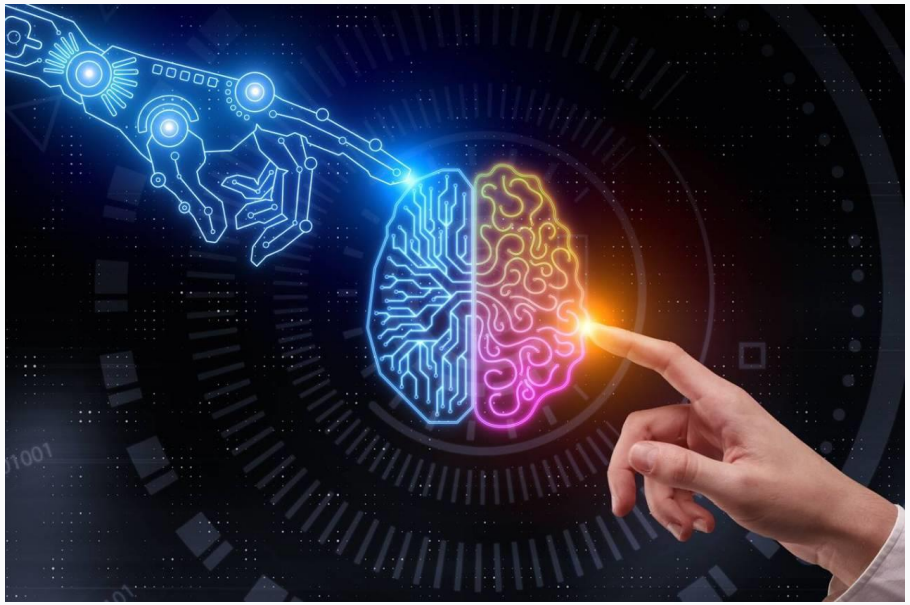


Inteligência Artificial

Representação de conhecimento

Profº - Dr. Thales Levi Azevedo Valente

thales.l.a.valente@gmail.com.br

Grupo da turma 2024.2

https://chat.whatsapp.com/JFB6CgOI7lMCoYmoIKEK62

Sejam Bem-vindos !

Os celulares devem ficar no silencioso ou desligados

Pode ser utilizado apenas em caso de emergência

Boa tarde/noite, por favor e com licença DEVEM ser usados

Educação é essencial

Introdução

"*Se era assim, devia ser; e se fosse assim, teria sido; mas como não é, não é mesmo. Isto é lógica.*"

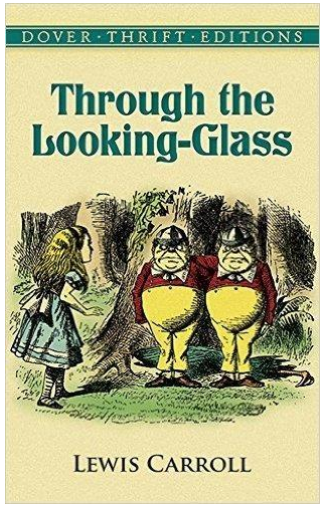
(Lewis Carroll)

LEWIS CARROLL

English logician, mathematician, photographer, and novelist.

Link:

http://academic.eb.com/EBchecked/topic/97087/Lewis-Carroll



Introdução

- **Essa citação de Lewis Carroll, que aparece em Through the Looking-Glass, ilustra o uso da lógica para abordar situações hipotéticas ou irreais. A frase brinca com ideias de causalidade e possibilidade:**
  - ✓ "Se era assim, devia ser": Se algo de fato fosse real, então deveria ser assim por alguma razão lógica.
  - ✓ "Se fosse assim, teria sido": Sugere que, no passado, as condições também deveriam se alinhar com a lógica que sustenta a hipótese.
  - ✓ "Mas como não é, não é mesmo": Refuta tudo isso com a realidade, mostrando que, se algo não é verdadeiro, toda a lógica construída em torno dessa suposição desmorona.

Conhecimento

- **Conhecer é uma operação ativa que precisa da**
  - ✓ Memorização de informações; e do
  - ✓ Bom uso dessas informações

Formas de **representar** os conhecimentos

Engenharia do Conhecimento

- **Área interdisciplinar que une conceitos da**
  - ✓ Inteligência artificial (IA)
  - ✓ Ciência da computação e
  - ✓ Outras disciplinas

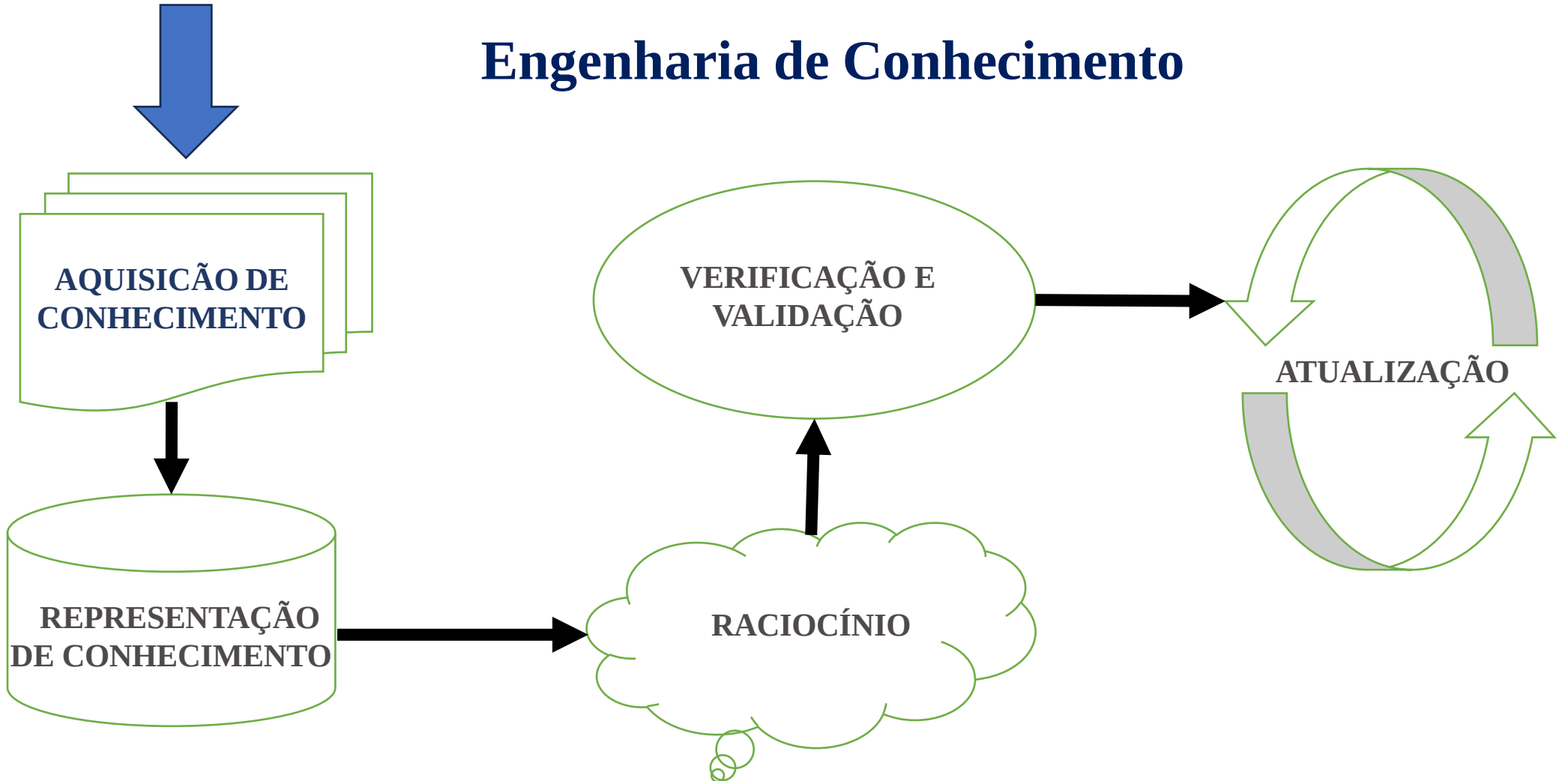
- **Campo fundamental para o desenvolvimento de sistemas baseados em conhecimento como**
  - ✓ Sistemas especialistas
  - ✓ Assistentes virtuais e
  - ✓ Ferramentas de suporte à decisão

