



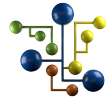
Formação Inteligência Artificial



Introdução à Inteligência Artificial

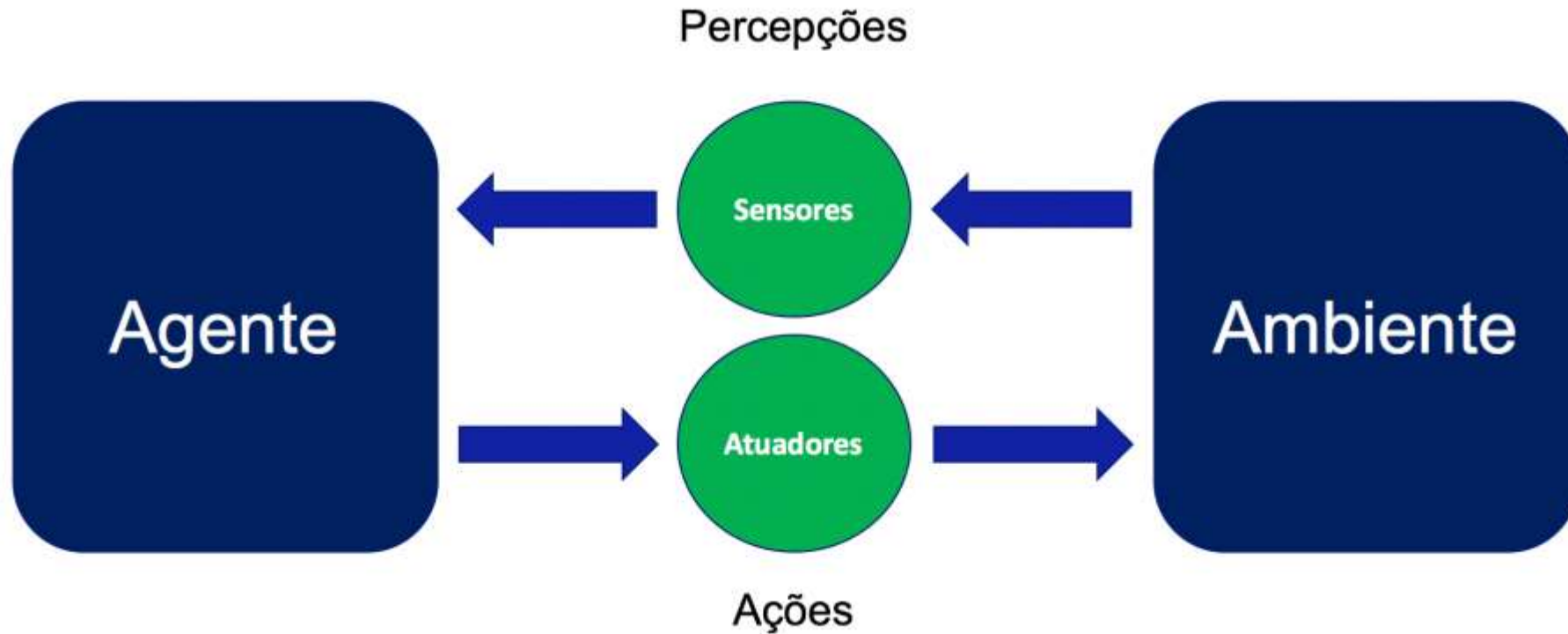


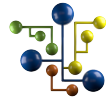
Agentes Lógicos



Agentes Inteligentes

Solução de Problemas





Agentes Inteligentes

Solução de Problemas

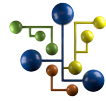
7	2	4
5		6
8	3	1

Estado Inicial

	1	2
3	4	5
6	7	8

Estado Objetivo



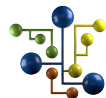


Agentes Lógicos

Baseados em Conhecimento

Agentes que podem formar representações do mundo, usar um processo de inferência para derivar novas representações sobre o mundo e utilizar essas novas representações para deduzir o que fazer

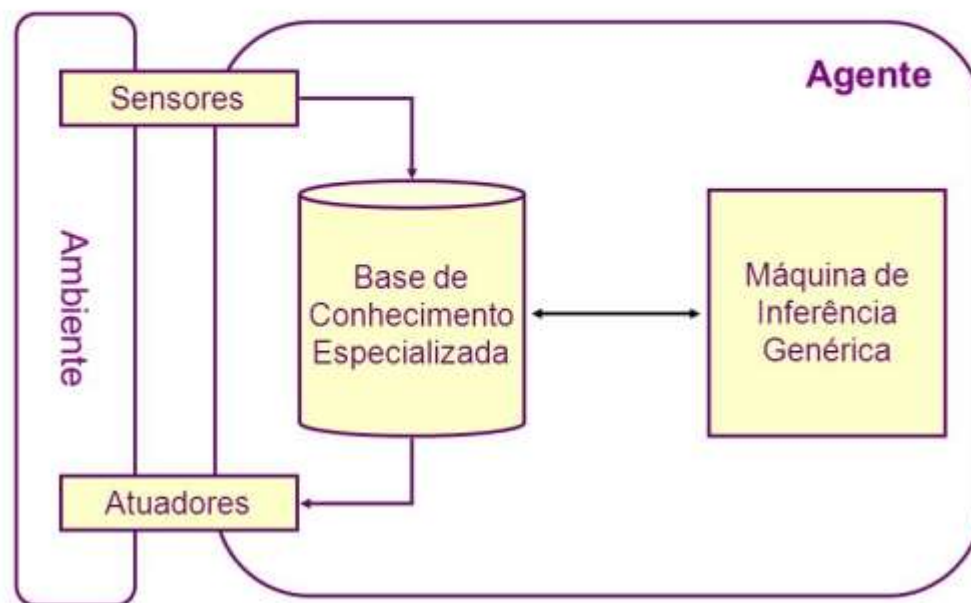


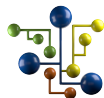


Agentes Lógicos

Baseados em Conhecimento

Agente Baseado em Conhecimento

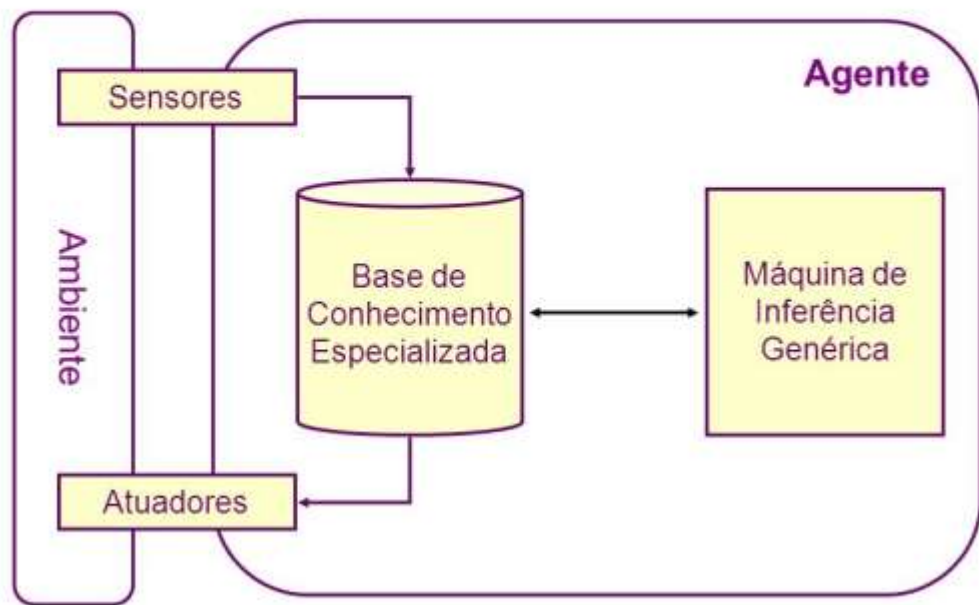




Agentes Lógicos

Baseados em Conhecimento

Agente Baseado em Conhecimento



E para que esse conhecimento seja adquirido e aprendido, desenvolvemos a lógica como uma classe geral de representações para apoiar agentes baseados no conhecimento.



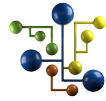


Agentes Lógicos

Neste Capítulo:

- Agentes Baseados em Conhecimento
- Lógica e Lógica Proposicional
- Lógica de Primeira Ordem
- Sintaxe e Semântica
- Inferência Proposicional
- Encadeamento
- Planejamento Clássico
- Planejamento Multiagente





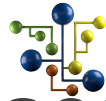
Agentes Lógicos



- Configurar o Ambiente de Desenvolvimento
- Atividades Práticas
 - Desenvolver um Agente em Python
 - Algoritmos de Busca em Python



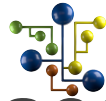
Agentes Baseados em Conhecimento



Agentes Baseados em Conhecimento

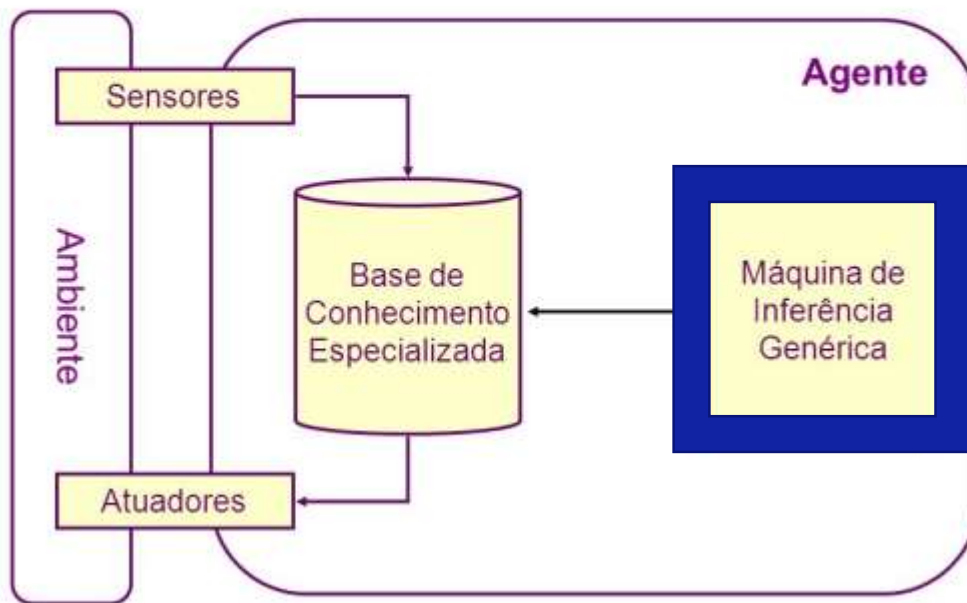


Base de Conhecimento ou Knowledge Base

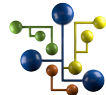


Agentes Baseados em Conhecimento

Agente Baseado em Conhecimento



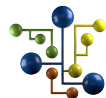
- TELL (informe)
- ASK (pergunte)



Agentes Baseados em Conhecimento

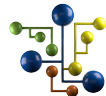
O componente central de um agente baseado em conhecimento é sua base de conhecimento.





Agentes Baseados em Conhecimento





Agentes Baseados em Conhecimento

- Podem lidar mais facilmente com ambientes parcialmente observáveis.
- O agente pode usar as suas percepções e conhecimento do mundo para inferir aspectos ainda desconhecidos do ambiente.
- São flexíveis e podem assumir novas tarefas na forma de objetivos explicitamente descritos.





