Universidade do Estado do Rio Grande do Norte - UERN
Departamento de Ciência da Computação
Curso de Ciência da Computação

JOSÉ SEVERINO F. BARBOZA

CONSTRUÇÃO DE UM SBC FUZZY PARA O EXERCÍCIO 2

Natal - RN

2024

Sistema de Autoatendimento com Lógica Fuzzy

Estrutura do Sistema

O sistema de autoatendimento é composto pelos seguintes componentes:

1. **Scanner de Código de Barras**:
    - Dispositivo utilizado para escanear o código de barras de cada produto.
    - Envia as informações do produto para o banco de dados.
2. **Balança**:
    - Dispositivo para medir o peso dos produtos após o escaneamento.
    - Compara o peso medido com o peso registrado no banco de dados.
3. **Banco de Dados**:
    - Armazena informações detalhadas dos produtos, incluindo o peso.
    - Fornece dados necessários para a verificação do peso.
4. **Interface do Usuário**:
    - Tela onde o usuário pode visualizar os produtos escaneados e outras informações.
    - Permite ao usuário interagir com o sistema durante o processo de checkout.
5. **Sistema de Alerta**:
    - Gera alertas quando há discrepâncias entre o peso medido e o peso registrado.
    - Impede que o usuário continue escaneando até que a discrepância seja resolvida.
6. **Módulo de Pagamento**:
    - Processa o pagamento dos produtos escaneados.
    - Pode incluir várias formas de pagamento, como cartão de crédito/débito, pagamento móvel, etc.
7. **Verificação de Peso Total**:
    - Verifica se o peso total dos produtos empacotados corresponde ao peso total registrado.
    - Assegura que todos os produtos foram escaneados corretamente.

Sistema Baseado em Conhecimento (SBC) com Lógica Fuzzy

Objetivo do SBC

O SBC deve garantir que os produtos escaneados e pagos correspondam aos produtos comprados, utilizando verificações de peso e medidas de segurança, e lidar com incertezas nos pesos dos produtos.

Componentes do SBC

1. **Base de Conhecimento**:
    - Armazena regras fuzzy e fatos sobre os produtos e seus pesos.
    - Inclui informações sobre tolerâncias de peso para cada produto.
2. **Motor de Inferência Fuzzy**:

    - Aplica as regras fuzzy da base de conhecimento para verificar a correspondência dos produtos.
    - Gera conclusões e ações baseadas nas informações fornecidas pelo scanner e pela balança.
3. **Interface de Comunicação**:
    - Permite a comunicação entre o SBC e os componentes do sistema de autoatendimento.
    - Envia comandos e recebe dados do scanner, balança, banco de dados, etc.