Engenharia do Conhecimento

- **Objetivo**
  - ✓ Permitir criar sistemas que possam simular ou auxiliar a inteligência humana em contextos específicos

- **Objetivos Específicos**
  - ✓ Capturar, modelar, organizar e aplicar o conhecimento em sistemas computacionais
  - ✓ Transformar o conhecimento humano em uma forma utilizável por máquinas
  - ✓ Permitir tarefas como raciocínio, aprendizado, tomada de decisão e resolução de problemas

IA: Conhecimento



Engenharia do Conhecimento

Aquisição do Conhecimento

- **Definição**
  - ✓ Processo de obter e organizar o conhecimento necessário para o sistema, diretamente de especialistas, bancos de dados ou de outras fontes

- **Métodos**
  - ✓ Entrevistas com especialistas
  - ✓ Mineração de dados
  - ✓ Extração automática de conhecimento (ex.: algoritmos de aprendizado de máquina).

Engenharia do Conhecimento

Aquisição do Conhecimento – Entrevista com especialistas

- **Definição**
  - ✓ Esse método envolve a interação direta com especialistas humanos para capturar o conhecimento que eles possuem sobre um domínio específico

- **Etapas principais**
  - ✓ **Planejamento**: Definir perguntas específicas ou cenários para explorar o conhecimento do especialista.
  - ✓ **Execução**: Realizar entrevistas detalhadas para documentar os fatos, regras e estratégias usadas pelos especialistas.
  - ✓ **Organização**: Estruturar o conhecimento coletado em formatos utilizáveis, como regras, fluxogramas ou tabelas.

Engenharia do Conhecimento

Aquisição do Conhecimento – Entrevista com especialistas

- **Exemplo prático: imagine o desenvolvimento de um sistema especialista para diagnóstico médico**
  - ✓ Um engenheiro entrevista cardiologistas para entender como eles diagnosticam doenças cardíacas com base em sintomas, exames e histórico do paciente.
  - ✓ O conhecimento capturado é transformado em regras como

> *"Se o paciente tem dor no peito e pressão arterial elevada, então considerar angina como hipótese principal"*

- **Vantagens e Desafios**
  - ✓ Conhecimento é adquirido com maior velocidade
  - ✓ Nem sempre se tem acesso eficiente ao especialista
  - ✓ Captura do conhecimento é subjetivo e suscetível a "ruídos"

Engenharia do Conhecimento

Aquisição do Conhecimento – Mineração de Dados

- **Definição**
  - ✓ Abordagem automatizada ou semi-automatizada para descobrir conhecimento
  - ✓ Utiliza grandes volumes de dados para identificar padrões, regras ou tendências

- **Etapas principais**
  - ✓ **Preparação dos dados**: Coleta e limpeza dos dados para garantir sua qualidade.
  - ✓ **Análise de padrões**: Utiliza técnicas como classificação, regressão, agrupamento (clustering) ou regras de associação para descobrir conhecimento.
  - ✓ **Validação**: Avaliar se os padrões encontrados são úteis e relevantes.

Engenharia do Conhecimento

Aquisição do Conhecimento – Mineração de Dados

- **Exemplo prático: no contexto de sistemas de recomendação, como os usados pela Netflix**
  - ✓ Dados sobre o histórico de visualização de milhões de usuários são minerados.
  - ✓ O sistema recomenda conteúdo personalizado ao descobrir padrões como

    *"**usuários** que **assistem** a filmes de **ação** também **gostam** de séries de **aventura**"*

- **Vantagens e Desafios**
  - ✓ Ideal para domínios com grandes volumes de dados
  - ✓ Descobre conhecimento novo, não evidente para humanos
  - ✓ A qualidade dos padrões depende diretamente da qualidade dos dados
  - ✓ Pode ser difícil interpreter/validar os resultados (muitas vezes especialista de domínio é necessário)

Engenharia do Conhecimento

Aquisição do Conhecimento – Extração Automática

- **Definição**
  - ✓ Utiliza algoritmos de aprendizado de máquina (machine learning) para extrair conhecimento de dados de forma automatizada
  - ✓ Abordagem poderosa para lidar com dados complexos e dinâmicos

- **Etapas principais**
  - ✓ **Treinamento de modelos**: algoritmos são treinados com dados rotulados (supervisionado) ou sem rótulos (não-supervisionado).
  - ✓ **Inferência**: o modelo criado é utilizado para fazer previsões ou classificar novos dados com base no que aprendeu.
  - ✓ **Atualização**: o modelo pode ser ajustado continuamente com novos dados.