Agentes Baseados em Conhecimento

função AGENTE-KB(*percepção*) **retorna** uma *ação*
persistente: *KB*, uma base de conhecimento

t, um contador, inicialmente igual a 0, indicando tempo

TELL(*KB*, CRIAR-SENTENÇA-DE-PERCEPÇÃO(*percepção*, *t*))

ação \leftarrow ASK(*KB* CRIAR-CONSULTA-DE-AÇÃO(*t*))

TELL(*KB*, CRIAR-SENTENÇA-DE-AÇÃO(*ação*, *t*))

t \leftarrow *t* + 1

retornar *ação*





Agentes Baseados em Conhecimento

função AGENTE-KB(*percepção*) **retorna** uma *ação*
persistente: *KB*, uma base de conhecimento

t, um contador, inicialmente igual a 0, indicando tempo

TELL(*KB*, CRIAR-SENTENÇA-DE-PERCEPÇÃO(*percepção*, *t*))

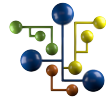
ação \leftarrow ASK(*KB* CRIAR-CONSULTA-DE-AÇÃO(*t*))

TELL(*KB*, CRIAR-SENTENÇA-DE-AÇÃO(*ação*, *t*))

t \leftarrow *t* + 1

retornar *ação*

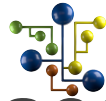




Agentes Baseados em Conhecimento

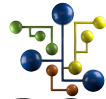
Níveis de descrição de um agente baseado em conhecimento



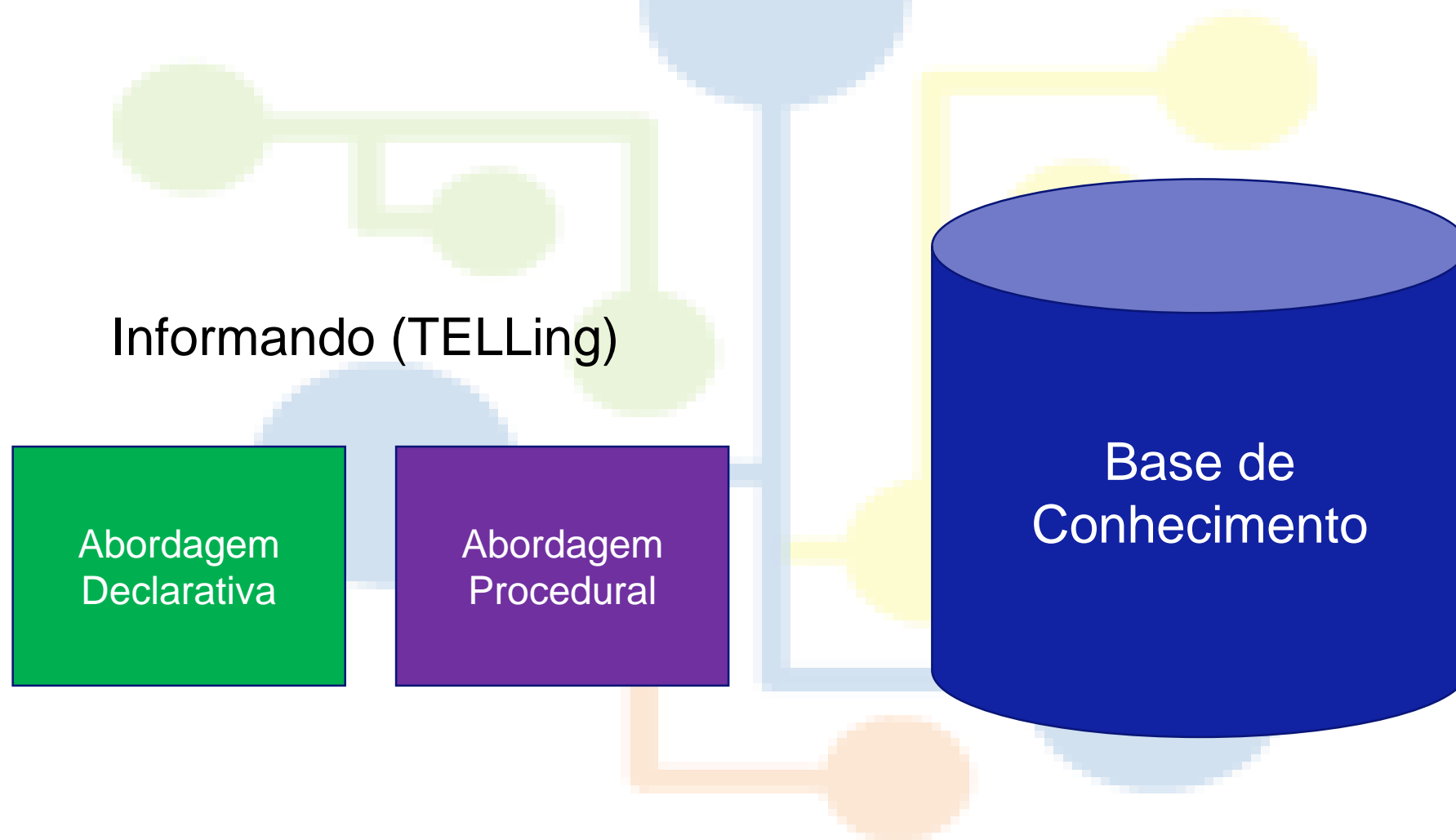


Agentes Baseados em Conhecimento





Agentes Baseados em Conhecimento

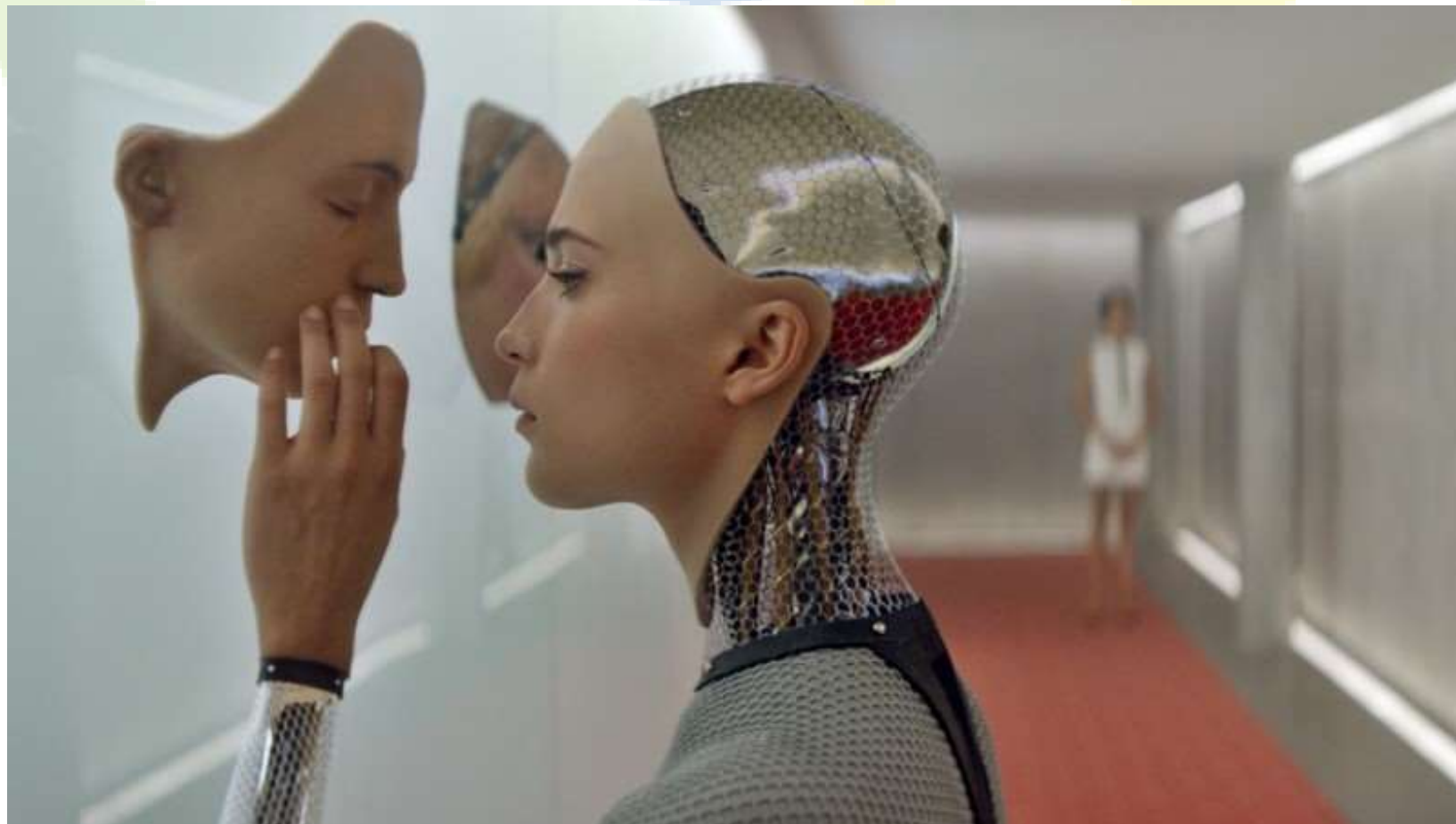




Data Science
Academy

Data Science Academy raphaelbsfontenelle@gmail.com 615c1fdde32fc361b30c9ec2

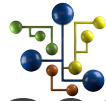
Agentes Baseados em Conhecimento



Data Science Academy

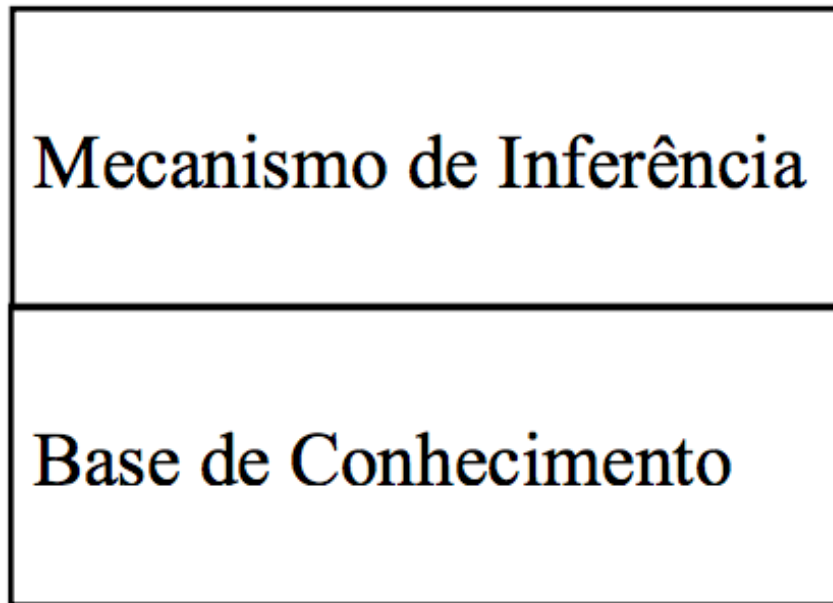


Data Science Academy



Agentes Baseados em Conhecimento

Agentes Baseados em Conhecimento



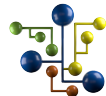
Mecanismo de Inferência

Base de Conhecimento

Algoritmos independentes do domínio (**objetivo de IA**)

Conhecimento específico do domínio (ontologias)





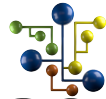
Agentes Baseados em Conhecimento

(1) Informa a base de conhecimento o que o agente esta percebendo do ambiente.

(2) Pergunta a base de conhecimento qual a próxima ação que deve ser executada. Um extensivo processo de raciocínio lógico é realizado sobre a base de conhecimento para que sejam decididas as ações que devem ser executadas.

(3) Realiza a ação escolhida e informa a base de conhecimento sobre a ação que está sendo realizada.

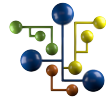




Agentes Baseados em Conhecimento

Como representamos a base de conhecimento de um agente?





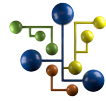
Agentes Baseados em Conhecimento

Como representamos a base de conhecimento de um agente?

