## Capítulo 3 - Agentes baseados em conhecimento

1. **Projete uma Base de Conhecimento simples em Lógica de Primeira Ordem para o agente “Orquestrador” com regras sobre as capacidades dos outros agentes. Inclua predicados como EspecialistaEm(agente, tópico) e PodeSintetizar(agente).**

1- EspecialistaEm (AgenteA, VisãoComputacional)

2- EspecialistaEm (AgenteB, Planeamento)

3- EspecialistaEm(AgenteC, LinguagemNatural)

4- PodeSintetizar (Osqiestrador)

Por exemplo, no tópico 1 indica que o agenteA é especialista em determinado topico neste caso é especialista em Visão Computacional.

O orquestrador coordena o sistema consultando os agentes especialistas em cada dominio e sintetizando as respostas para tomar decisões globais.

1. **Crie uma regra SE-ENTÃO para o “Orquestrador” que decida quando usar um agente “VerificadorDeFactos”. A condição pode depender do tipo de fonte da informação (e.g., “fonte\_é\_wikipedia”).**

Se a informação pesquisada for aberta ou potencialmente não confiavel então é chamado o agente de VerificadorDeFactos para o orquestrador ter mais precisão e fiabilidade

1. Utilize a representação de Frames para modelar um “Agente de Software” no vosso sistema. Crie um frame genérico e depois frames específicos para “AgenteInvestigador” e “AgenteEscritor”, mostrando a herança de slots.

Frame Agente de Software:

* nome [texto]
* tipo [investigador,escritor]
* objetivo [texto]
* estado [ativo/inativo]
* comunica\_com: [agentes]
* metodo\_executar(): realiza a tarefa principal do agente

Frame AgenteInvestigador:  
Herda-de: Agente de Software:

fontes\_de\_pesquisa: [Google,Base de dados,Wikipedia]

metodo\_executar(): procurar informação credível

validar\_fontes(): verificar se a fonte é confiavel

Frame AgenteEscritor:  
Herda-de: Agente de Software:

estilo\_de\_escrita: [criativo,formal,tecnico]

metodo\_executar(): redigir texto com base nos dados fornecidos

revisa\_texto(): rever, corrigir e melhorar o texto

1. **Como poderia uma Rede Semântica ser usada para representar as relações entre os tópicos de investigação que o “CogniVerse” processa, ajudando o sistema a sugerir tópicos relacionados?**

Uma rede semântica pode representar o conhecimento do CogniVerse como um grafo:

* Os nós representam tópicos de investigação (ex: IA, Machine Learning).
* Já as arestas vão representar relações semânticas entre nós (ex: “é a subárea de”).

Ou seja, quando o sistema processar um tópico ele pode navegar pelos nós conectados para identificar e automáticamente sugerir tópicos relacionados, com base em relações semânticas que existem.

1. **O vosso sistema precisa de decidir se uma fonte de informação é fiável. Proponha 3 regras para um sistema pericial simples que tome esta decisão, com base em atributos como autor\_conhecido publicação\_revisada\_por\_pares, e data\_publicação.**
2. Se o autor\_conhecido = true e publicação\_revisada\_por\_pares = true então fonte\_fiável.
3. Se publicação\_revisada\_por\_pares = false e data\_publicação > 5 anos então fonte\_pouco\_fiável.
4. Se o autor\_conhecido = false e publicação\_revisada\_por\_pares= false então fonte\_não\_fiável.
5. O agente “Orquestrador” recebe um novo pedido: “Criar um resumo sobre IA em medicina”. Descreva como o ciclo TELL e ASK seria usado para interagir com a sua base de conhecimento e iniciar a tarefa.

TELL -> o orquestrador utiliza a base de conhecimento com a nova informação recebida

ASK -> O orquestrador consulta a base de conhecimento para decidir quais agentes podem realizar a tarefa

Resposta da base:

AgenteInvestigador -> recolhe informação

AgenteEscritor -> escreve o resumo

1. Discuta a vantagem de usar uma representação de conhecimento explícita (como regras ou frames) para o “Orquestrador”, em vez de ter a sua lógica de decisão “hard-coded” em Python.

1. Facilidade de atualização: Podem ser feitas novas regras ou alterar políticas sem alteração do código fonte.
2. Transparência: Lógica de decisão mais clara e compreensível, facilitando a revisão e explicação de decisões.
3. Flexibilidade: Permite integrar novos agentes sem reescrever funções complexas.
4. Raciocínio automático: O sistema pode perceber novas conclusões a partir de regras existentes, algo difícil de acontecer com código fixo.