Engenharia do Conhecimento

Aquisição do Conhecimento – Extração Automática

- **Exemplo prático: No desenvolvimento de um sistema de diagnóstico por imagem médica**
  - ✓ Um algoritmo de aprendizado de máquina é treinado com milhares de imagens rotuladas, indicando se há ou não uma lesão.
  - ✓ Após o treinamento, o sistema pode identificar automaticamente anomalias em imagens de raio-X ou ressonâncias magnéticas
- **Vantagens e Desafios**
  - ✓ Pode lidar com grandes volumes de dados complexos
  - ✓ Capaz de aprender e melhorar com o tempo
  - ✓ Requer dados de alta qualidade para treinamento.Modelos de aprendizado profundo podem ser difíceis de interpretar (caixa-preta).

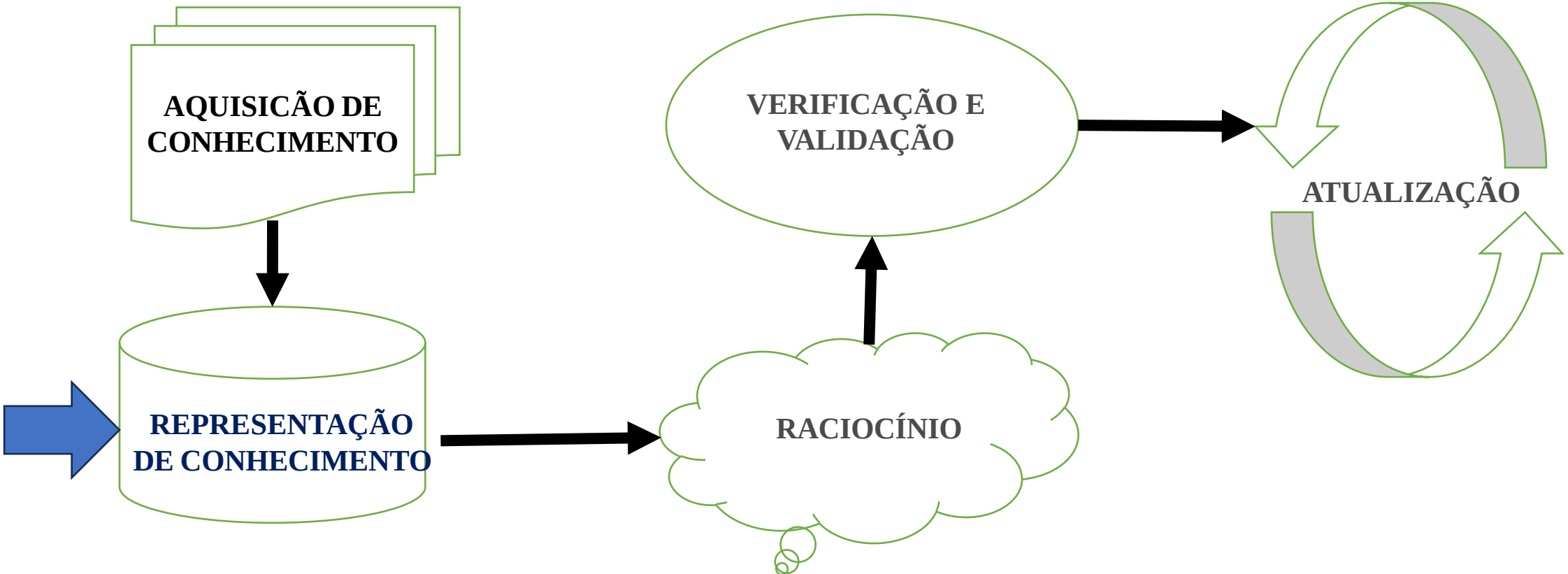
Engenharia do Conhecimento

Aquisição do Conhecimento – Resumo

Método	Vantagens	Desafios
Entrevistas com Especialistas	Conhecimento especializado direto.	Demorado; subjetividade na captura.
Mineração de Dados	Descobre padrões úteis em grandes volumes de dados.	Depende da qualidade dos dados; difícil interpretar padrões.
Extração Automática (ML)	Aprendizado contínuo; ideal para dados dinâmicos e complexos.	Requer dados de alta qualidade; modelos podem ser opacos.

IA: Conhecimento

Engenharia de Conhecimento



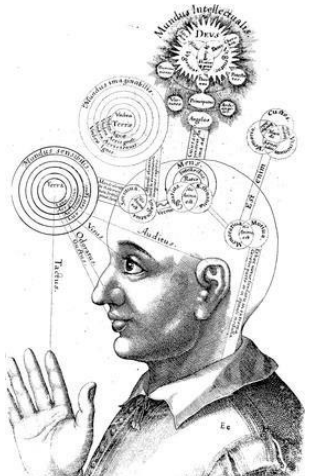
Engenharia do Conhecimento

Representação de Conhecimento

- **Definição**
  - ✓ Estruturar o conhecimento para que ele seja compreensível e utilizável por sistemas computacionais

"*Knowledge representation (KR) is the study of how knowledge about the world can be represented and what kinds of reasoning can be done with that knowledge*".

→ *Visa levar o "conhecimento" para a máquina e dotá-la de alguma capacidade de raciocínio (reasoning).*



Engenharia do Conhecimento

Representação de Conhecimento

- **Definição**
  - ✓ Estruturar o conhecimento para que ele seja compreensível e utilizável por sistemas computacionais
  - ✓ Convenção sintática e semântica para descrição das informações
  - ✓ Deve explicitar o conhecimento e ser manipulável
- **Ou seja, consiste em estruturar e organizar o conhecimento adquirido de maneira que ele possa ser processado por máquinas para:**
  - ✓ Realizar inferências
  - ✓ Resolver problemas
  - ✓ Tomar decisões
  - ✓ Ser atualizado

Engenharia do Conhecimento

Representação de Conhecimento

- **Objetivos específicos**
  - ✓ Expressar conhecimento de forma clara e não ambígua
  - ✓ Permitir raciocínio automático, como deduções, previsões e diagnósticos
  - ✓ Adaptar-se ao domínio de aplicação (ex.: medicina, direito, engenharia)
  - ✓ Ser flexível para atualização e expansão
- **Aplicações**
  - ✓ Sistemas especialistas (médicos, industriais)
  - ✓ Agentes inteligentes (assistentes virtuais, chatbots)
  - ✓ Mineração de texto e análise de dados
  - ✓ Robótica (planejamento e tomada de decisão)