- Lógica Proposicional
- Lógica de Primeira ordem
- Outras linguagens lógicas



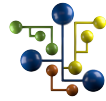
Lógica



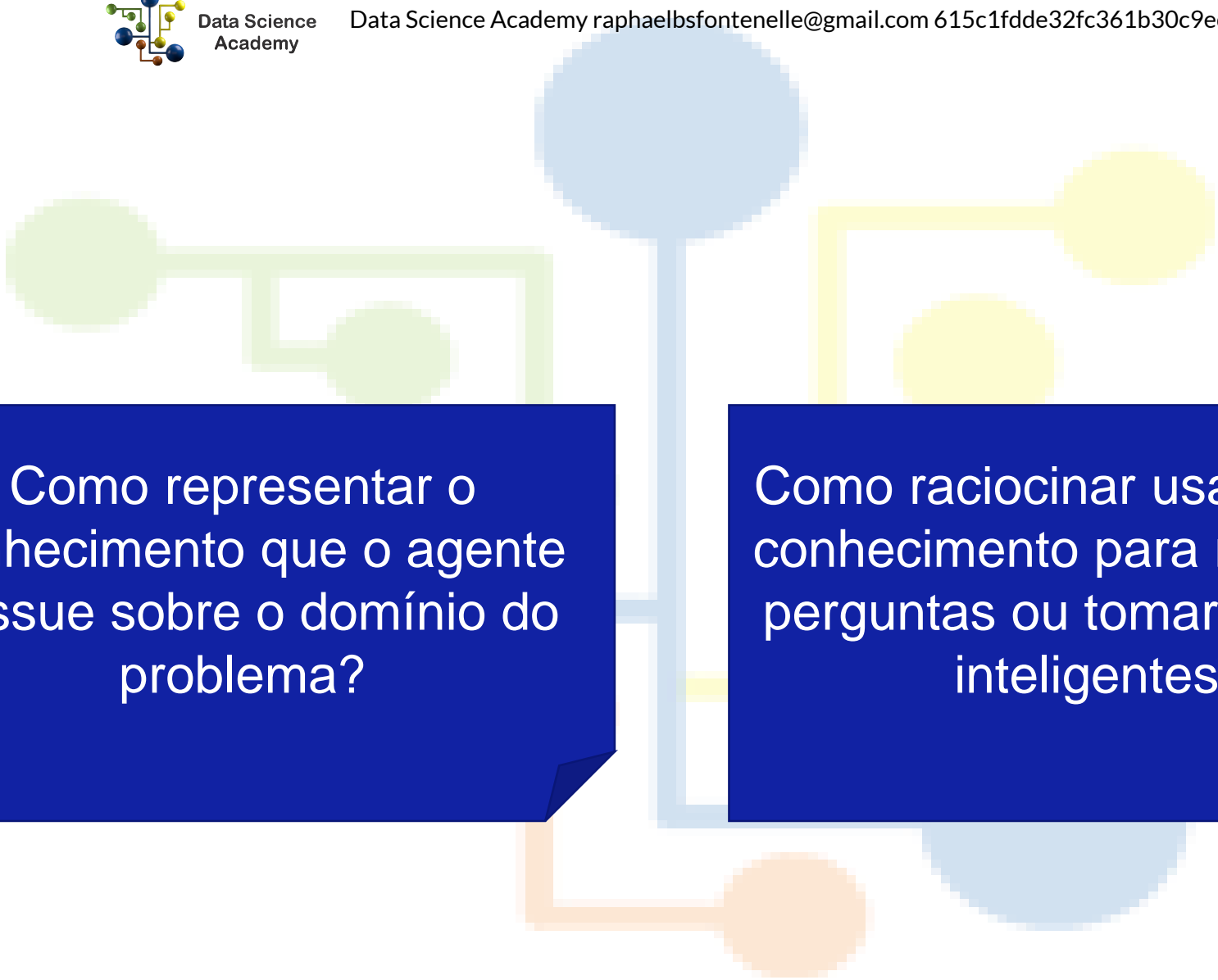
Lógica

A base de conhecimento de um agente é formada por um conjunto de sentenças expressas através de uma linguagem de representação de conhecimento.





Lógica

A decorative background diagram consisting of a central vertical blue line. To the left, a green line branches out to two green circles. To the right, a yellow line branches out to two yellow circles. At the bottom, an orange line branches out to two orange circles. The two blue boxes are positioned in the middle of the central blue line.

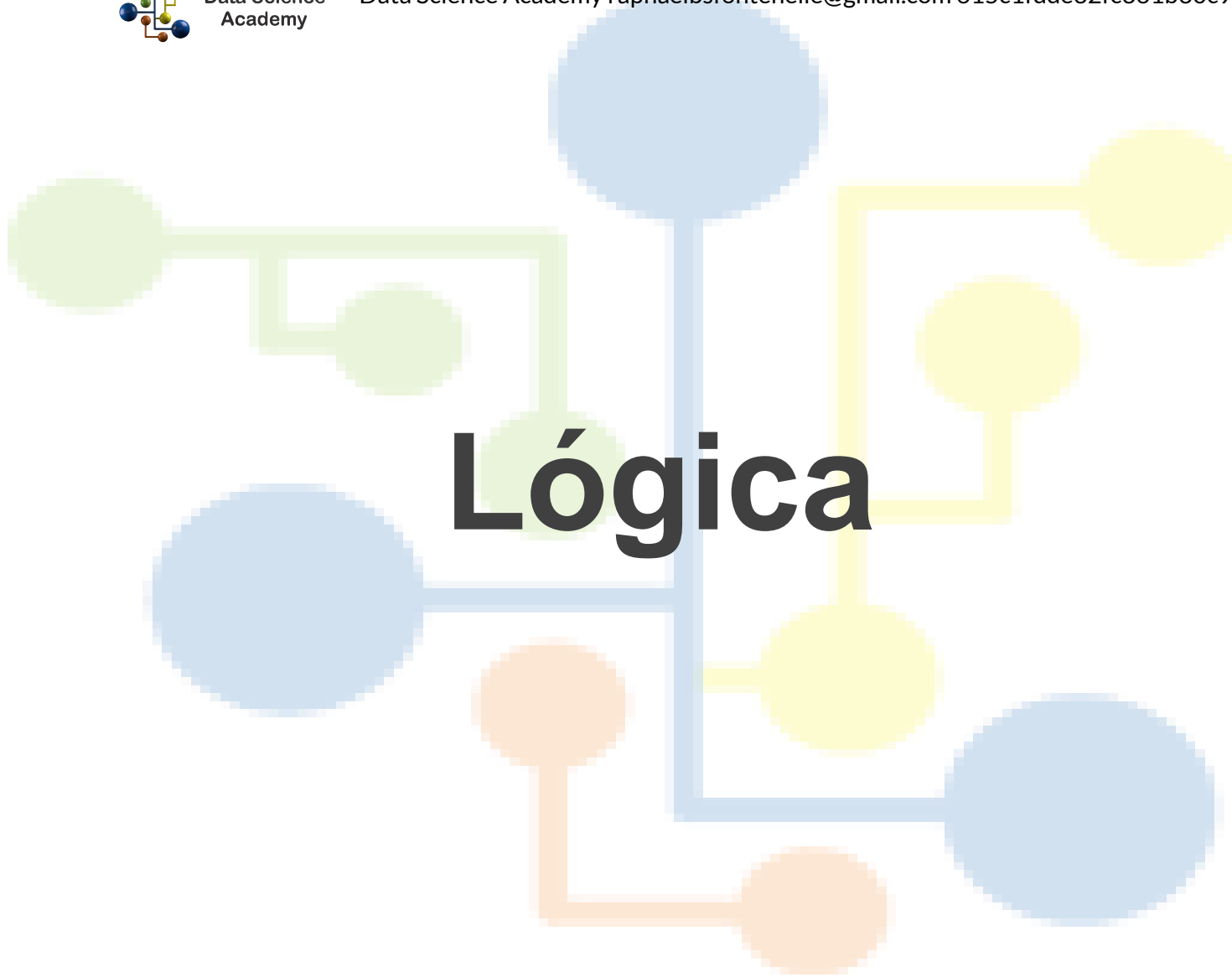
Como representar o conhecimento que o agente possui sobre o domínio do problema?

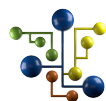
Como raciocinar usando esse conhecimento para responder perguntas ou tomar decisões inteligentes?



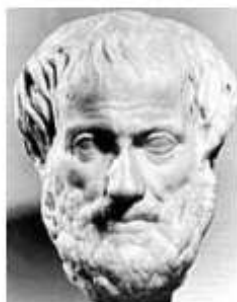


Lógica





Lógica



Aristóteles (384 a.C.–322 a.C.), filósofo grego. Produziu uma obra rica e multifacetada. Nela encontramos uma exhaustiva compilação dos conhecimentos do seu tempo, mas também, uma filosofia que ainda hoje influencia a nossa maneira de pensar.

Responsável por escrever os primeiros grandes trabalhos de lógica:

- Coleção de regras para raciocínio dedutivo que pode ser usado em qualquer área do conhecimento.



Gottfried Wilhelm Leibniz (1646–1716), filósofo e matemático alemão, provavelmente mais conhecido por ter inventado o cálculo integral e diferencial independentemente de Isaac Newton.

Propõe o uso de símbolos para mecanizar o processo de raciocínio dedutivo.

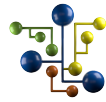


George Boole (1815–1864), matemático e filósofo inglês.



Augustus De Morgan (1806–1871), matemático inglês.

Propõem as bases da lógica simbólica moderna usando as idéias de Leibniz.

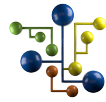


Lógica

Pesquisa continua sendo aplicada em áreas como:

- Projeto de circuito lógico
- Teoria de autômatos e computabilidade
- Teoria de bancos de dados relacionais
- Teoria de linguagens
- Teoria de sistemas distribuídos
- Inteligência artificial

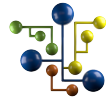




Lógica

LÓGICA é a ciência que estuda princípios e métodos de inferência, tendo o objetivo principal de determinar em que condições certas coisas se seguem (são consequência), ou não, de outras.





Lógica

Um Agente Baseado em Conhecimento deve:

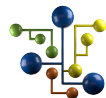
Representar
conhecimento sobre
o mundo em uma
linguagem formal
(BC)

Raciocinar sobre o
mundo usando
inferências na
linguagem (sobre a
BC)

Decidir que ação
tomar inferindo que
a ação selecionada é
a melhor

BC = Base de Conhecimento

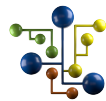




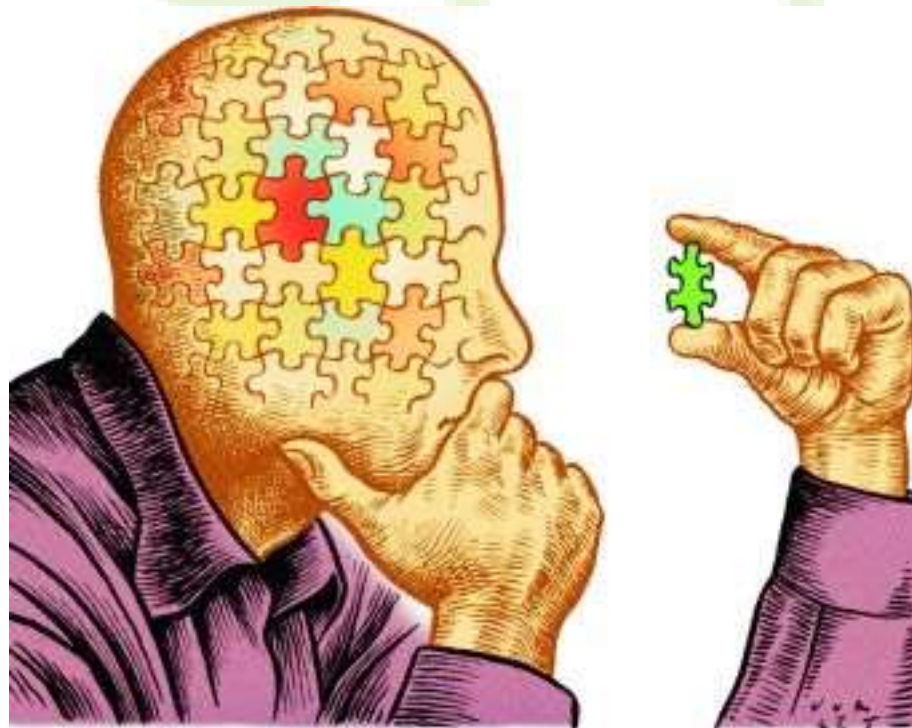
Lógica

Um Agente Baseado em Conhecimento deve:

