Universidade Federal do Piauí - UFPI
Disciplina: Inteligência Artificial
Prof.: Vinícius Ponte Machado
Ciência da Computação - CCN - DC

Trabalho Prático - Sistema Multiagente
Calculadora

Componentes:
Isabela Martins Melo
Matheus Luis Webber
Lucas Willian Gomes do Amaral
Pedro Henryque Nery de Oliveira

2022

1. Introdução

Neste trabalho prático vamos implementar um sistema multiagente, utilizando o framework SPADE, que realiza operações matemáticas. Para cada uma das operações (adição, subtração, multiplicação, divisão, exponenciação e radiciação) existirá um agente, que funcionará de maneira independente, e que serão acionados a partir do agente coordenador.

2. Objetivo

O objetivo deste trabalho é fazer com que os agentes do sistema se comuniquem entre si para resolver as equações matemáticas respeitando a ordem de precedência dos operadores - operações entre parêntesis em primeiro, em segundo potenciação e radiciação, multiplicação e divisão depois e as demais operações por último - percorrendo da esquerda para a direita.

3. Desenvolvimento

Para o desenvolvimento do sistema foi usada a linguagem python na versão 3.6, o framework para construção de sistemas multiagente SPADE e o servidor XMPP openfire server.

O sistema é composto por 8 agentes principais, 6 deles são os agentes de operações de soma, subtração, multiplicação, divisão, potência e raiz quadrada, 1 outro é o agente de parênteses, que fica encarregado de resolver as expressões que ficam dentro de parênteses, e o último é o agente coordenador, responsável por receber a expressão a ser calculada, tratar essa expressão de modo a torná-la adequada para a resolução e selecionar a ordem das operações, de acordo com a prioridade, a serem enviadas para os agentes de operação.

3.1 Framework

Foi utilizado o framework python SPADE, que implementa várias funções essenciais para a construção de um sistema multiagente, os diferentes comportamentos dos agentes, a própria classe de agente utilizada para criar os agentes, bem como uma estrutura para lidar com a construção e envio de mensagens. A função que cria a mensagem (Message()) só permite que sejam trocadas cadeias de caracteres entre os agentes, além de permitir que sejam adicionados os campos de performativa, ontologia, etc, previstos pela FIPA, apesar de esses campos não terem sido usados neste trabalho.

3.2 Comunicação entre agentes

A comunicação entre os agentes do sistema foi implementada usando um servidor XMPP, o XMPP é um protocolo de mensagens instantâneas que permite que os agentes possam se comunicar entre si utilizando sequências de caracteres (strings).

Para que seja possível a comunicação, cada agente precisa ter uma conta com nome do agente e senha cadastrada no servidor XMPP, esse cadastro permite que os agentes do sistema possam se conectar ao servidor e utilizar as contas cadastradas para enviarem mensagens direcionadas para um determinado agente.

Para isso basta que os agentes saibam o nome de outro agente cadastrado no servidor para enviar a mensagem, o servidor recebe a mensagem, identifica o destinatário e redireciona a mensagem para o destinatário.

3.3 Funcionamento dos agentes:

Os agentes foram implementados usando a classe Agent, com dois comportamentos: OneShotBehaviour, usado no coordenador, e CyclicBehaviour, usado nos agentes de operação e parênteses. Para o envio de mensagens foi utilizada a função Message() para criar as mensagens a serem enviadas e as co rotinas self.receive() e self.send() para fazer, respectivamente, o recebimento e envio de mensagens.

3.3.1 Agente coordenador:

O agente coordenador é responsável por receber a expressão que foi digitada pelo usuário, ele recebe essa expressão como uma cadeia de caracteres, após o recebimento da expressão, que deve ter os operandos e operadores separados por espaço, o coordenador trata essa string da expressão, transformando-a em uma lista de operadores e operandos, então o coordenador utiliza uma função que seleciona o próximo operando de maior prioridade (Ex: o operando de potência). Após selecionar o operando, o coordenador prepara uma mensagem com os operandos envolvidos na operação (Ex: 10 e 2, na operação 10 + 2) para o agente de operação que corresponde àquela operação, por fim o coordenador espera a resposta do agente de operação, ao receber a resposta ele adiciona a resposta à expressão, substituindo o operando e os operadores que foram usados.