Engenharia do Conhecimento

Representação de Conhecimento

- **O que é uma boa representação?**
  - ✓ Adequação da representação
  - ✓ Adequação da inferência
  - ✓ Eficiência da inferência
  - ✓ Eficiência da aquisição

Engenharia do Conhecimento

Representação de Conhecimento

- **Adequação da representação**
  - ✓ É o poder da representação (expressividade)
- **Adequação da inferência**
  - ✓ Uma representação deve permitir fazer inferências, ou seja, deduções de novos conhecimentos
- **Eficiência da inferência**
  - ✓ Favorecer (usando algumas informações adicionais) alguns caminhos de pesquisa
- **Eficiência da aquisição**
  - ✓ Facilitar a aquisição de novos conhecimentos pelo usuário e/ou pelo sistema

Engenharia do Conhecimento

Representação de Conhecimento

- **Visão Geral das Principais Técnicas**

Técnica	Características Principais	Aplicação Típica
Regras de Produção	Baseadas em IF-THEN (SE-ENTÃO)	Sistemas especialistas
Redes Semânticas	Grafo de conceitos e relações	Relações conceituais, domínios moderados
Lógica de Primeira Ordem	Base formal para dedução e provas	Raciocínio dedutivo, provas matemáticas
Ontologias	Estruturas hierárquicas de conceitos	Modelagem de domínios, Web Semântica
Redes Bayesianas	Grafos probabilísticos para incerteza	Domínios incertos (diagnóstico médico, previsão)

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules)

- **Definição**
  - ✓ Uma regra de produção pode ser vista como uma declaração condicional no formato SE (condição) ENTÃO (ação/conclusão)
  - ✓ Condições normalmente consultam fatos sobre o estado atual do sistema (ou base de conhecimento)
  - ✓ As ações podem tanto modificar o estado do sistema, registrar uma conclusão ou disparar outras regras
- **Destaque Histórico**
  - ✓ Década de 1970 e 1980, com o desenvolvimento dos primeiros Sistemas Especialistas (ex.: MYCIN, para diagnóstico médico, e DENDRAL, para análise química)

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Vantagens e Desvantagens

Vantagens	Descrição
Simplicidade e Intuitividade	As regras SE-ENTÃO são muito próximas do raciocínio humano, facilitando desenvolvimento e comunicação com especialistas
Modularidade	Cada regra pode ser vista como uma “peça” independente. É relativamente simples adicionar novas regras sem afetar as existentes (em domínios pequenos)
Execução Fácil de Rastrear	Podemos acompanhar por que uma ação foi disparada (explicabilidade), útil para auditoria e depuração

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Vantagens e Desvantagens

Desvantagens	Descrição
Escalabilidade Limitada	À medida que o número de regras cresce, a manutenção se torna complexa. Conflitos e redundâncias podem surgir
Gerenciamento Complexo	Atualizações e mudanças no domínio exigem revisões das regras existentes. Sem uma estratégia robusta de versionamento, pode haver inconsistência
Dificuldade de Capturar Incerteza	Modelar situações com probabilidades ou incerteza explícita é mais complicado. Regras puramente lógicas não lidam bem com graus de incerteza (embora existam extensões)

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules)

- **Conceito Central**

> *"Se uma condição for satisfeita, então execute uma ação ou conclua algo"*

- **Mecanismos de Inferência**
  - ✓ **Encadeamento para Frente (Forward Chaining)**: parte dos fatos até chegar a uma conclusão
  - ✓ **Encadeamento para Trás (Backward Chaining)**: parte de uma meta (hipótese) e busca os fatos que a suportam

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Encadeamento para Frente (Forward Chaining)

- **Descrição**
  - ✓ Iniciamos a partir de um conjunto de fatos conhecidos na base de conhecimento
  - ✓ Cada fato é comparado com as condições de todas as regras.
  - ✓ Se uma regra tiver suas condições satisfeitas, sua ação é executada, possivelmente gerando novos fatos
  - ✓ Este processo continua até que nenhuma nova regra possa ser disparada ou até se chegar a uma conclusão desejada

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Encadeamento para Frente (Forward Chaining)

- **Aplicação Típica**
  - ✓ Sistemas reativos, como controle de processos industriais (ex.: se certo sensor atinge valor X, dispare regra Y

- **Exemplo**
  - ✓ **Fato:** Temperatura = 102°C
  - ✓ **Regra:** SE temperatura > 100°C ENTÃO desligar_motor
  - ✓ **Após disparo, adiciona-se novo fato:** motor_desligado = verdadeiro.
  - ✓ **Isso pode disparar nova regra:** SE motor_desligado e temperatura > 100°C ENTÃO ativar_sistema_resfriamento.

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Encadeamento para Frente (Forward Chaining)

- **Exemplo de disparo de regras**
  - ✓ Vamos supor um cenário em que temos duas regras
  - ✓ **Regra 1:** SE temperatura > 100°C ENTÃO desligar_motor
  - ✓ **Regra 2:** SE motor_desligado E temperatura > 100°C ENTÃO ativar_sistema_resfriamento

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Encadeamento para Frente (Forward Chaining)

- **Passo a Passo (Encadeamento para Frente)**
  - ✓ O sistema lê um sensor que indica temperatura = 102°C
  - ✓ Verifica a Regra 1: a condição "temperatura > 100°C" é satisfeita
  - ✓ Dispara a ação "desligar_motor", gerando um novo fato: motor_desligado = verdadeiro
  - ✓ Agora, com esse fato, o sistema verifica novamente todas as regras
  - ✓ A Regra 2 tem como condição "motor_desligado E temperatura > 100°C".
  - ✓ Ambos são verdadeiros (motor_desligado é verdadeiro e temperatura = 102°C)
  - ✓ Dispara a ação: "ativar_sistema_resfriamento"
  - ✓ Após isso, não há mais regras que possam ser disparadas com base nos fatos atuais

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Encadeamento para Trás (Backward Chaining)

- **Descrição**
  - ✓ Iniciamos de uma meta (por exemplo, “Quero concluir se o motor deve ser desligado”)
  - ✓ Procuramos regras que concluam essa meta como ação
  - ✓ Para cada regra relevante, verificamos se suas condições são satisfeitas, potencialmente gerando outras submetas a serem checadas
  - ✓ Processo se repete até que as condições sejam confirmadas ou negadas