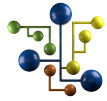




Lógica



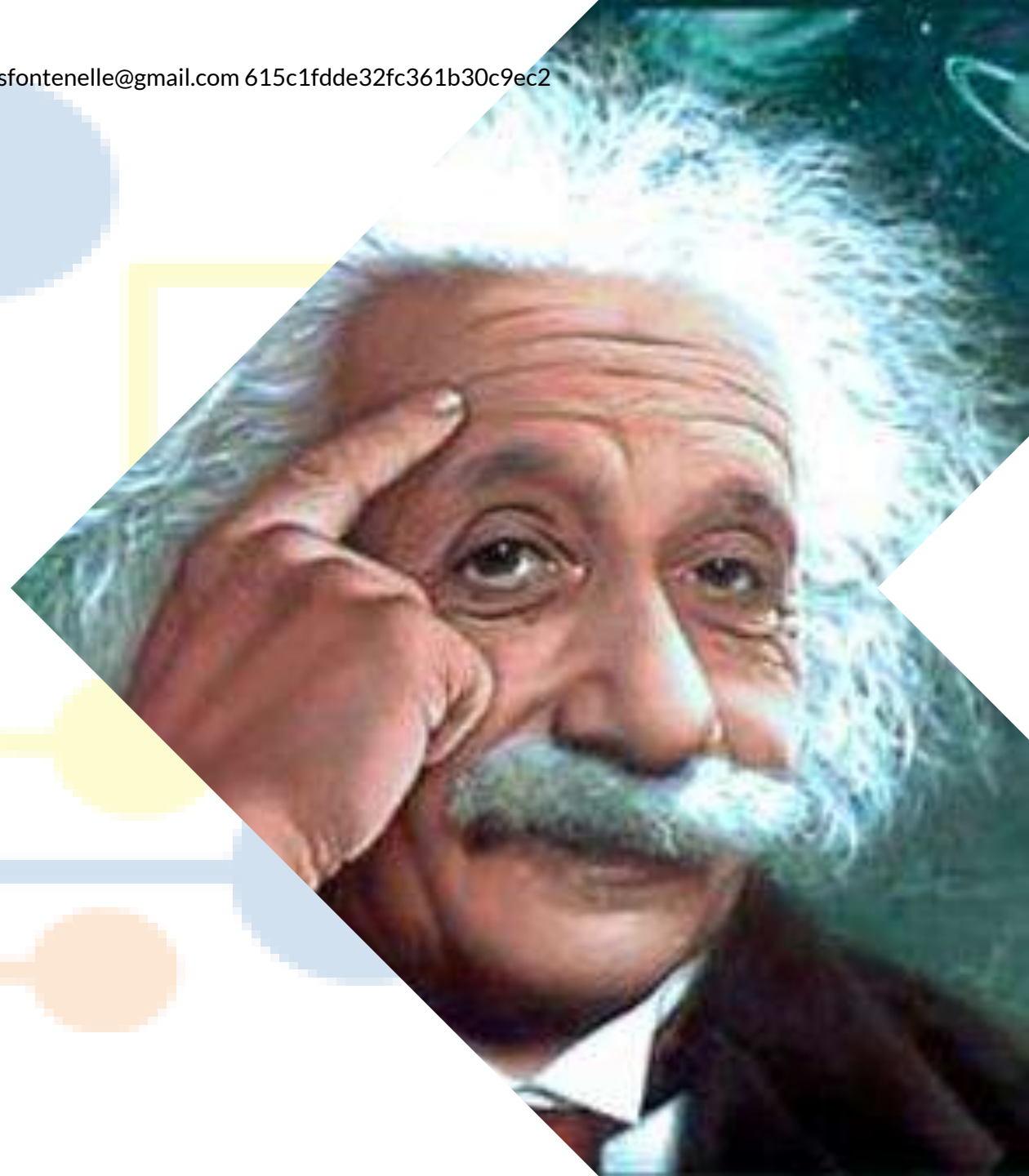
Podemos concluir que a melhor maneira de se obter comportamento inteligente será como um produto de um ***raciocínio correto*** sobre uma ***representação correta***

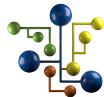


Lógica

- Lógica proposicional
- Lógica de predicados
- Lógica multivalorada
- Lógica modal
- Lógica temporal
- Lógica paraconsistente

Lógica de Primeira Ordem



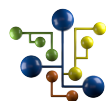


Lógica

Uma lógica inclui:

- **Sintaxe:** especifica os símbolos na linguagem e como eles podem ser combinados para formar sentenças.
- **Semântica:** define o “significado” de sentenças, isto é, define a verdade de uma sentença no mundo.
- **Procedimento de Inferência:** método mecânico para computar (derivar) novas sentenças (verdades) a partir de outras.





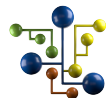
Lógica

- $x + 2 > y$ é uma sentença
- $x + y >$ não é uma sentença
- $x + 2 \geq y$ é verdade se o número $x + 2$ não for menor que o número y
- $x + 2 > y$ é verdade no mundo onde $x = 7, y = 1$
- $x + 2 > y$ é falso no mundo onde $x = 0, y = 6$

fatos

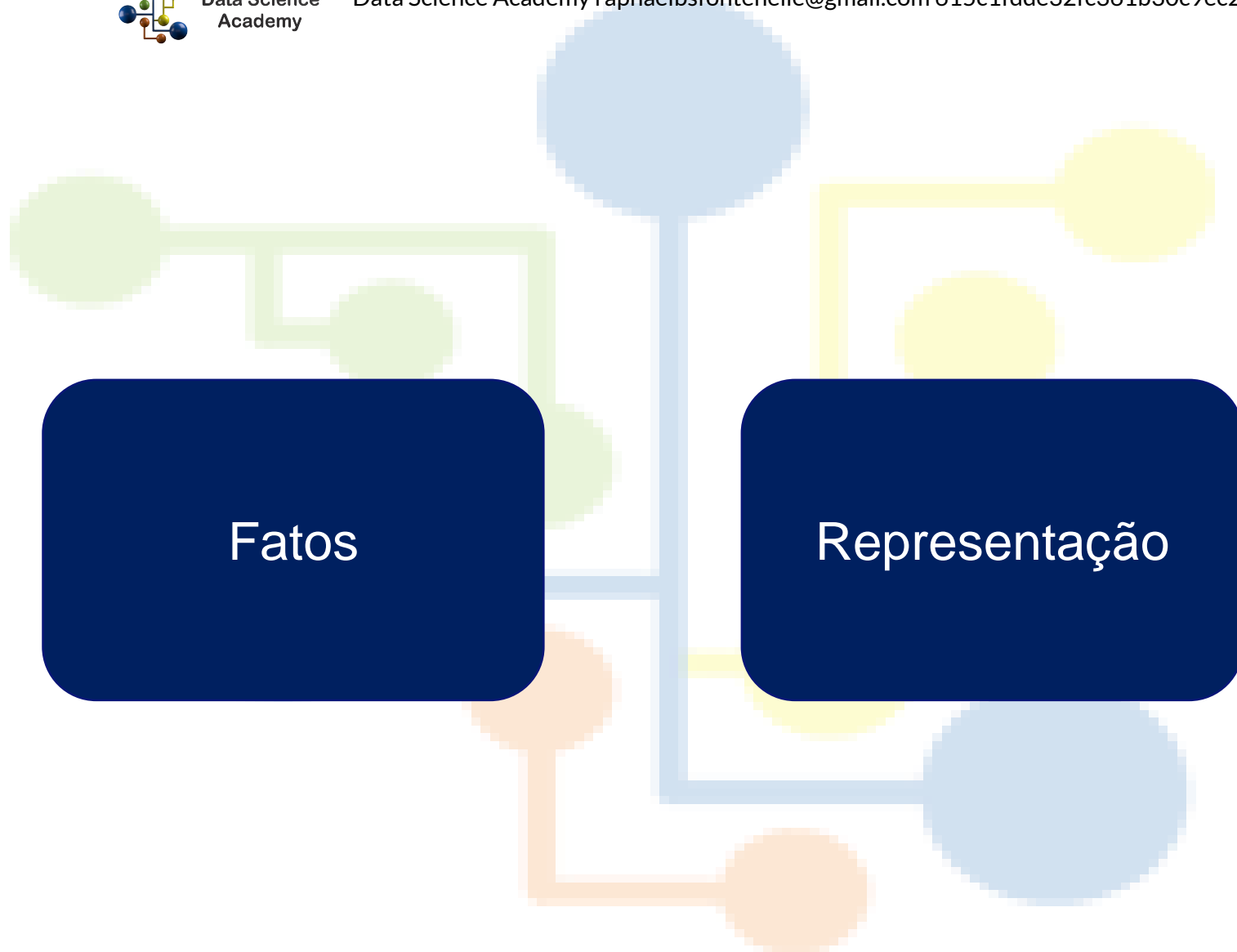


Lógica



Data Science
Academy

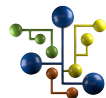
Data Science Academy raphaelbsfontenelle@gmail.com 615c1fdde32fc361b30c9ec2



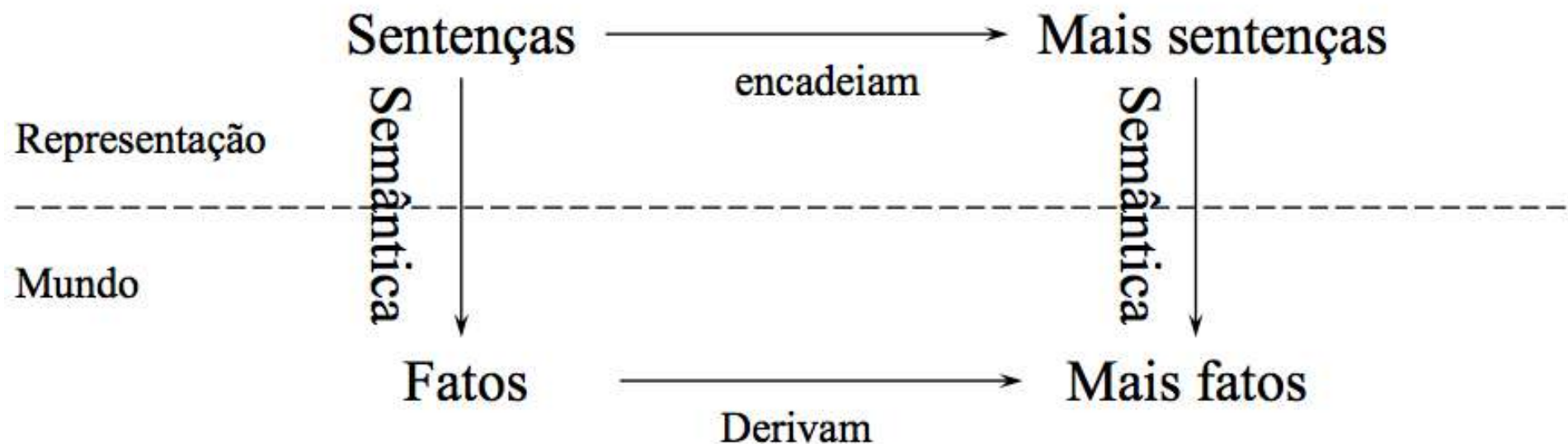
Data Science Academy



Data Science Academy



Lógica



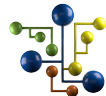
Mecanismo de Inferência infere todas as sentenças encadeadas pela BC
→ O mecanismo é completo!