Construção do SBC com Lógica Fuzzy

Definição das Regras Fuzzy

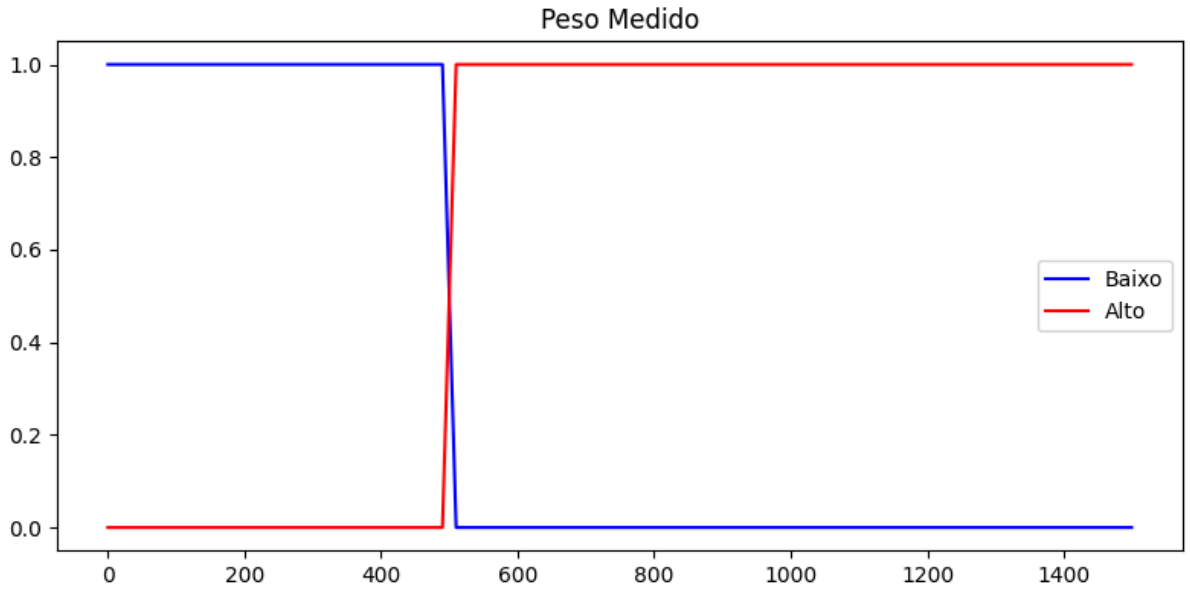
1. **Regra para Produtos Individuais**:
    - **Regra 1**:
        - **Se** o peso medido do produto está dentro da tolerância do peso registrado, **então** o produto é considerado válido.
        - Exemplo: **Se** (peso medido >= peso registrado - tolerância) e (peso medido <= peso registrado + tolerância), **então** produto é válido.
    - **Regra 2**:
        - **Se** o peso medido do produto está fora da tolerância do peso registrado, **então** o produto é considerado inválido.
        - Exemplo: **Se** (peso medido < peso registrado - tolerância) ou (peso medido > peso registrado + tolerância), **então** produto é inválido.
2. **Regras para Peso Total dos Produtos**:
    - **Regra 3**:
        - **Se** o peso total medido dos produtos está dentro da tolerância do peso total registrado, **então** não há discrepância.
        - Exemplo: **Se** (peso total medido >= peso total registrado - tolerância total) e (peso total medido <= peso total registrado + tolerância total), **então** não há discrepância.
    - **Regra 4**:
        - **Se** o peso total medido dos produtos está fora da tolerância do peso total registrado, **então** há discrepância.
        - Exemplo: **Se** (peso total medido < peso total registrado - tolerância total) ou (peso total medido > peso total registrado + tolerância total), **então** há discrepância.