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Encadeamento para Trás (Backward)

- **Aplicação Típica**
  - ✓ Sistemas de diagnóstico médicos ou industriais por exemplo, onde se parte de uma hipóteses (“O paciente tem doença X?”/ Devo ativar o resfriamento?) e busca-se evidências para confirmá-la

- **Exemplo**
  - ✓ **Meta:** Determinar se devemos ativar_sistema_resfriamento
  - ✓ **Regra:** SE motor_desligado E temperatura > 100°C ENTÃO ativar_sistema_resfriamento
  - ✓ **Precisamos checar “motor_desligado = verdadeiro?” e “temperatura > 100°C?”**
  - ✓ **Caso “motor_desligado” não seja conhecido,** partimos em busca de regras que concluam esse fato

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Encadeamento para Trás (Backward Chaining)

- **Passo a Passo (Encadeamento para Trás)**
  - ✓ **O sistema começa verificando se deve ativar_sistema_resfriamento**
  - ✓ **Meta:** ativar_sistema_resfriamento = verdadeiro
  - ✓ **Identificar regras relevantes: A Regra 2 conclui a ação ativar_sistema_resfriamento**
  - ✓ **Condições da Regra 2:** motor_desligado = verdadeiro E temperatura > 100°C
  - ✓ O sistema precisa verificar se ambas as condições são verdadeiras
  - ✓ **Submetas para verificar as condições da Regra 2**
  - ✓ Submeta 1: Verificar se motor_desligado = verdadeiro
  - ✓ Submeta 2: Verificar se temperatura > 100°C.

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Encadeamento para Trás (Backward Chaining)

- **Passo a Passo (Encadeamento para Trás)**
  - ✓ **O sistema começa verificando se deve ativar_sistema_resfriamento**
  - ✓ **Meta:** ativar_sistema_resfriamento = verdadeiro
  - ✓ **Identificar regras relevantes: A Regra 2 conclui a ação ativar_sistema_resfriamento**
  - ✓ **Condições da Regra 2:** motor_desligado = verdadeiro E temperatura > 100°C
  - ✓ O sistema precisa verificar se ambas as condições são verdadeiras
  - ✓ **Submetas para verificar as condições da Regra 2**
  - ✓ Submeta 1: Verificar se motor_desligado = verdadeiro
    - ✓ Para satisfazer essa condição, o sistema procura uma regra que conclua motor_desligado = verdadeiro
    - ✓ A Regra 1 conclui que motor_desligado = verdadeiro se temperatura > 100°C
    - ✓ Portanto, o sistema precisa verificar a condição da Regra 1: temperatura > 100°C.

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Encadeamento para Trás (Backward Chaining)

- **Passo a Passo (Encadeamento para Trás)**
  - ✓ **Submetas para verificar as condições da Regra 2**
  - ✓ Submeta 1: Verificar se motor_desligado = verdadeiro
  - ✓ Submeta 2: Verificar se temperatura > 100°C
    - ✓ Esta condição pode ser confirmada diretamente pelos fatos conhecidos
    - ✓ O sensor do sistema indica que temperatura = 102°C, satisfazendo temperatura > 100°C
  - ✓ Conclusão da Submeta 1
    - ✓ Com base nos fatos, temperatura > 100°C é verdadeira, então a Regra 1 é disparada, concluindo que motor_desligado = verdadeiro
  - ✓ Conclusão da Submeta 2
    - ✓ O sistema já confirmou que temperatura > 100°C

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Encadeamento para Trás (Backward Chaining)

- **Passo a Passo (Encadeamento para Trás)**
  - ✓ **Submetas para verificar as condições da Regra 2**
  - ✓ Submeta 1: Verificar se motor_desligado = verdadeiro
  - ✓ Submeta 2: Verificar se temperatura > 100°C
  - ✓ Conclusão da Submeta 1
  - ✓ Conclusão da Submeta 2
  - ✓ Resultado Final
    - ✓ Com motor_desligado = verdadeiro e temperatura > 100°C, a Regra 2 pode ser disparada, e o sistema conclui que ativar_sistema_resfriamento = verdadeiro

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Tabela Comparativa

Aspecto	Encadeamento para Frente	Encadeamento para Trás
Ponto de Partida	Fatos conhecidos	Meta ou conclusão desejada
Objetivo	Identificar todas as possíveis conclusões	Verificar se uma meta específica é verdadeira
Uso Típico	Controle de processos, sistemas reativos	Diagnóstico, sistemas baseados em hipóteses
Exemplo	Sistema de controle de temperatura	Diagnóstico de falha em um motor

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules)

Tabelas Comparativas

Aspecto	Encadeamento para Frente	Encadeamento para Trás
Vantagens	➢ Executa de forma automática para explorar todas as possibilidades ➢ Útil para sistemas onde todas as ações potenciais precisam ser consideradas ➢ Boa para sistemas reativos e em tempo real	➢ Focado na meta, economizando processamento desnecessário ➢ Ideal para sistemas de diagnóstico e de inferência direcionada ➢ Avalia apenas o necessário para atingir a meta
Limitações	➢ Pode ser ineficiente quando há muitas regras e poucas são relevantes ➢ Não direcionado, avalia todas as possibilidades	➢ Ineficiente quando a cadeia de dependências é muito longa ou complexa ➢ Requer que todas as condições intermediárias sejam verificáveis
Estratégia	Parte dos fatos e tenta disparar todas as regras possíveis até que não reste nenhuma regra a ser disparada	Parte da meta desejada, buscando as regras que a satisfaçam e verificando recursivamente suas condições

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Boas Práticas no Uso de Regras de Produção

- **Agrupamento Temático de Regras**
  - ✓ Organizar regras relacionadas a cada subdomínio em blocos ou módulos
  - ✓ Ex.: “Regras de controle de temperatura”, “Regras de segurança do motor”, “Regras de diagnóstico de falhas elétricas”, etc

- **Uso de Variáveis e Funções**
  - ✓ Regras podem ter variáveis que tornam mais geral o processo de disparo
  - ✓ Ex.: SE (temperaturaMotor(X) > limiteMax(X)) ENTÃO desligar_motor(X).