O coordenador então fica repetindo esses passos até que a lista da expressão a ser resolvida tenha apenas 1 elemento, significando que a expressão foi resolvida, então o agente mostra o resultado da expressão e termina sua execução. O comportamento usado pelo coordenador é um OneShotBehaviour, que significa que o agente irá receber a expressão e resolvê-la apenas uma vez, ao terminar de resolver uma expressão o agente não repete mais o comportamento.

3.3.2 Agente de parênteses:

O agente de parênteses é responsável por receber uma expressão que estava dentro de parênteses do coordenador e resolver essa sub-expressão fazendo um processo similar ao que o coordenador faz. O agente de parênteses recebe uma expressão completa do coordenador, com operandos e operadores, então o agente seleciona a operação com maior precedência e envia uma mensagem com os operandos para o agente de operação correspondente e fica

esperando a resposta do agente de operação, ele repete até resolver a sub-expressão, com a sub-expressão resolvida o agente envia o resultado para o coordenador que continua a resolução.

3.3.3 Agentes de operação:

O funcionamento dos agentes de operação é simples, eles possuem um comportamento cíclico (CyclicBehaviour), cada agente de operação fica em loop esperando o recebimento de alguma mensagem, quando alguma mensagem é recebida, geralmente os dois operandos da operação, o agente separa esses operandos e os coloca em uma lista e então faz a operação correspondente utilizando esses dois operandos, por exemplo o agente de subtração ao receber a mensagem “10 30”, irá efetuar uma operação de soma utilizando os valores 10 e 30, após resolver a operação o agente envia a resposta para o coordenador contendo o resultado da operação.

3.3.3.1 Agente de soma: Recebe uma mensagem no qual contém dois números separados por espaço, utiliza a função split() para separá-los em uma lista e utiliza os valores da lista para executar a operação enviando o resultado da soma desses dois números para o agente que enviou a mensagem.

3.3.3.2 Agente de subtração: Recebe uma mensagem no qual contém dois números separados por espaço, utiliza a função split() para separá-los em uma lista e utiliza os valores da lista para executar a operação enviando o resultado da diferença desses dois números para o agente que enviou a mensagem.

3.3.3.3 Agente de multiplicação: Recebe uma mensagem no qual contém dois números separados por espaço, utiliza a função split() para separá-los em uma lista e utiliza os valores da lista para executar a operação enviando o resultado da multiplicação desses dois números para o agente que enviou a mensagem.

3.3.3.4 Agente de divisão: Recebe uma mensagem no qual contém dois números separados por espaço, utiliza a função split() para separá-los em uma lista e utiliza os valores da lista para executar a operação enviando o resultado da divisão do primeiro pelo segundo para o agente que enviou a mensagem.

3.3.3.5 Agente de potência: Recebe uma mensagem no qual contém dois números separados por espaço, utiliza a função split() para separá-los em uma lista e utiliza os valores da lista para executar a operação enviando o resultado da potência do primeiro elevado ao segundo para o agente que enviou a mensagem.

3.3.3.6 Agente de raiz quadrada: Recebe uma mensagem no qual contém um número, utiliza esse número para fazer a operação de raiz quadrada e envia o resultado da raiz quadrada desse número para o agente que enviou a mensagem.

3.4 Tratamento da expressão

Para que possamos resolver a equação, precisamos que todos os elementos da equação (operandos e operadores) estejam separados por espaço para que seja possível usar a função split(“ ”) para dividir os elementos da equação em uma lista em que cada operador e cada operando estão em uma posição específica. Após isso, tentamos procurar o operador de maior prioridade da equação através de uma função que percorre a lista da equação e compara os elementos para tentar os operadores e entre esses operadores achar aquele que possui maior prioridade.