Mecanismo de Inferência infere sentenças que correspondem aos fatos que derivam dos fatos iniciais sobre o mundo
→ O mecanismo é correto!

Melhor correto do que completo !!

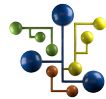


Lógica Proposicional



Lógica Proposicional

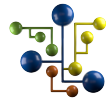




Lógica Proposicional

Álgebra das proposições, também conhecida por **lógica proposicional** é um tema muito cobrado especialmente em concursos públicos e também em alguns curso de graduação, como Ciência e Engenharia da Computação.



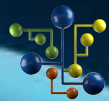


Lógica Proposicional

Proposição:

É uma sentença declarativa, seja ela expressa de forma afirmativa ou negativa, na qual podemos atribuir um valor lógico “V” (verdadeiro) ou “F”(falso). Uma proposição também pode ser expressa por símbolos.





Lógica Proposicional

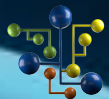
Brasília é a capital do Brasil – É uma sentença declarativa expressa de forma afirmativa. Podemos atribuir um valor lógico, como a sentença é verdadeira seu valor lógico é “V”.

A Argentina não é um país pertencente ao continente Africano – É uma sentença declarativa expressa na forma negativa. Podemos atribuir um valor lógico, como a sentença é verdadeira, seu valor lógico é “V”.

Todos os homens são mortais – É uma sentença declarativa expressa na forma afirmativa. Podemos atribuir um valor lógico, como a sentença é verdadeira, seu valor lógico é “V”.

12 é um número par positivo – É uma sentença declarativa expressa na forma afirmativa. Podemos atribuir um valor lógico, como a sentença é verdadeira, seu valor lógico é “V”



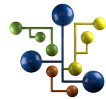


Lógica Proposicional

$7+5 = 10$ – É uma sentença declarativa expressa na forma afirmativa. Podemos atribuir um valor lógico, como a sentença é falsa, seu valor lógico é “F”.

$x - 2 = 5$ – Não é uma proposição, pois não sabemos o valor da variável “x”, ou melhor, não podemos atribuir um valor lógico “V” ou “F”. Porém para “torná-la” proposição bastaremos usar os chamados *quantificadores*.



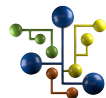


Lógica Proposicional

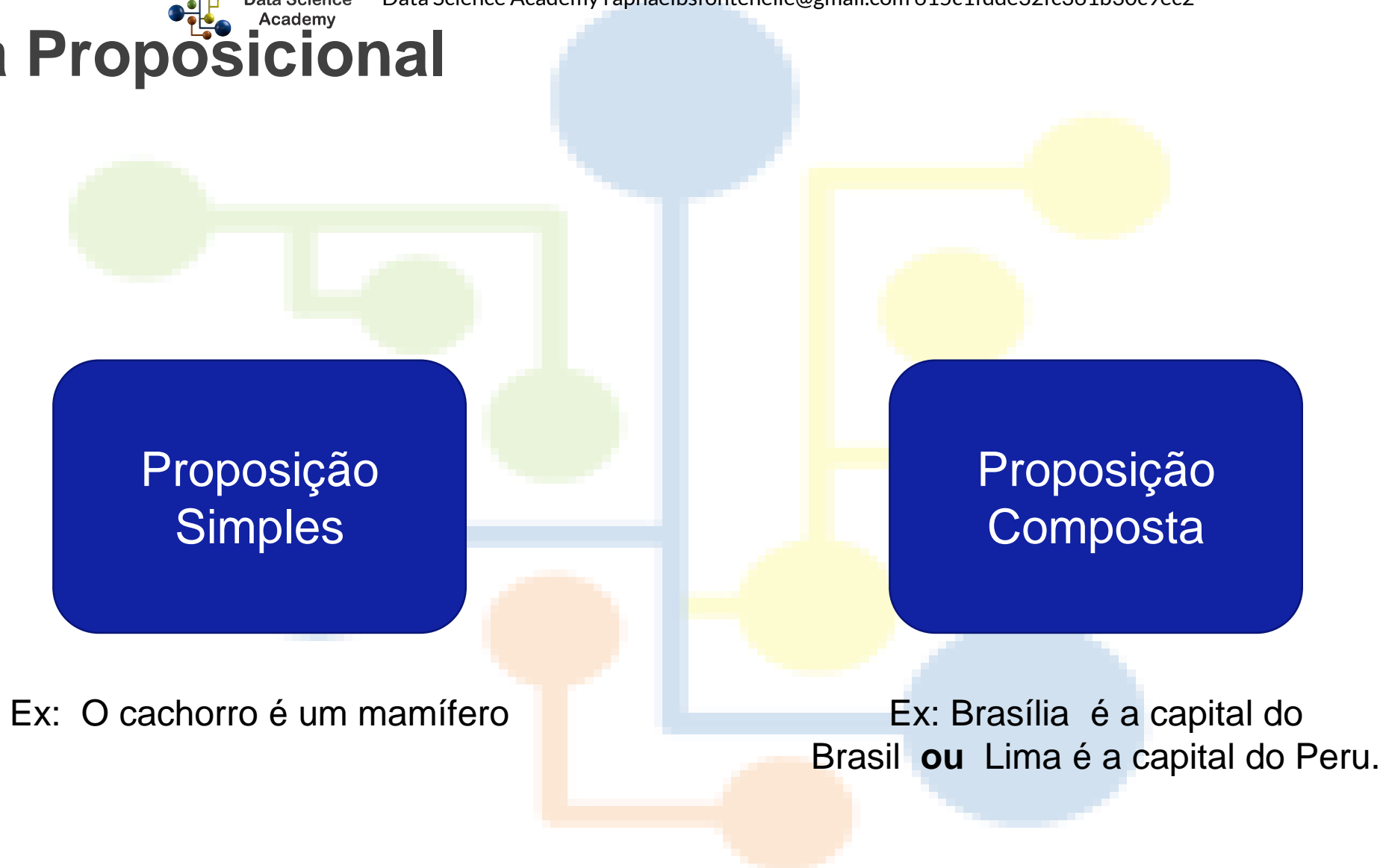
Exemplos de sentenças que **NÃO** são proposições:

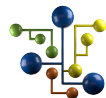
- Sentenças Interrogativas. Ex: “Onde você mora”?
- Sentenças Imperativas. Ex: “Venha aqui imediatamente.”
- Sentenças Exclamativas. Ex: “Opa!”
- Sentenças abertas
- Poemas





Lógica Proposicional





Lógica Proposicional

Sintaxe da Lógica Proposicional

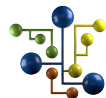
P, Q, R, W1,3 e Norte



Símbolos Proposicionais

Existem dois símbolos proposicionais com significados fixos:
Verdadeiro é a proposição sempre verdadeira e Falso é a
proposição sempre falsa.



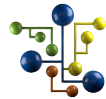


Lógica Proposicional

Conectivos Lógicos

- \neg negação
- \wedge conjunção (AND)
- \vee disjunção (OR)
- \rightarrow implicação (ou condicional)
- \leftrightarrow equivalência (bicondicional)





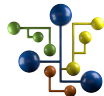
Lógica Proposicional

Semântica da Lógica Proposicional

Por exemplo, se as sentenças na base de conhecimento fazem uso dos símbolos proposicionais $P1,2$, $P2,2$ e $P3,1$, um modelo possível será:

$m1 = \{P1,2 = \text{falsa}, P2,2 = \text{falsa}, P3,1 = \text{verdadeira}\}$





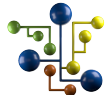
Lógica Proposicional

Semântica da Lógica Proposicional

- Verdadeiro é verdadeiro em todo modelo e Falso é falso em todo modelo.
- O valor-verdade de todos os outros símbolos proposicionais deve ser especificado diretamente no modelo. Por exemplo, no modelo $m1$ abaixo, $P1,2$ é falsa.

$m1 = \{P1,2 = \text{falsa}, P2,2 = \text{falsa}, P3,1 = \text{verdadeira}\}$



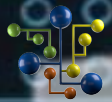


Lógica Proposicional

Para sentenças complexas, temos cinco regras, que valem para quaisquer subsentenças P e Q em qualquer modelo m (aqui “sse” significa “se e somente se”):

- $\neg P$ é verdadeiro sse P for falso em m .
- $P \wedge Q$ são verdadeiros sse P e Q forem verdadeiros em m .
- $P \vee Q$ é verdadeiro sse P ou Q for verdadeiro em m .
- $P \rightarrow Q$ é verdadeiro, a menos que P seja verdadeiro e Q seja falso em m .
- $P \leftrightarrow Q$ é verdadeiro sse P e Q forem ambos verdadeiros ou ambos falsos em m .





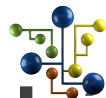
Data Science
Academy

Data Science Academy raphaelbsfontenelle@gmail.com 615c1fdde32fc361b30c9ec2

Conectivos Lógicos



Data Science Academy



Conectivos Lógicos

Operação	Conectivo	Estrutura Lógica	Exemplos
Negação	\neg	Não p	A bicicleta não é azul
Conjunção	\wedge	P e q	Thiago é médico e João é Engenheiro
Disjunção Inclusiva	\vee	P ou q	Thiago é médico ou João é Engenheiro
Disjunção Exclusiva	$\underline{\vee}$	Ou p ou q	Ou Thiago é Médico ou João é Engenheiro
Condicional	\rightarrow	Se p então q	Se Thiago é Médico então João é Engenheiro
Bicondicional	\leftrightarrow	P se e somente se q	Thiago é médico se e somente se João é Médico



Conjunção (AND)

P	Q	$P \wedge Q$
V	V	V
V	F	F
F	V	F
F	F	F

P: Irei ao cinema
Q: Irei ao clube

Conjunção: $p \wedge q$ (p e q)

Irei ao cinema **e** ao clube.



Disjunção Inclusiva (OR)

P	Q	$P \vee Q$
V	V	V
V	F	V
F	V	V
F	F	F

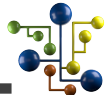
P: Darei a você uma camisa

Q: Darei a você um pijama

Disjunção: $p \vee q$ (p ou q)

Darei a você uma camisa **ou** darei a você um pijama.





Disjunção Exclusiva (XOR)

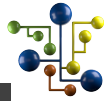
P	Q	$P \underline{\vee} Q$
V	V	F
V	F	V
F	V	V
F	F	F

P: Irei Jogar Basquete
Q: Irei à casa de José

Disjunção Exclusiva:
 $p \underline{\vee} q$ (ou p ou q)

Ou irei jogar basquete **ou** irei à casa de João





Condicional

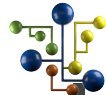
P	Q	$P \rightarrow Q$
V	V	V
V	F	F
F	V	V
F	F	V

P: Nasci em Salvador
Q: Sou Baiano

Condicional: $p \rightarrow q$
(Se... então)

Se nasci em Salvador, **então** sou Baiano.





Bicondicional

P	Q	$P \leftrightarrow Q$
V	V	V
V	F	F
F	V	F
F	F	V

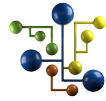
P: 4 é maior que 2

Q: 2 é menor que 4

Bicondicional: $p \leftrightarrow q$
(p se e somente se q)

4 é maior que 2 **se e somente se** 2 for menor que 4 .





Negação

P: O Brasil é um País pertencente a América do Sul.

¬P: O Brasil **não** é um País pertencente a América do Sul.