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Boas Práticas no Uso de Regras de Produção

- **Ferramentas de Versionamento e Testes**
  - ✓ Manter as regras em repositórios que permitam controle de versão (Git)
  - ✓ Criar uma suite de testes para validar cada nova regra ou alteração nas regras existentes

- **Integração com Outras Abordagens**
  - ✓ Para lidar com incerteza, pode-se usar regras de produção em conjunto com fatores de certeza ou módulos Bayesianos, dependendo da complexidade do projeto
  - ✓ Para grandes bases de conhecimento, combinar regras de produção com ontologias ou bases semânticas pode trazer mais robustez e organização

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas

- **Cenário**
  - ✓ Imagine uma estufa automatizada para gerenciar a temperatura, umidade, CO_2, luz, entre outros fatores, garantindo o crescimento ideal das plantas

- **Motivação**
  - ✓ Queremos um sistema de controle de estufa que, em condições normais, ligue o aquecimento quando a temperatura estiver abaixo de 20°C
  - ✓ Entretanto, se houver a meta de "economia de energia", vamos restringir o uso do aquecimento apenas se estiver muito frio (por exemplo, abaixo de 15°C)
  - ✓ Em Prolog "puro", não existe um mecanismo automático de prioridade de regras. A ideia é estruturar o raciocínio para que uma regra mais restrita (aquecimento prioritário) seja consultada antes da mais geral (aquecimento normal).

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas

- **Regras Básicas Propostas**
  - ✓ Regra de Irrigação
    - ✓ SE Umidade < 30% ENTÃO ativar_irrigacao
  - ✓ Regra de Ventilação
    - ✓ SE Temperatura > 35°C ENTÃO ligar_ventilacao
  - ✓ Regra de Aquecimento
    - ✓ SE Temperatura < 20°C ENTÃO ligar_aquecimento

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Regras e Fatos

- **Fatos que descrevem a situação atual da estufa**
  - ✓ Exemplo, umidade(Valor), temperatura(Valor), etc

- **Regras (no estilo "SE condição ENTÃO conclusão")**
  - ✓ Em Prolog, normalmente lidamos com "cabeça :- corpo.“
  - ✓ A "cabeça" é o que concluímos; o "corpo" é a condição

```
ativar_irrigacao :-
    umidade(Um),
    Um < 30.

ligar_ventilacao :-
    temperatura(T),
    T > 35.

ligar_aquecimento :-
    temperatura(T),
    T < 20.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Conflito de Regras

- **Cenário: A temperatura está em 18°C (o que sugeriria ligar_aquecimento), mas o sistema quer economizar energia e evitar uso excessivo de aquecedores.**
  - ✓ Questão: Como priorizar regras conflitantes? (ex.: conforto vs. economia)

```
ativar_irrigacao :-
   umidade(Um),
   Um < 30.

ligar_ventilacao :-
   temperatura(T),
   T > 35.

ligar_aquecimento :-
   temperatura(T),
   T < 20.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Conflito de Regras

- **Cenário: A temperatura está em 18°C (o que sugeriria ligar_aquecimento), mas o sistema quer economizar energia e evitar uso excessivo de aquecedores.**
  - ✓ Questão: Como priorizar regras conflitantes? (ex.: conforto vs. economia)

- **Solução Proposta 1: atribuir prioridades (pesos)**
  - ✓ Em Prolog "puro", não há um mecanismo nativo de "prioridade de regra". Porém, podemos estruturar o raciocínio
    - ✓ Primeiro, verificar se há uma meta de economia de energia
    - ✓ Se for crucial manter baixa energia, só ligamos o aquecimento se estiver muito frio (ex.: temperatura < 15°C).

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Conflito de Regras

- **Solução Proposta 1: atribuir prioridades (pesos)**
  - ✓ Em Prolog "puro", não há um mecanismo nativo de "prioridade de regra". Porém, podemos estruturar o raciocínio
    - ✓ Primeiro, verificar se há uma meta de economia de energia
    - ✓ Se for crucial manter baixa energia, só ligamos o aquecimento se estiver muito frio (ex.: temperatura < 15°C).

```
ligar_aquecimento_prioritario :-
    temperatura(T),
    objetivo(economia_energia),
    T < 15.  % Regra fica mais "restrita"

ligar_aquecimento_normal :-
    temperatura(T),
    \+ objetivo(economia_energia),  %
Não estamos economizando energia
    T < 20.

ligar [illegible]to :-
    te[illegible]ra(
    T <
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Uso do dynamic

- **Vamos usar** :- dynamic **e definir** temperatura/1 **e** objetivo/1
  - ✓ :- dynamic: Permite alterar dinamicamente os fatos temperatura/1 e objetivo/1
  - ✓ Em Prolog, temperatura/1 significa que o predicado chamado temperatura tem aridade 1
  - ✓ A aridade é o número de argumentos que um predicado aceita. No caso de temperatura/1, o predicado aceita 1 argumento. Por exemplo:
    - ✓ Exemplo: temperatura(20).
    - ✓ Aqui, o predicado temperatura possui um argumento, que é o valor 20.

```
:- dynamic(temperatura/1).
:- dynamic( objetivo/1).

ligar_aquecimento_prioritario :-
   temperatura(T),
   objetivo(economia_energia),
   T < 15.  % Regra fica mais "restrita"

ligar_aquecimento_normal :-
   temperatura(T),
   \+ objetivo(economia_energia),  %
Não estamos economizando energia
   T < 20.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Aridade

- **Como identificar aridade em Prolog?**
  - ✓ Exemplos
    - ✓ temperatura/1: Predicado com 1 argumento
    - ✓ objetivo/1: Predicado com 1 argumento.
    - ✓ verifica/2: Predicado com 2 argumentos.
    - ✓ status/3: Predicado com 3 argumentos.
- **Por que usamos a aridade em Prolog?**
  - ✓ A aridade permite que Prolog diferencie predicados com o mesmo nome, mas com números diferentes de argumentos

```
:- dynamic(temperatura/1).
:- dynamic( objetivo/1).

ligar_aquecimento_prioritario :-
   temperatura(T),
   objetivo(economia_energia),
   T < 15.  % Regra fica mais "restrita"

ligar_aquecimento_normal :-
   temperatura(T),
   \+ objetivo(economia_energia),  %
Não estamos economizando energia
   T < 20.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Aridade

- **Por que a aridade não foi especificada diretamente no código em ligar_aquecimento_prioritario?**
  - ✓ Aridade em Prolog é implícita na definição do predicado quando este é sem argumentos
    - ✓ Prolog automaticamente entende que ele tem aridade 0.
- **Predicados sem argumentos**
  - ✓ ligar_aquecimento_prioritario/0 e ligar_aquecimento_normal/0 são projetados para apenas verificar condições internas (usando fatos como temperatura/1 e objetivo/1) e retornar true ou false.
  - ✓ Por isso, não é necessário passar argumentos diretamente para eles.

