INSTITUTO TECNOLOGICO DE MEXICO

CAMPUS PACHUCA

Lenguajes y Automatas

3.2 Caso Practico

Baume Lazcano Rodolfo

Gabriela Dyvheke Espinosa Antelis

1 Mayo 2024

Aplicación: Sistema de Autenticación de Usuario basado en Autómata Finito

Descripción del Problema:

El objetivo es diseñar un sistema de autenticación para una aplicación que verifique las contraseñas de los usuarios asegurándose de que cumplan con criterios específicos de seguridad antes de permitir el acceso. Estos criterios incluyen longitud mínima, uso de caracteres especiales, y combinación de mayúsculas y minúsculas.

Estados Especificos:

Inicio: Estado inicial, donde no se ha leído ningún carácter.

Mayúscula: Estado que indica que se ha leído al menos una letra mayúscula.

Minúscula: Estado que indica que se ha leído al menos una letra minúscula.

Número: Estado que indica que se ha leído al menos un número.

Carácter Especial: Estado que indica que se ha leído al menos un carácter especial.

Combinado: Estado que refleja la combinación de los anteriores, indicando una combinación válida.

Válido: Estado final que indica que la contraseña cumple con todos los requisitos.

Inválido: Estado que se alcanza si la contraseña no cumple con alguno de los requisitos.

Transiciones Especificas:

Desde Inicio, dependiendo del tipo de carácter leído, se puede mover a cualquier otro estado excepto Válido.

Transiciones entre Mayúscula, Minúscula, Número, y Carácter Especial dependiendo de la entrada de caracteres subsiguientes.

Una vez alcanzados los requisitos (por ejemplo, longitud mínima y todos los tipos de caracteres requeridos), se transita al estado Válido.

Si en algún momento se detecta una violación de las reglas (por ejemplo, una secuencia de caracteres inadecuada o longitud insuficiente), se transita a Inválido.

Acciones:

Validación: En cada transición, se valida el tipo de carácter y se actualiza el estado.

Finalización: Al alcanzar el estado Válido, se permite el acceso al usuario. Si se llega al estado Inválido, se niega el acceso y se solicita una nueva contraseña.

Para profundizar en el sistema de autenticación de usuario utilizando un Autómata Finito Determinista (DFA) para la validación de contraseñas, debemos considerar varios aspectos importantes que contribuyen a su efectividad y eficiencia. Vamos a expandir la descripción inicial, detallando más sobre el diseño, implementación, y consideraciones adicionales que impactan en la seguridad y usabilidad del sistema.

Ampliación de la Implementación del Autómata Finito

Diseño del DFA

El DFA diseñado para la validación de contraseñas debe ser capaz de identificar múltiples patrones y cumplir con los requisitos de seguridad específicos. Aquí detallamos el diseño y las transiciones del autómata:

Estados y Transiciones:

Estado Inicial (Inicio): No se ha procesado ningún carácter. Dependiendo del carácter leído, el autómata se mueve a uno de los estados específicos (mayúscula, minúscula, número, carácter especial).

Estados Intermedios (Mayúscula, Minúscula, Número, Carácter Especial): Cada uno representa haber encontrado al menos un carácter de su tipo respectivo. El autómata puede permanecer en un estado o moverse a otro estado intermedio si se lee un carácter de un nuevo tipo.

Estado de Combinación (Combinado): Se alcanza este estado cuando se han encontrado todos los tipos necesarios de caracteres. Este estado verifica que se sigan cumpliendo las condiciones mientras se procesan caracteres adicionales hasta alcanzar la longitud mínima requerida.

Estado Final (Válido): Se alcanza solo cuando la contraseña tiene la longitud adecuada y contiene todos los tipos de caracteres necesarios.

Estado de Error (Inválido): Este estado se alcanza si en cualquier momento la entrada no cumple con los requisitos de longitud o composición.

Reglas de Transición:

Transiciones basadas en el tipo de carácter de entrada.

Transiciones que consideran la longitud de la contraseña, agregando una comprobación después de cada entrada para determinar si se ha alcanzado la longitud mínima.

Implementación Técnica

La implementación de este DFA en un sistema real implica escribir un algoritmo que procese cada carácter de la contraseña y cambie los estados según las reglas definidas.

Código Mejorado: El siguiente código Python muestra un ejemplo de la implementacien el DFA incluyendo sus estados y transiciones

```
class PasswordValidator:
    def __init__(self, min_length=8):
        self.min_length = min_length
        self.reset()

    def reset(self):
        self.has_upper = False
        self.has_lower = False
        self.has_number = False
        self.has_special = False
        self.length = 0

    def process_char(self, char):
        if char.isupper():
            self.has_upper = True
        elif char.islower():
            self.has_lower = True
        elif char.isdigit():
            self.has_number = True
        elif not char.isalnum():
            self.has_special = True
        self.length += 1

    def is_valid(self):
        if (self.length >= self.min_length and self.has_upper and self.has_lower
and
            self.has_number and self.has_special):
```

```
            return True
        return False

    def validate_password(self, password):
        self.reset()
        for char in password:
            self.process_char(char)
        return self.is_valid()

# Usage
validator = PasswordValidator()
password = "Example@123"
if validator.validate_password(password):
    print("Password is valid.")
else:
    print("Password is invalid.")
```

Consideraciones de Seguridad y Usabilidad

Seguridad: Asegurarse de que el DFA no es vulnerable a ataques de tiempo, donde el tiempo que toma validar una contraseña podría proporcionar pistas sobre su validez.

Usabilidad: Proporcionar mensajes de error claros que ayuden al usuario a entender por qué su contraseña es inválida puede mejorar la experiencia del usuario y ayudar a educar sobre buenas prácticas de seguridad.

Mantenimiento y Escalabilidad: El diseño debe permitir ajustes fáciles en los parámetros del autómata, como cambiar la longitud mínima de la contraseña o los tipos de caracteres requeridos, sin necesidad de una reescritura completa.

Conclusión

Este enfoque detallado del diseño e implementación de un DFA para la validación de contraseñas ilustra cómo los principios de teoría de autómatas se pueden aplicar en sistemas reales para resolver problemas complejos de forma eficiente. Al enfocarse tanto en la precisión del diseño como en la seguridad y la usabilidad, este sistema puede proporcionar una solución robusta para la gestión de autenticación de usuarios en cualquier aplicación moderna.