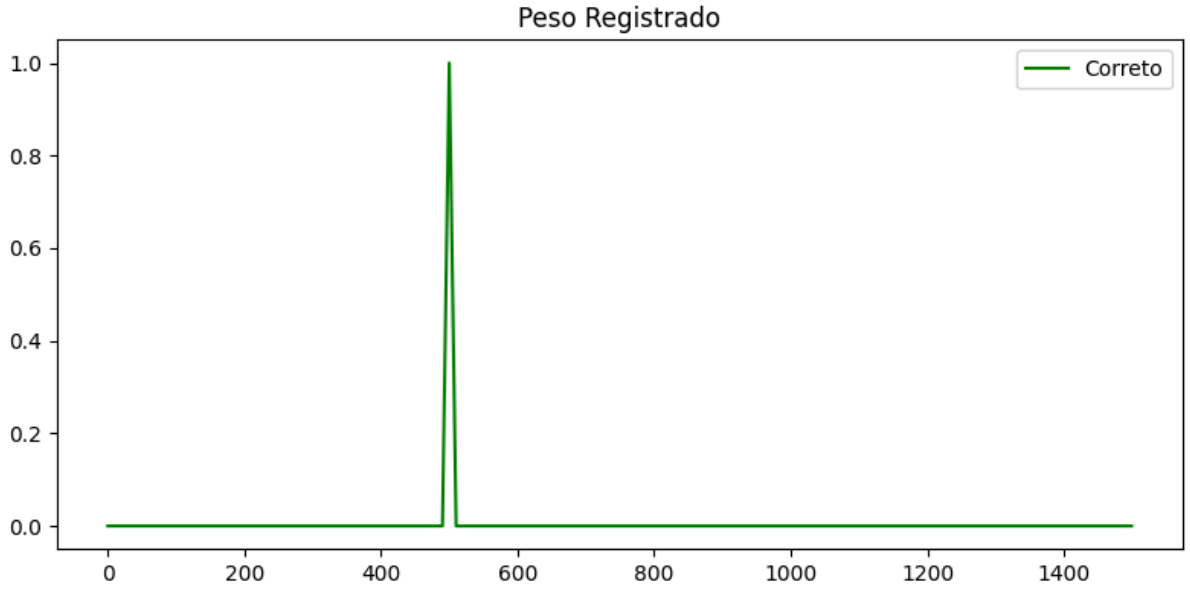
Implementação em Python

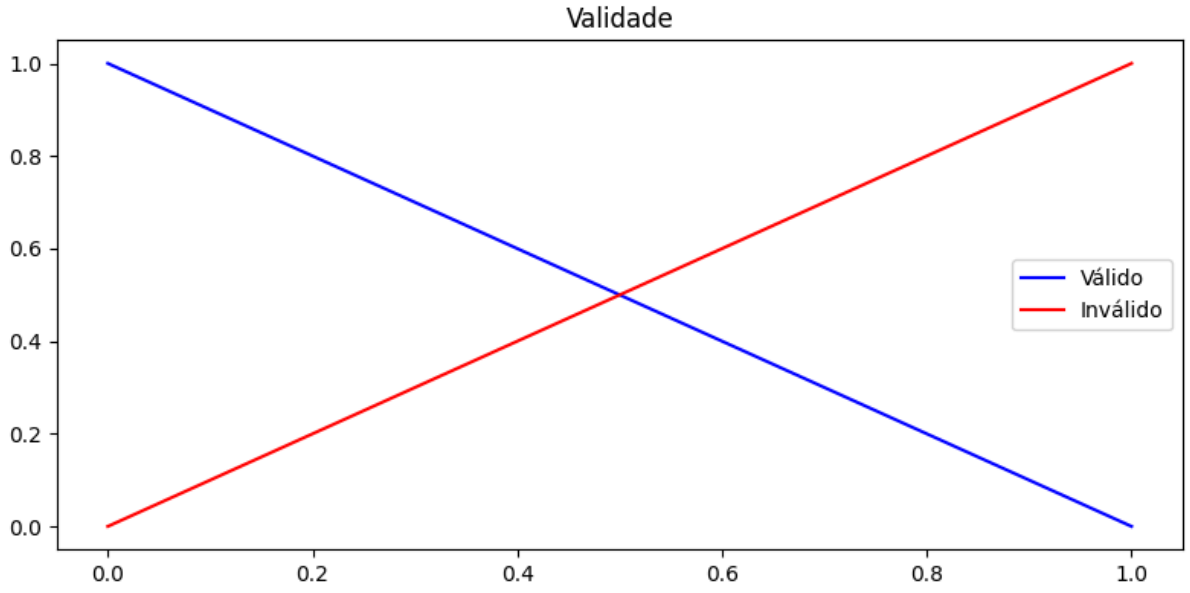
Código Completo com Funções de Pertinência, Regras Fuzzy, Simulação e Gráficos