```
:- dynamic(temperatura/1).
:- dynamic( objetivo/1).

ligar_aquecimento_prioritario :-
   temperatura(T),
   objetivo(economia_energia),
   T < 15.  % Regra fica mais "restrita"

ligar_aquecimento_normal :-
   temperatura(T),
   \+ objetivo(economia_energia),  %
Não estamos economizando energia
   T < 20.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Código Completo

```
% Initialization:- initialization(main).

% Função Main
main :-
    % Definindo fatos iniciais
    retractall(temperatura(_)),  % Remove qualquer fato de temperatura anterior
    retractall(objetivo(_)),     % Remove qualquer objetivo anterior
    assertz(temperatura(14)),    % Define a temperatura atual como 14°C
    assertz(objetivo(economia_energia)),  % Define o objetivo como economia de
energia

    % Chama o predicado para ligar o aquecimento
    ligar_aquecimento.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Código Completo

```
% Função principal para decidir qual aquecimento ligar
ligar_aquecimento :-
      ligar_aquecimento_prioritario,
      !.
% O corte impede que outras regras sejam verificadas.
ligar_aquecimento :-
     ligar_aquecimento_normal,
     !.
ligar_aquecimento :-
     write('Nenhuma regra de aquecimento aplicada.'), nl,    fail.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Código Completo

```
% Declarando predicados dinâmicos para permitir alteração durante execução
:- dynamic(temperatura/1).
:- dynamic( objetivo/1).

% Regras de Aquecimento
ligar_aquecimento_prioritario :-
    temperatura(T),
    objetivo(economia_energia),
    T < 15.  % Liga o aquecimento prioritário se a temperatura estiver abaixo de 15°C.
    write('Aquecimento Prioritário Ligado!'), nl.

ligar_aquecimento_normal :-
    temperatura(T),
    \+ objetivo(economia_energia),  % Verifica se não estamos em economia de energia
    T < 20.  % Liga o aquecimento normal se a temperatura estiver abaixo de 20°C.
    write('Aquecimento Normal Ligado!'), nl.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Explicação

- **Função Main**
  - ✓ Remover fatos antigos
    - ✓ retractall(temperatura(_)) e retractall(objetivo(_)) apagam todos os fatos já existentes para os predicados temperatura/1 e objetivo/1. Isso garante que estamos começando "do zero", sem configurações passadas.
  - ✓ Definir novos valores
    - ✓ assertz(temperatura(14)) insere no banco de fatos que a temperatura atual é de 14°C.
    - ✓ assertz(objetivo(economia_energia)) define o objetivo como economia_energia.
    - ✓ Esses novos fatos serão usados pelas regras que decidem se devemos ligar o aquecimento
  - ✓ Chamar o predicado principal
    - ✓ ligar_aquecimento é o predicado que contém a lógica para determinar qual tipo de aquecimento deve ser ligado (ou se nenhum deve ser ligado)

```
% Initialization:- initialization(main).

% Função Main
main :-
    % Definindo fatos iniciais
    retractall(temperatura(_)),
    retractall(objetivo(_)),
    assertz(temperatura(14)),
    assertz(objetivo(economia_energia)),

    ligar_aquecimento.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Explicação

- **Predicados Dinâmicos**
  - ✓ Significa que temperatura/1 e objetivo/1 podem ser alterados (inseridos ou removidos) durante a execução do programa.
  - ✓ Sem essa declaração, o Prolog impediria mudanças nesses fatos após o início do programa

```
:- dynamic(temperatura/1).
:- dynamic( objetivo/1).
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Explicação

- **Regras de Aquecimento**
  - ✓ ligar_aquecimento_prioritario/0: verifica duas condições simultaneamente
    - ✓ A temperatura atual (T) é menor que 15°C.
    - ✓ O objetivo do sistema é economia_energia.
    - ✓ Quando é bem-sucedido? Se ambas as condições forem verdadeiras ao mesmo tempo, o Prolog considera essa regra satisfatória. Significa que devemos ligar o aquecimento prioritário (porque está muito frio e estamos em modo de economia de energia)

```
ligar_aquecimento_prioritario :-
    temperatura(T),
    objetivo(economia_energia),
    T < 15.
```

  - ✓ ligar_aquecimento_normal/0: Verifica outras duas condições
    - ✓ A temperatura atual (T) é menor que 20°C.
    - ✓ O objetivo não é economia_energia (representado por \+ objetivo(economia_energia)).
    - ✓ Quando é bem-sucedido? Se a temperatura estiver abaixo de 20°C e o objetivo do sistema não for economia de energia, aciona-se o aquecimento normal

```
ligar_aquecimento_normal :-
    temperatura(T),
    \+ objetivo(economia_energia),
    T < 20.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Explicação

- **Predicado Principal:** ligar_aquecimento/0

```
ligar_aquecimento :-
   ligar_aquecimento_prioritario,
   !.
```

  - ✓ Primeira verificação
    - ✓ Tenta executar ligar_aquecimento_prioritario.
    - ✓ Se a condição para aquecimento prioritário for satisfeita (temperatura < 15 e objetivo = economia_energia), escreve "Aquecimento Prioritário Ligado!" na tela.
    - ✓ O operador ! (corte) interrompe qualquer tentativa de buscar outras regras depois desta, pois já se encontrou a decisão correta.
  - ✓ Segunda verificação
    - ✓ Se a primeira regra falhou (ou seja, não foi possível ligar o aquecimento prioritário), o Prolog tenta ligar_aquecimento_normal.
    - ✓ Se a condição para aquecimento normal for satisfeita (temperatura < 20 e objetivo != economia_energia), escreve "Aquecimento Normal Ligado!".
    - ✓ Novamente, o ! corta o restante das regras.

```
ligar_aquecimento :-
   ligar_aquecimento_normal,
   !.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Explicação

- **Predicado Principal:** ligar_aquecimento/0
  - ✓ Caso nenhum seja aplicado
    - ✓ Se as duas condições (prioritário e normal) falharem, resta esta cláusula, que escreve "Nenhuma regra de aquecimento aplicada." e retorna fail.
    - ✓ Isso indica ao Prolog que não foi possível ligar o aquecimento sob nenhuma das condições.