Q: X é Par

¬Q: X **não** é par





Conectivos Lógicos

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
falso	falso	verdadeiro	falso	falso	verdadeiro	verdadeiro
falso	verdadeiro	verdadeiro	falso	verdadeiro	verdadeiro	falso
verdadeiro	falso	falso	falso	verdadeiro	falso	falso
verdadeiro	verdadeiro	falso	verdadeiro	verdadeiro	verdadeiro	verdadeiro



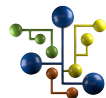


Lógica Proposicional

Base de
Conhecimento

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$
<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>
<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>verdadeiro</i>
.
.
.
<i>falso</i>	<i>verdadeiro</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>
<i>falso</i>	<i>verdadeiro</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>verdadeiro</i>
<i>falso</i>	<i>verdadeiro</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>verdadeiro</i>	<i>falso</i>
<i>falso</i>	<i>verdadeiro</i>	<i>falso</i>	<i>falso</i>	<i>falso</i>	<i>verdadeiro</i>	<i>verdadeiro</i>
<i>falso</i>	<i>verdadeiro</i>	<i>falso</i>	<i>falso</i>	<i>verdadeiro</i>	<i>falso</i>	<i>falso</i>
.
.
.
<i>verdadeiro</i>	<i>verdadeiro</i>	<i>verdadeiro</i>	<i>verdadeiro</i>	<i>verdadeiro</i>	<i>verdadeiro</i>	<i>verdadeiro</i>





Lógica Proposicional

Algoritmo de
enumeração de tabela-
verdade para decidir a
consequência lógica
proposicional

função CONSEQUÊNCIA-LÓGICA-TV?(BC, α) **devolve** verdadeiro ou falso
entradas: BC , a base de conhecimento, uma sentença em lógica proposicional
 α , a consulta, uma sentença em lógica proposicional

$\text{simbolos} \leftarrow$ uma lista dos símbolos proposicionais em BC e α
devolver VERIFICAR-TODOS-TV($BC, \alpha, \text{simbolos}, \{ \}$)

função VERIFICAR-TODOS-TV($BC, \alpha, \text{simbolos}, \text{modelo}$) **devolve** verdadeiro ou falso
se VAZIO?(simbolos) **então**
 se VERDADEIRO-LP?(BC, modelo) **então** **devolver** VERDADEIRO-LP?(α, modelo)
 senão **devolver** verdadeiro // quando BC for falso, sempre retornar verdadeiro
senão faça
 $P \leftarrow$ PRIMEIRO(simbolos)
 $\text{restante} \leftarrow$ RESTO(simbolos)
 devolver VERIFICAR-TODOS-TV($BC, \alpha, \text{restante}, \text{modelo} \cup \{P = \text{verdadeiro}\}$)
 e
 VERIFICAR-TODOS-TV($BC, \alpha, \text{restante}, \text{modelo} \cup \{P = \text{falso}\}$)



Planejamento Clássico



Planejamento Clássico

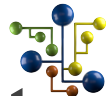
Agentes Baseados em Busca

- Busca cega;
- Busca heurística;
- Busca local;

Agentes Lógicos

- Lógica proposicional;
- Lógica de primeira ordem;
- Prolog;



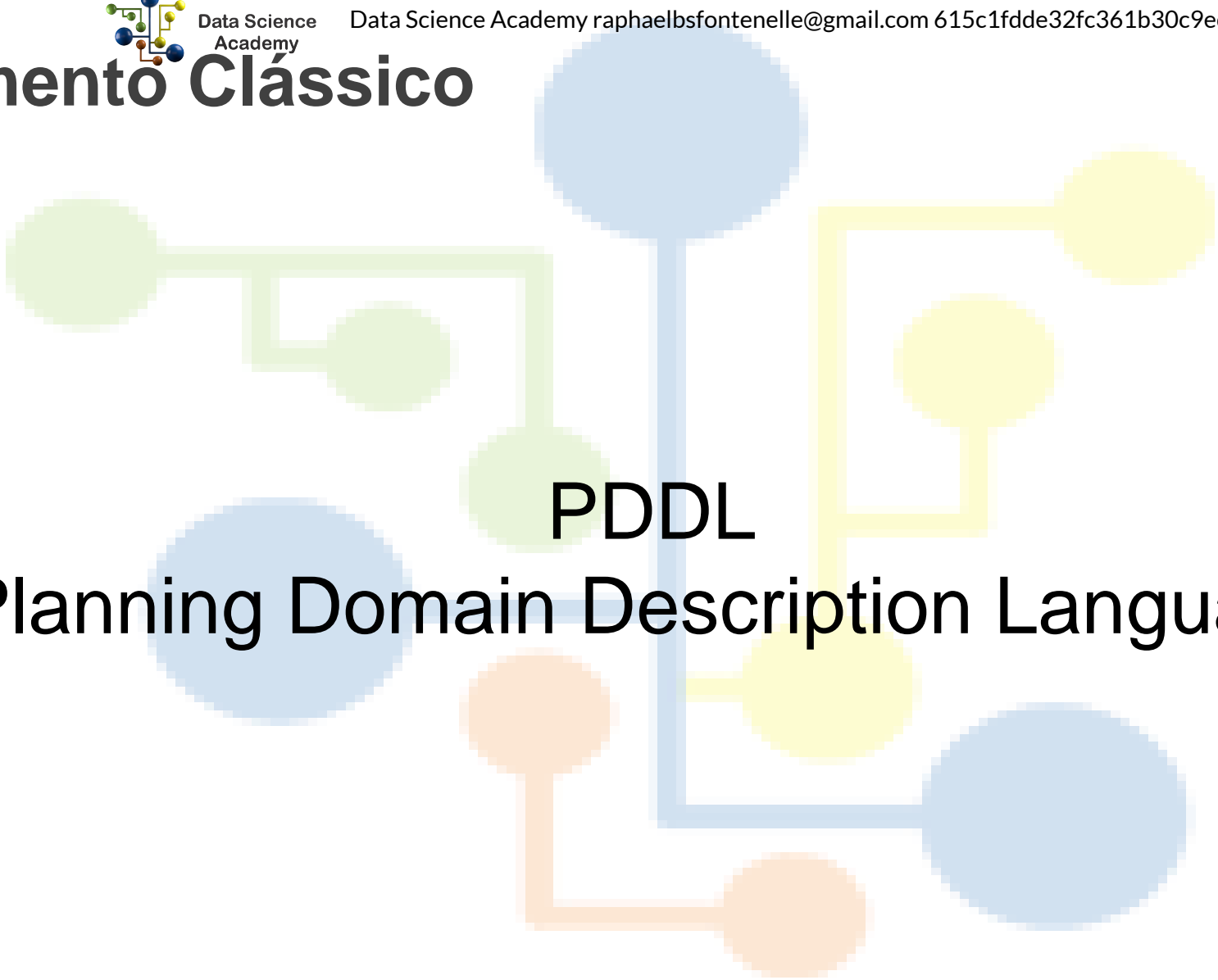


Planejamento Clássico



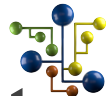


Planejamento Clássico

A large, faint background diagram of a network graph. It consists of several circular nodes connected by lines. The nodes are colored in light blue, light green, light yellow, and light orange. The connections form a complex web, with some nodes having multiple connections and others being isolated or part of small clusters.

PDDL Planning Domain Description Language





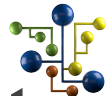
Planejamento Clássico

PDDL Planning Domain Description Language

Componentes da linguagem PDDL:

- Objetos: objetos que compõem o problema de planejamento.
- Predicados: propriedades dos objetos – podem ser verdadeiros ou falsos.
- Estado Inicial: estado do mundo onde o processo de planejamento se inicia.
- Objetivos: predicados que devem ser verdade para concluir o processo de planejamento.
- Ações/Operadores: ações que podem ser executadas e modificam o estado do mundo.





Planejamento Clássico

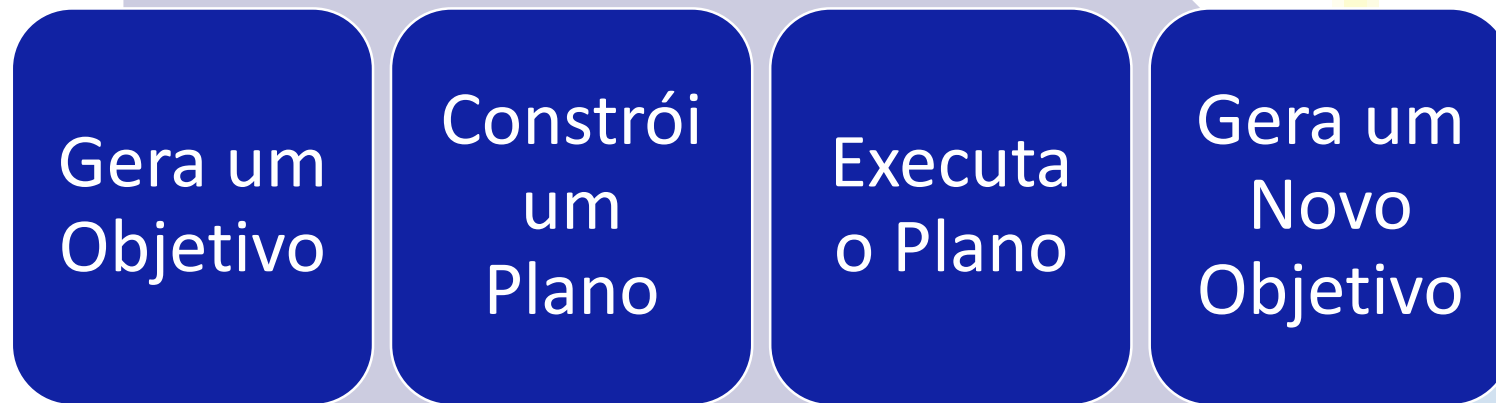
Planejamento consiste na tarefa de apresentar uma sequência de ações para alcançar um determinado objetivo

Ir(Mercado), Comprar(Biscoito), Ir(Farmácia), Comprar(Remédio), Ir(Casa)

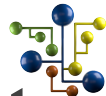




Planejamento Clássico



Funcionamento de um
Agente Planejador



Planejamento Clássico

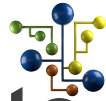
Algoritmos de busca tendem a tomar ações irrelevantes.

- Grande fator de ramificação.
- Pouco conhecimento para guiar a busca.

Planejador não considera ações irrelevantes.

- Faz conexões diretas entre estados (sentenças) e ações (pré-condições + efeitos)
- Objetivo: Ter(Leite).
 - Ação: Comprar(Leite) => Ter(Leite)





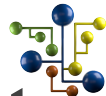
Planejamento Clássico

Linguagem STRIPS

– Inicial: Em(Casa) – Final: Em(Casa) \wedge Ter(Leite) \wedge Ter(Bananas) \wedge Ter(Furadeira)

Hipótese do mundo fechado: qualquer condição não mencionada em um estado é considerada negativa
Exemplo: \neg Ter(Leite) \wedge \neg Ter(Bananas) \wedge \neg Ter(Furadeira)





Planejamento Clássico

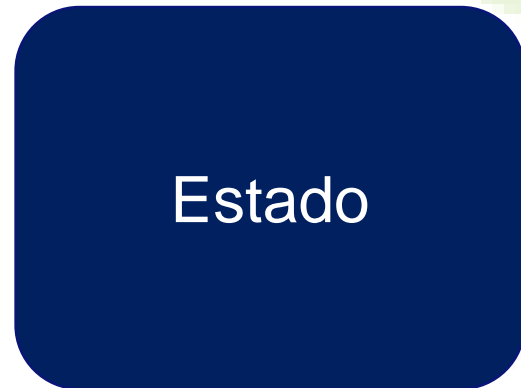
Mostraremos agora como PDDL descreve os quatro itens que precisamos para definir um problema de busca:

- O estado inicial
- As ações que estão disponíveis em um estado
- O resultado da aplicação de uma ação e
- O teste de objetivo





Planejamento Clássico



Proposições válidas:

Pobre \wedge Desconhecido

$\text{Em}(\text{Caminhão1}, \text{Recife}) \wedge \text{Em}(\text{Caminhão2}, \text{Fortaleza})$

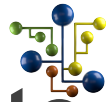
Proposições Não Permitidas:

$\text{Em}(x, y)$

$\neg \text{Pobre}$

$\text{Em}(\text{Pai}(\text{Fred}), \text{Sydney})$

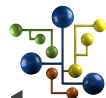




Planejamento Clássico



AÇÕES(s) e RESULTADO(s, a) necessárias para fazer uma busca de resolução de problema



Planejamento Clássico

Ação

Ação (*Voar*(p , de , $para$),

PRECOND: $Em(p, de) \wedge Avião(p) \wedge Aeroporto(de) \wedge Aeroporto(para)$

EFEITO: $\neg Em(p, de) \wedge Em(p, para)$).



