarını iki ana gruba ayırarak başlar: kabul edilen durumlar (accept states) ve kabul edilmeyen durumlar (non-accept states). Bu gruplar, algoritmanın başlangıç partisyonlarını oluşturur.

Algoritma, her bir grubu iteratif olarak kontrol eder ve durumların giriş sembollerine göre diğer gruplara olan geçişlerini analiz eder. Eğer bir gruptaki durumlar aynı giriş sembolüyle farklı gruplara geçiyorsa, bu grup iki alt gruba ayrılır. Bu işlem, grupların daha fazla bölünemediği, yani tüm durumların kendi içinde tamamen eşdeğer olduğu noktaya kadar devam eder. Bölme işlemi sırasında her giriş sembolü ve her grup dikkate alınır. Bölme tamamlandıktan sonra, elde edilen son gruplar minimize edilmiş DFA'nın yeni durumlarını temsil eder.

Hopcroft algoritmasının son aşamasında, yeni gruplara birer isim atanır ve eski geçiş fonksiyonları, bu yeni durumlara göre yeniden düzenlenir. Başlangıç durumu ve kabul durumları da bu yeni gruplar içinde yeniden tanımlanır. Sonuç olarak, minimize edilmiş DFA, orijinal DFA ile aynı davranışı sergilerken çok daha az sayıda duruma sahiptir. Hopcroft algoritması, durum sayısı ve giriş sembollerine bağlı olarak O(nlogn) zaman karmaşıklığına sahip olup, DFA minimizasyonunda hızlı ve güvenilir bir yöntem olarak kabul edilir.

**Kullanılan Kütüphaneler**

**yazı tipi, metin, ekran görüntüsü, grafik içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Bu programda kullanılan “graphviz” kütüphanesi, DFA'nın durumlarını ve geçişlerini görselleştirmek için kullanılmıştır. Program, “graphviz”in Digraph sınıfı ile DFA’nın başlangıç durumu, kabul durumları ve geçişlerini grafiksel olarak temsil eder ve .png formatında dosya oluşturur. Bu sayede DFA’nın yapısı kullanıcıya daha anlaşılır ve analiz edilebilir bir şekilde sunulur.

“collections.defaultdict” ise, DFA’nın geçiş fonksiyonunu tersine çevirmek ve durumlar arasındaki ilişkileri düzenlemek için kullanılmıştır. Varsayılan bir değer türü belirlediği için anahtar hatalarını önler ve minimizasyon algoritmasında, geçişlerin doğru bir şekilde takip edilmesini sağlar.

Bu iki kütüphane, programın görsel çıktılar üretme ve geçiş fonksiyonlarını yönetme gibi temel ihtiyaçlarını karşılar.

**Programın Adımları**

1. *Kullanıcıdan DFA Bileşenlerini Alma:*
   * Durumlar (states): DFA'nın tüm durumları kullanıcıdan alınır.
   * Alfabe (alphabet): DFA'nın çalıştığı giriş semboller kümesi tanımlanır.
   * Başlangıç Durumu (start\_state): DFA'nın başlatıldığı durum belirtilir.
   * Kabul Durumları (accept\_states): DFA'nın giriş dizilerini kabul ettiği durumlar belirtilir.
   * Geçiş Fonksiyonları (transition\_function): DFA'nın her durumu ve giriş sembolü için hedef durumları tanımlayan fonksiyon alınır.
2. *Orijinal DFA'nın Görselleştirilmesi:* Kullanıcının verdiği bilgilerle tanımlanan DFA'nın görsel bir temsili oluşturulur.
3. *DFA Minimizasyonu:*
   * DFA'nın minimize edilmesi için:
     + Kabul edilen ve edilmeyen durumlar gruplara ayrılır.
     + Durumların giriş sembollerine göre eşdeğerlik kontrolü yapılır.
     + Fazladan durumlar kaldırılır ve eşdeğer durumlar birleştirilir.
   * Yeni DFA tanımlanır ve hem terminalde hem de görsel olarak gösterilir.
4. *Minimize Edilmiş DFA'nın Görselleştirilmesi:* Minimize edilmiş DFA, optimize edilmiş yeni durumlarla birlikte (minimized\_dfa.png) kaydedilir.
5. *Gramer ve Yeni Durumların Yazdırılması:* Minimize edilmiş DFA'nın başlangıç durumu, kabul durumları ve geçiş fonksiyonları terminale yazdırılır. Yeni durumların, orijinal DFA'nın hangi durumlarını içerdiği detaylı olarak listelenir.