```
ligar_aquecimento :-
    write('Nenhuma regra de aquecimento aplicada.'), nl,
    fail.
```

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Explicação

- **Como tudo se conecta?**
  - ✓ Ao chamar main
    - ✓ Remove fatos antigos (garantindo um estado limpo).
    - ✓ Define temperatura(14) e objetivo(economia_energia).
  - ✓ Chama ligar_aquecimento
    - ✓ O Prolog primeiro tenta ligar_aquecimento_prioritario.
    - ✓ Verifica se temperatura(14) < 15 (verdadeiro) e se objetivo(economia_energia) (verdadeiro).
    - ✓ Como ambas condições são satisfeitas, ele exibe “Aquecimento Prioritário Ligado!” e para (corte).
  - ✓ Se a regra prioritária não fosse satisfeita
    - ✓ O Prolog passaria para a segunda regra, tentaria ligar_aquecimento_normal.
    - ✓ Se essa também falhasse, exibiria “Nenhuma regra de aquecimento aplicada.” e retornaria fail..

Engenharia do Conhecimento

Representação de Conhecimento - Regras de Produção (Production Rules):
Sistema de Gerenciamento de Clima em Estufas – Explicação

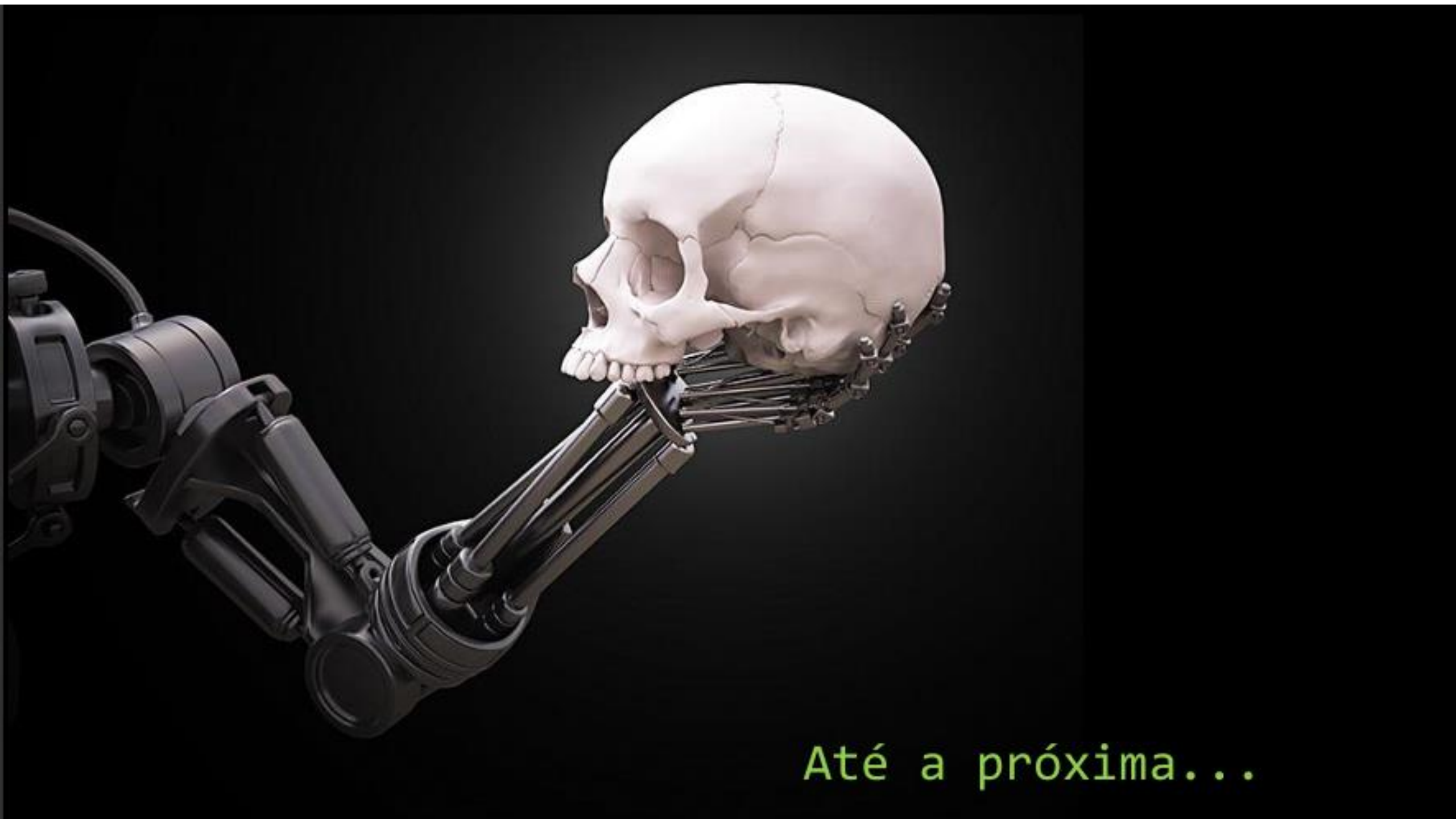
- **Resumo**
  - ✓ main: Ponto de entrada que define o estado inicial (temperatura, objetivo) e chama a decisão de aquecimento.
  - ✓ Regras: Checam condições (temperatura e objetivo) para escolher o tipo de aquecimento.
  - ✓ Corte (!): Assim que uma regra é satisfeita, o programa interrompe a verificação das regras subsequentes.
  - ✓ Executando: Ao rodar main., você verá a mensagem sobre qual aquecimento foi ligado (ou se nenhum foi ligado).

- **Esse é o fluxo geral que mostra como, em Prolog, definimos fatos dinâmicos, limpamos o estado antigo, estabelecemos um novo estado e, então, chamamos as regras para tomar decisões com base nesses fatos.**

Dúvidas?

Atividade de Fixação

teaching/artificial-intelligence/exercises/prolog at main · thalesvalente/teaching · GitHub


Até a próxima...

Apresentador

Thales Levi Azevedo Valente

E-mail:
thales.l.a.valente@gmail.com

Referências

- Links referenciados nos respectivos slides.
- T.B. Borchartt . *Introdução à Inteligência Artificial*. 2024. 37 slides. Universidade Federal do Maranhão.
- A.O. B. Filho. *Inteligência Artificial - Introdução*. 2024. 31 slides. Universidade Federal do Maranhão.