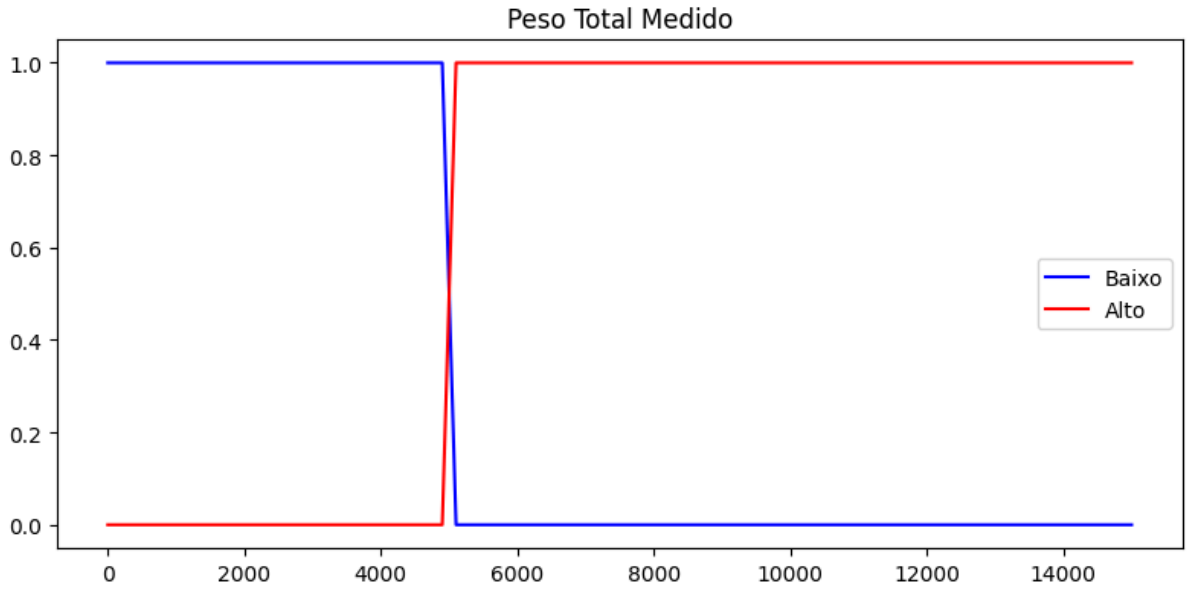
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Definindo as variáveis fuzzy
peso_medido = ctrl.Antecedent(np.arange(0, 1500, 1), 'peso_medido')
peso_registrado = ctrl.Antecedent(np.arange(0, 1500, 1), 'peso_registrado')
validade = ctrl.Consequent(np.arange(0, 1.01, 0.01), 'validade')

peso_total_medido = ctrl.Antecedent(np.arange(0, 15000, 1), 'peso_total_medido')
peso_total_registrado = ctrl.Antecedent(np.arange(0, 15000, 1), 'peso_total_registrado')
discrepancia = ctrl.Consequent(np.arange(0, 1.01, 0.01), 'discrepancia')

# Funções de pertinência para peso medido
peso_medido['baixo'] = fuzz.trapmf(peso_medido.universe, [0, 0, 490, 510])
peso_medido['alto'] = fuzz.trapmf(peso_medido.universe, [490, 510, 1500, 1500])

# Função de pertinência para peso registrado
peso_registrado['correto'] = fuzz.trimf(peso_registrado.universe, [490, 500, 510])

# Funções de pertinência para validade
validade['válido'] = fuzz.trimf(validade.universe, [0, 0, 1])
validade['inválido'] = fuzz.trimf(validade.universe, [0, 1, 1])

# Funções de pertinência para peso total medido
peso_total_medido['baixo'] = fuzz.trapmf(peso_total_medido.universe, [0, 0, 4900, 5100])
peso_total_medido['alto'] = fuzz.trapmf(peso_total_medido.universe, [4900, 5100, 15000,
15000])

# Função de pertinência para peso total registrado
peso_total_registrado['correto'] = fuzz.trimf(peso_total_registrado.universe, [4900, 5000,
5100])

# Funções de pertinência para discrepância
discrepancia['sem_discrepancia'] = fuzz.trimf(discrepancia.universe, [0, 0, 1])
discrepancia['com_discrepancia'] = fuzz.trimf(discrepancia.universe, [0, 1, 1])