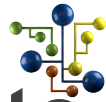


Planejamento Clássico

Ação

$Ação(Voar(P_1, SFO, JFK),$
 $PRECOND: Em(P_1, SFO) \wedge Avião(P_1) \wedge Aeroporto(SFO) \wedge Aeroporto(JFK)$
 $EFEITO: \neg Em(P_1, SFO) \wedge Em(P_1, JFK).$





Planejamento Clássico



$$(a \in \text{AÇÕES}(s)) \Leftrightarrow s \models \text{PRECOND}(a),$$

Uma ação a pode ser executada no estado s se a precondição de a for uma consequência lógica de s .

$$\forall p, de, para \ (\text{Voar}(p, de, para) \in \text{AÇÕES}(s)) \Leftrightarrow s \models (\text{Em}(p, de) \wedge \text{Avião}(p) \wedge \text{Aeroporto}(de) \wedge \text{Aeroporto}(para)).$$

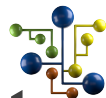




Planejamento Clássico

Em PDDL, os tempos e os estados estão implícitos nos esquemas de ação:
a precondição sempre se refere ao tempo t e o efeito ao tempo $t + 1$.





Data Science
Academy

Data Science Academy raphaelbsfontenelle@gmail.com 615c1fdde32fc361b30c9ec2

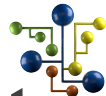
Planejamento Clássico



Data Science Academy



Data Science Academy



Planejamento Clássico

$\text{Início}(\text{Pneu}(\text{Furado}) \wedge \text{Pneu}(\text{Sobressalente}) \wedge \text{Em}(\text{furado}, \text{Eixo}) \wedge \text{Em}(\text{sobressalente}, \text{Porta-malas}))$

$\text{Objetivo}(\text{Em}(\text{Sobressalente}, \text{Eixo}))$

$\text{Ação}(\text{Remover}(\text{obj}, \text{loc}),$

$\text{PRECOND: } \text{Em}(\text{obj}, \text{loc})$

$\text{EFEITO: } \neg \text{Em}(\text{obj}, \text{loc}) \wedge \text{Em}(\text{obj}, \text{Chão}))$

$\text{Ação}(\text{Colocar}(t, \text{Eixo}),$

$\text{PRECOND: } \text{Pneu}(t) \wedge \text{Em}(t, \text{Chão}) \wedge \neg \text{Em}(\text{Furado}, \text{Eixo})$

$\text{EFEITO: } \neg \text{Em}(t, \text{Chão}) \wedge \text{Em}(t, \text{Eixo}))$

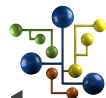
$\text{Ação}(\text{DeixarDuranteNoite},$

PRECOND:

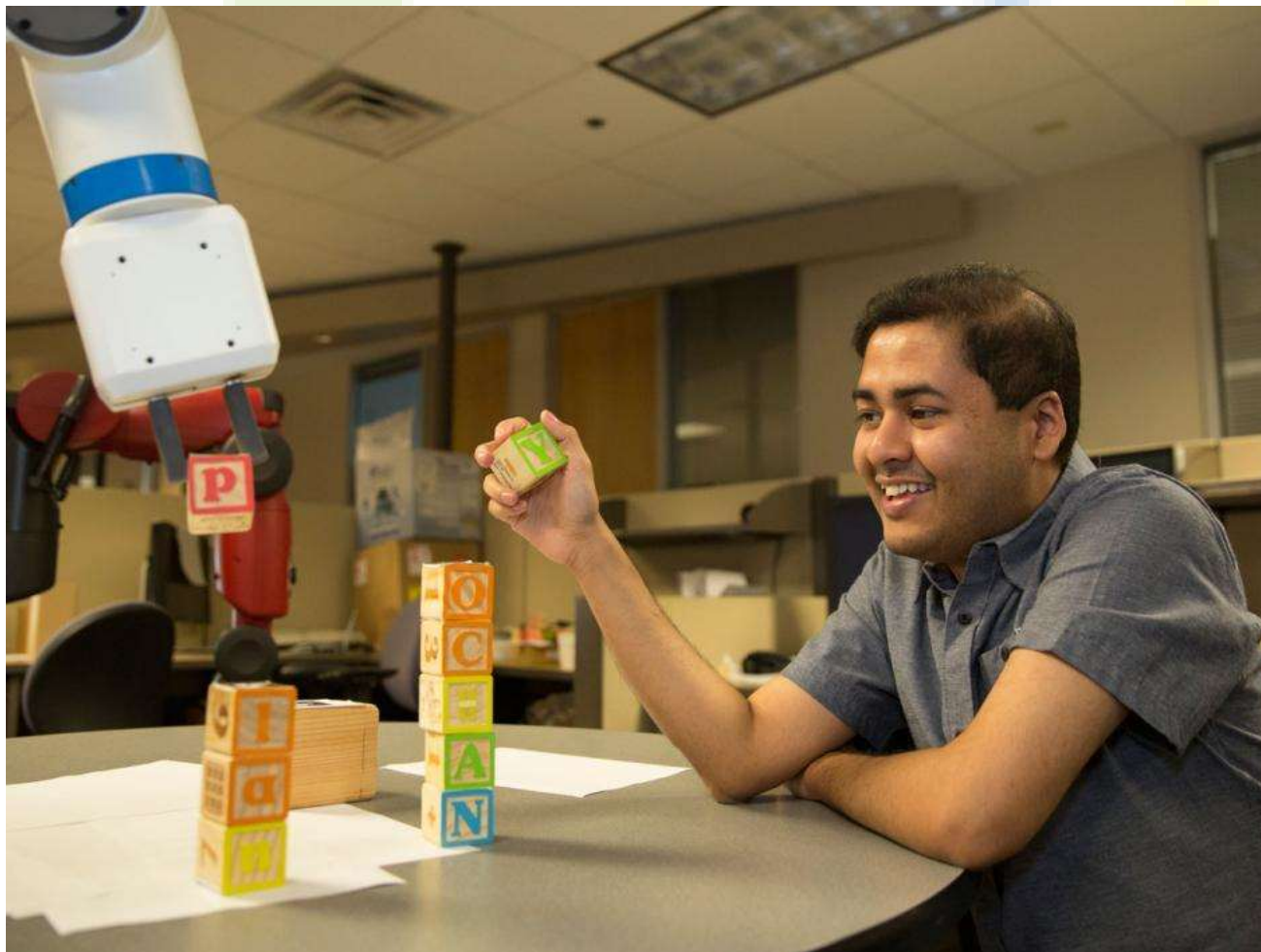
$\text{EFEITO: } \neg \text{Em}(\text{Sobressalente}, \text{Chão}) \wedge \neg \text{Em}(\text{Sobressalente}, \text{Eixo}) \wedge \neg \text{Em}(\text{Sobressalente}, \text{Porta-mala})$

$\wedge \neg \text{Em}(\text{Furado}, \text{Chão}) \wedge \neg \text{Em}(\text{Furado}, \text{Eixo}) \wedge \neg \text{Em}(\text{Furado}, \text{Porta-malas}))$





Planejamento Clássico



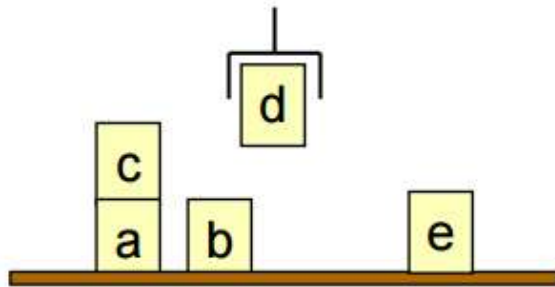
Mundo dos Blocos



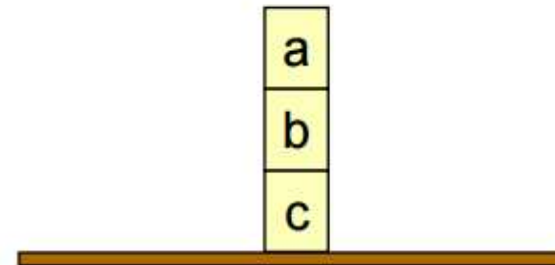


Planejamento Clássico

- Mesa infinitamente larga, número finito de blocos
- Ignora a posição em que um bloco está sobre a mesa
- Um bloco pode estar sobre a mesa ou sobre um outro bloco
- Os blocos devem ser movidos de uma configuração para outra

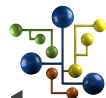


Estado Inicial



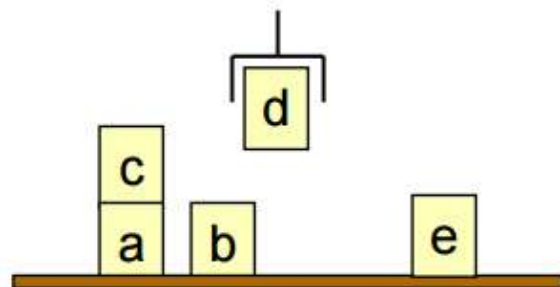
Estado Objetivo



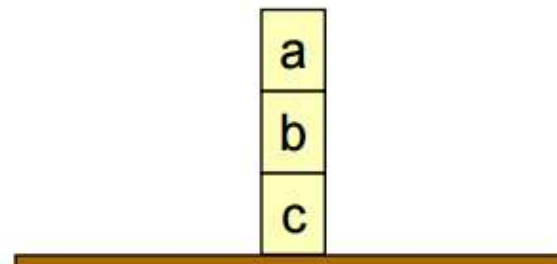


Planejamento Clássico

$\text{Início}(\text{Sobre}(A, \text{Mesa}) \wedge \text{Sobre}(B, \text{Mesa}) \wedge \text{Sobre}(C, A)$
 $\wedge \text{Bloco}(A) \wedge \text{Bloco}(B) \wedge \text{Bloco}(C) \wedge \text{Livre}(B) \wedge \text{Livre}(C))$
 $\text{Objetivo}(\text{Sobre}(A, B) \wedge \text{Sobre}(B, C))$
 $\text{Ação}(\text{Mover}(b, x, y),$
PRECOND: $\text{Sobre}(b, x) \wedge \text{Livre}(b) \wedge \text{Livre}(y) \wedge \text{Bloco}(b) \wedge \text{Bloco}(y) \wedge (b \neq x) \wedge (b \neq y)$
 $\wedge (x \neq y),$
EFEITO: $\text{Sobre}(b, y) \wedge \text{Livre}(x) \wedge \neg \text{Sobre}(b, x) \wedge \neg \text{Livre}(y))$
 $\text{Ação}(\text{MoverParaMesa}(b, x),$
PRECOND: $\text{Sobre}(b, x) \wedge \text{Livre}(b) \wedge \text{Bloco}(b) \wedge (b \neq x),$
EFEITO: $\text{Sobre}(b, \text{Mesa}) \wedge \text{Livre}(x) \wedge \neg \text{Sobre}(b, x))$