A ordem usada para avaliar os operadores foi: parênteses(“(”), potência(“^”), raiz quadrada(“#”), multiplicação(“*”), divisão(“/”), subtração(“-”) e adição(“+”).

Para encontrar os operandos que vão ser usados com a operação, o programa percorre e equação até achar o operador e a partir disso vai refazendo os operandos a partir da posição do operador, ou seja, no operando esquerdo vão ser adicionados elementos um a um em uma string da posição do operador até que seja encontrado outro operador e também é incrementado um contador que representa a distância do último caractere de um operando até o operador e após esse processo ser feito também com o operando da direita esses valores são retornados para o agente coordenador. Depois de acharmos o operador de maior precedência, os operandos e enviarmos a operação para o agente de operação o coordenador insere o resultado da operação na equação e após isso apaga da equação os operandos e o operador que já foram resolvidos até que na equação sobre apenas um elemento que seria o resultado final dela.

4. Resultados

- Exemplo resumido 1:

```
Digite a expressão: 23 + 12 * # 4 + 55 - ( 2 + 4 * 20 ) - ( 2 * ( 8 + 2 ) ) / 2 ^ 2
CoordinatorAgent: Expressao a ser resolvida:  23 + 12 * # 4 + 55 - ( 2 + 4 * 20 ) - ( 2 * ( 8 + 2 ) ) / 2 ^ 2
CoordinatorAgent: Expressao a ser resolvida:  23 + 12 * # 4 + 55 - 82.0 - ( 2 * ( 8 + 2 ) ) / 2 ^ 2
CoordinatorAgent: Expressao a ser resolvida:  23 + 12 * # 4 + 55 - 82.0 - ( 2 * 10.0 ) / 2 ^ 2
CoordinatorAgent: Expressao a ser resolvida:  23 + 12 * # 4 + 55 - 82.0 - 20.0 / 2 ^ 2
CoordinatorAgent: Expressao a ser resolvida:  23 + 12 * # 4 + 55 - 82.0 - 20.0 / 4.0
CoordinatorAgent: Expressao a ser resolvida:  23 + 12 * 2.0 + 55 - 82.0 - 20.0 / 4.0
CoordinatorAgent: Expressao a ser resolvida:  23 + 24.0 + 55 - 82.0 - 20.0 / 4.0
CoordinatorAgent: Expressao a ser resolvida:  23 + 24.0 + 55 - 82.0 - 5.0
CoordinatorAgent: Expressao a ser resolvida:  23 + 24.0 + -27.0 - 5.0
CoordinatorAgent: Expressao a ser resolvida:  23 + 24.0 + -32.0
CoordinatorAgent: Expressao a ser resolvida:  47.0 + -32.0
Resultado da expressão encontrado: 15.0
Agents finished
```

- Exemplo resumido 2:

```
Digite a expressão: 23 + 12 - 55 + ( 2 + 4 ) - 8 / 2 ^ 2
CoordinatorAgent: Expressao a ser resolvida:  23 + 12 - 55 + ( 2 + 4 ) - 8 / 2 ^ 2
CoordinatorAgent: Expressao a ser resolvida:  23 + 12 - 55 + 6.0 - 8 / 2 ^ 2
CoordinatorAgent: Expressao a ser resolvida:  23 + 12 - 55 + 6.0 - 8 / 4.0
CoordinatorAgent: Expressao a ser resolvida:  23 + 12 - 55 + 6.0 - 2.0
CoordinatorAgent: Expressao a ser resolvida:  23 + -43.0 + 6.0 - 2.0
CoordinatorAgent: Expressao a ser resolvida:  23 + -43.0 + 4.0
CoordinatorAgent: Expressao a ser resolvida:  -20.0 + 4.0
Resultado da expressão encontrado: -16.0
Agents finished
```