# Plotando as funções de pertinência
fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, figsize=(8, 12))

ax0.plot(peso_medido.universe, peso_medido['baixo'].mf, 'b', linewidth=1.5, label='Baixo')
ax0.plot(peso_medido.universe, peso_medido['alto'].mf, 'r', linewidth=1.5, label='Alto')
```

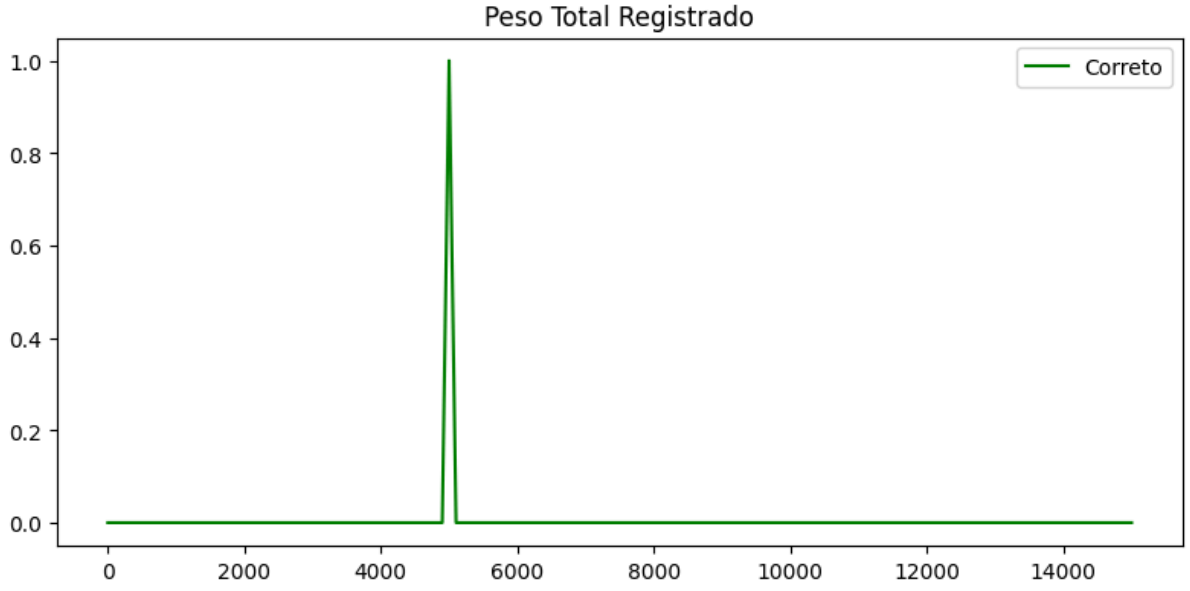
```
ax0.set_title('Peso Medido')
ax0.legend()

ax1.plot(peso_registrado.universe, peso_registrado['correto'].mf, 'g', linewidth=1.5,
label='Correto')
ax1.set_title('Peso Registrado')
ax1.legend()

ax2.plot(validade.universe, validade['válido'].mf, 'b', linewidth=1.5, label='Válido')
ax2.plot(validade.universe, validade['inválido'].mf, 'r', linewidth=1.5, label='Inválido')
ax2.set_title('Validade')
ax2.legend()

plt.tight_layout()
plt.show()

fig, (ax3, ax4, ax5) = plt.subplots(nrows=3, figsize=(8, 12))

ax3.plot(peso_total_medido.universe, peso_total_medido['baixo'].mf, 'b', linewidth=1.5,
label='Baixo')
ax3.plot(peso_total_medido.universe, peso_total_medido['alto'].mf, 'r', linewidth=1.5,
label='Alto')
ax3.set_title('Peso Total Medido')
ax3.legend()

ax4.plot(peso_total_registrado.universe, peso_total_registrado['correto'].mf, 'g',
linewidth=1.5, label='Correto')
ax4.set_title('Peso Total Registrado')
ax4.legend()

ax5.plot(discrepancia.universe, discrepancia['sem_discrepancia'].mf, 'b', linewidth=1.5,
label='Sem Discrepância')
ax5.plot(discrepancia.universe, discrepancia['com_discrepancia'].mf, 'r', linewidth=1.5,
label='Com Discrepância')
ax5.set_title('Discrepância')
ax5.legend()