**DFA Minimizasyon Algoritmasının Adımları**

Minimizasyon algoritması, DFA'nın gereksiz veya eşdeğer davranış sergileyen durumlarını kaldırmayı hedefler. Bu işlem şu adımlarla gerçekleştirilir:

**1. Durumların Gruplandırılması:** İlk olarak DFA'nın durumları, kabul edilen (accept states) ve kabul edilmeyen (non-accept states) durumlar olarak iki gruba ayrılır. Bu gruplar başlangıç partisyonlarını oluşturur.

Örnek:

* + Kabul edilen durumlar: {q2, q4}
  + Kabul edilmeyen durumlar: {q0, q1, q3}

Bu gruplar, durumların eşdeğer olup olmadığını belirlemede başlangıç noktasıdır.

**2. Geçişlere Göre Ayırma (Refinement):** Her grubun durumları, giriş sembollerine (alphabet) göre kontrol edilir. Eğer aynı gruptaki iki durum farklı sembollerde farklı gruplara geçiş yapıyorsa, bu durumlar ayrılır ve yeni gruplar oluşturulur.

Örnek:

* + {q0, q1} grubundaki q0 sembol 0 ile {q2} grubuna, q1 ise {q3} grubuna geçiyor.
  + Bu durumda {q0, q1} grubu ikiye ayrılır:
    - {q0}
    - {q1}

Bu işlem tüm gruplar ve semboller için tekrarlanır.

**3. Durum Gruplarının İteratif Bölünmesi:** Durumlar tamamen ayrışıp her grubun içinde yalnızca eşdeğer durumlar kalana kadar işlem devam eder. Bu işlem iteratif olarak yapılır:

* Her grup, giriş sembollerine göre kontrol edilir.
* Eğer farklı geçişlere sahip durumlar varsa grup ikiye bölünür.
* Bölme işlemi daha fazla bölünme mümkün olmayana kadar devam eder.

**4. Yeni Durumların Oluşturulması:** Elde edilen son partisyonlar, minimize edilmiş DFA'nın yeni durumlarını temsil eder. Her gruba bir isim atanır (örneğin, s0, s1, s2).

Örnek:

* + {q0, q1} → s0
  + {q2} → s1
  + {q3, q4} → s2

**5. Yeni Geçiş Fonksiyonlarının Oluşturulması:** Yeni durumlar oluşturulduktan sonra, eski geçiş fonksiyonları kullanılarak yeni geçişler tanımlanır:

* Her eski durum, içinde bulunduğu gruba karşılık gelen yeni durumla değiştirilir.
* Giriş sembollerine göre eski geçişler, yeni durumlara uygun şekilde düzenlenir.

Örnek:

* + Eski geçiş: q0 --0--> q2
  + Yeni geçiş: s0 --0--> s1

**6. Yeni Başlangıç ve Kabul Durumlarının Belirlenmesi**

* Başlangıç durumu (start\_state), içinde bulunduğu yeni gruba atanır.
  + Örneğin, q0 başlangıç durumu s0 grubuna aitse, yeni başlangıç durumu s0 olur.
* Kabul durumları (accept\_states), içinde bulundukları yeni gruplarla değiştirilir.

Örnek:

* + Eski kabul durumları: {q2, q4}
  + Yeni kabul durumları: {s1, s2}

**7. Sonuçların Görselleştirilmesi ve Yazdırılması:** Yeni durumlar, geçiş fonksiyonları ve gramer terminalde yazdırılır.Minimize edilmiş DFA'nın görseli oluşturularak “.png” dosyasına kaydedilir.

Bu program, DFA minimizasyonu için pratik bir çözüm sunar. Program, giriş olarak DFA'nın durumlarını, alfabe sembollerini, başlangıç ve kabul durumlarını, geçiş fonksiyonlarını alır. Çıktı olarak, minimize edilmiş DFA'nın görsel temsili, grameri ve yeni durumların bilgisi sağlanır.

Minimizasyon işlemi, DFA'nın durumlarını azaltarak daha verimli bir sonlu otomat oluşturmayı sağlar. Bu, DFA'nın daha az bellek kullanması ve daha hızlı çalışması için önemlidir. Program hem algoritmanın işleyişini hem de görsel çıktıları ile kullanıcıya sonuçları net bir şekilde sunar.

**Program çıktıları**

|  |  |  |
| --- | --- | --- |
| **metin, ekran görüntüsü, tasarım içeren bir resim  Açıklama otomatik olarak oluşturuldu** |  |  |

|  |  |
| --- | --- |
| DFA Çizimi | Minimize edilmiş DFA |
| taslak, çizim, diyagram, beyaz içeren bir resim  Açıklama otomatik olarak oluşturuldu | taslak, çizim, daire, tasarım içeren bir resim  Açıklama otomatik olarak oluşturuldu |