plt.tight_layout()
plt.show()
```

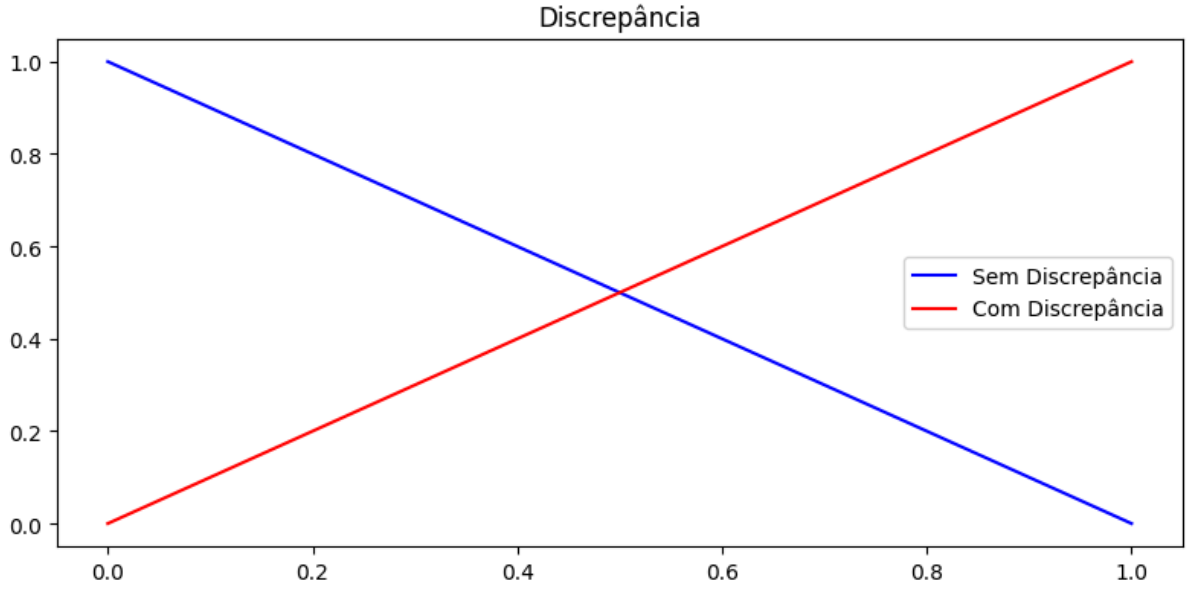

Peso Medido
Baixo
Alto
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
1200
1400


Peso Registrado
Correto
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
1200
1400


Validade
Válido
Inválido
0.0
0.2
0.4
0.6
0.8
1.0


Peso Total Medido
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000
12000
14000
Baixo
Alto


Peso Total Registrado
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000
12000
14000
Correto


Discrepância
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Sem Discrepância
Com Discrepância

1. **Definição das Variáveis Fuzzy**:
    - Variáveis fuzzy são definidas para `peso_medido`, `peso_registrado`, `validade`, `peso_total_medido`, `peso_total_registrado`, e `discrepancia`.
2. **Funções de Pertinência**:
    - As funções de pertinência são criadas usando formas trapezoidais e triangulares para representar diferentes conjuntos fuzzy.
3. **Plotagem das Funções de Pertinência**:
    - Os gráficos são criados para visualizar as funções de pertinência de cada variável fuzzy.
4. **Definição das Regras Fuzzy**:
    - Regras fuzzy são definidas para determinar a validade do produto e a discrepância no peso total dos produtos.
5. **Simulação e Defuzzificação**:
    - A simulação é executada para entradas específicas e os resultados fuzzy são defuzzificados para obter valores precisos.
    - Os resultados da defuzzificação são plotados para visualização.

Conclusão

Este documento fornece uma visão completa do processo de implementação de um sistema de autoatendimento com lógica fuzzy. Inclui a definição das funções de pertinência, a criação de regras fuzzy, a simulação e a defuzzificação dos resultados. Os gráficos ajudam a visualizar como os valores fuzzy são transformados em decisões precisas, garantindo a correspondência correta entre os produtos escaneados e pagos.