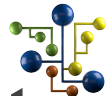


Estado Inicial



Estado Objetivo





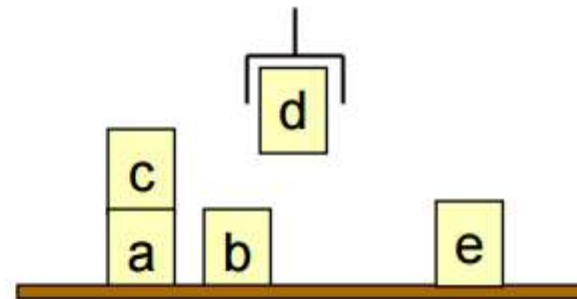
Planejamento Clássico

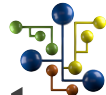
Símbolos constantes:

- Os blocos: a, b, c, d, e

Predicados:

- | | |
|----------------|--|
| – $ontable(x)$ | - bloco x está sobre a mesa |
| – $on(x,y)$ | - bloco x está sobre o bloco y |
| – $clear(x)$ | - bloco x não tem nada sobre ele |
| – $holding(x)$ | - a garra do robô está segurando o bloco x |
| – $handempty$ | - a garra do robô não está segurando nada |





Planejamento Clássico

Operadores

unstack(x,y)

Precond: $\text{on}(x,y)$, $\text{clear}(x)$, handempty

Effects: $\sim\text{on}(x,y)$, $\sim\text{clear}(x)$, $\sim\text{handempty}$,
 $\text{holding}(x)$, $\text{clear}(y)$

stack(x,y)

Precond: $\text{holding}(x)$, $\text{clear}(y)$

Effects: $\sim\text{holding}(x)$, $\sim\text{clear}(y)$,
 $\text{on}(x,y)$, $\text{clear}(x)$, handempty

pickup(x)

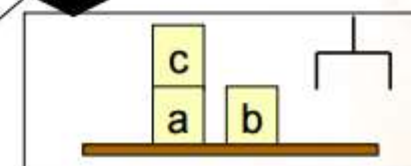
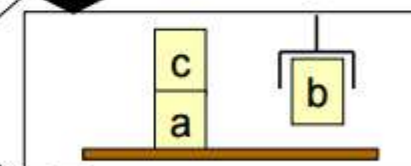
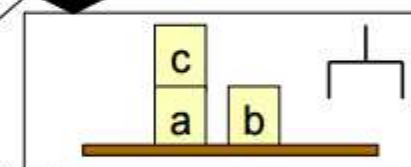
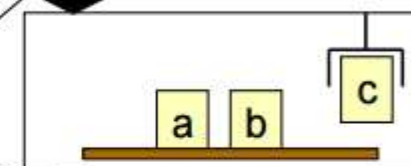
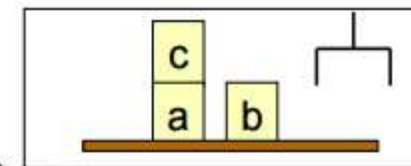
Precond: $\text{ontable}(x)$, $\text{clear}(x)$, handempty

Effects: $\sim\text{ontable}(x)$, $\sim\text{clear}(x)$,
 $\sim\text{handempty}$, $\text{holding}(x)$

putdown(x)

Precond: $\text{holding}(x)$

Effects: $\sim\text{holding}(x)$, $\text{ontable}(x)$,
 $\text{clear}(x)$, handempty



Algoritmos de Planejamento

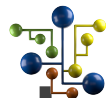


Algoritmos de Planejamento

The diagram illustrates the classification of planning algorithms. It features a central vertical blue line. To the left, a green line branches off, leading to two green circles. To the right, a yellow line branches off, leading to three yellow circles. At the bottom, an orange line branches off, leading to one orange circle. Two large blue rounded rectangles are positioned in the center, one on the left and one on the right, containing text about progressive and regressive algorithms. The background is white with faint, larger versions of the colored circles and lines.

Progressivo
estado inicial -> objetivo

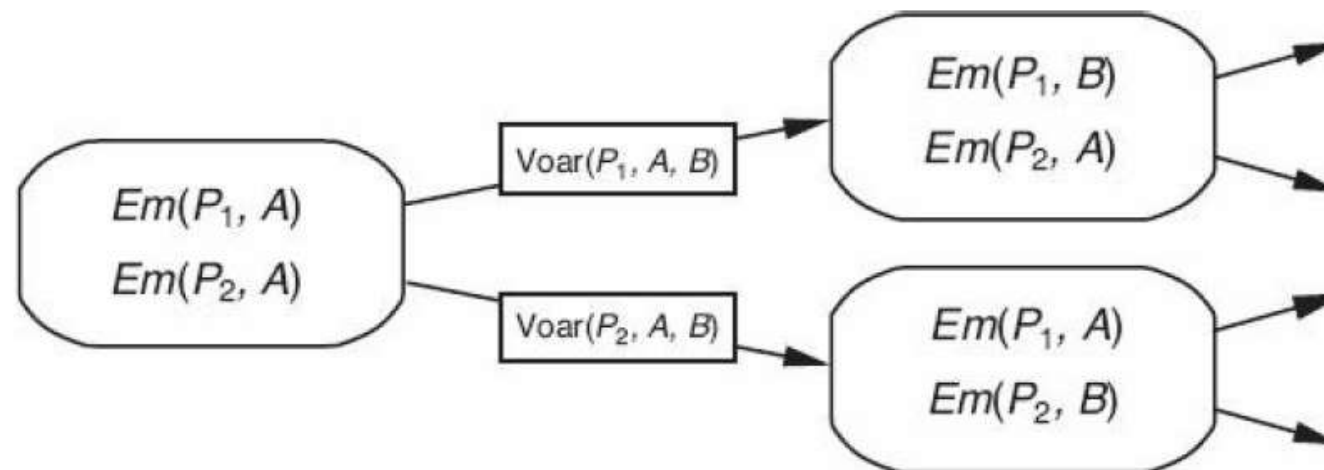
Regressivo
objetivo -> estado inicial



Algoritmos de Planejamento

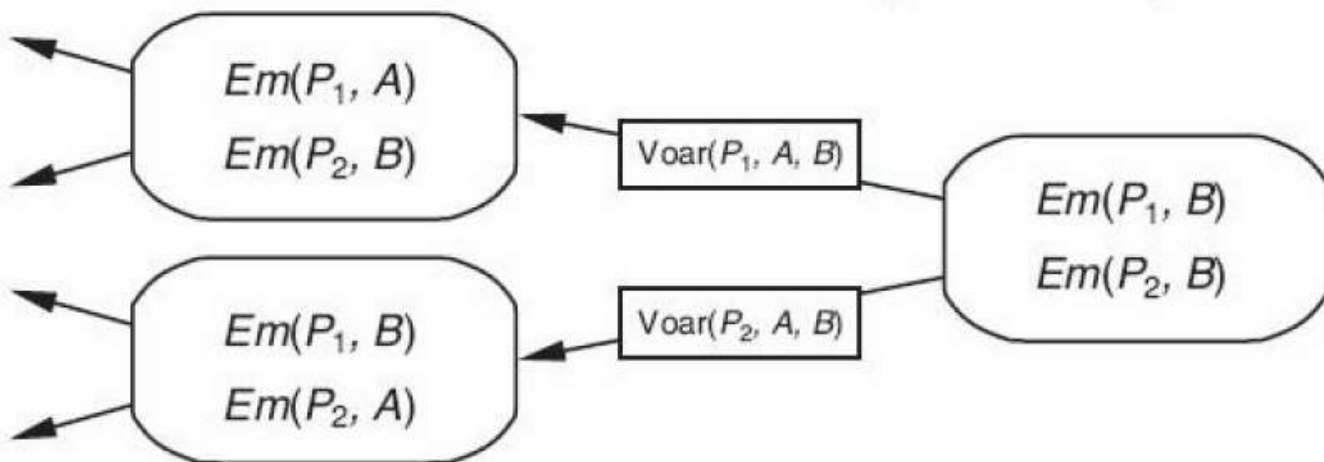
Progressivo

(a)



Regressivo

(b)





Algoritmos de Planejamento

Busca em espaço de estados para a frente (progressão)

A busca para a frente é propensa a explorar ações irrelevantes

Forward-search(O, s_0, g)

$s \leftarrow s_0$

$\pi \leftarrow$ the empty plan

loop

if s satisfies g then return π

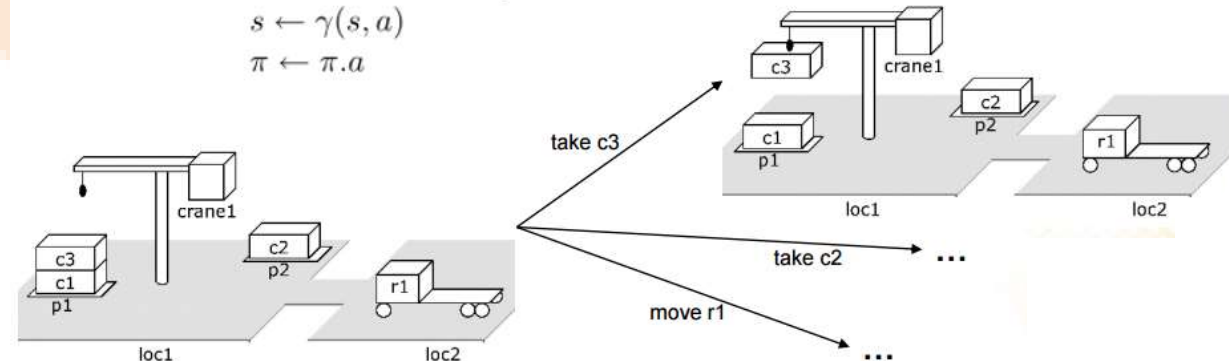
$E \leftarrow \{a \mid a \text{ is a ground instance an operator in } O,$
and $\text{precond}(a)$ is true in $s\}$

if $E = \emptyset$ then return failure

nondeterministically choose an action $a \in E$

$s \leftarrow \gamma(s, a)$

$\pi \leftarrow \pi.a$





Algoritmos de Planejamento

Busca em espaço de estados para a frente (progressão)

Considere a tarefa de comprar uma cópia de um livro de uma livraria on-line.

Suponha que exista um esquema de ação:

Comprar(isbn)

com efeito:

Possui(isbn)

Forward-search(O, s_0, g)

$s \leftarrow s_0$

$\pi \leftarrow$ the empty plan

loop

if s satisfies g then return π

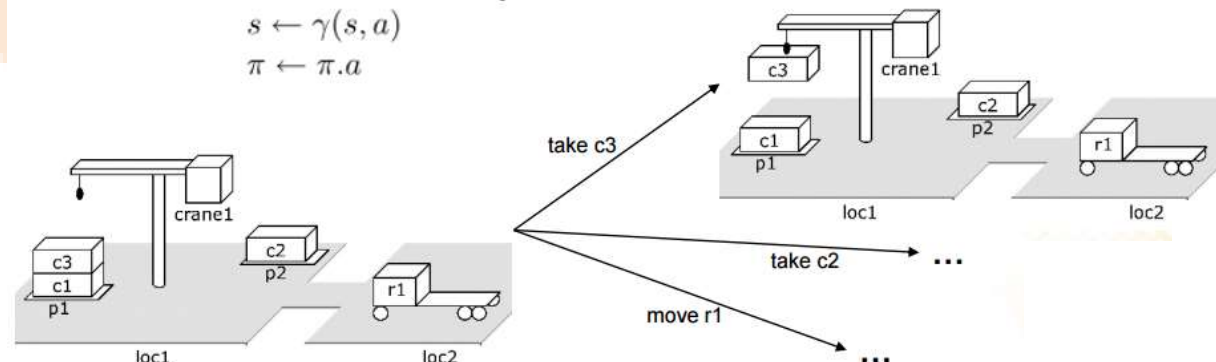
$E \leftarrow \{a \mid a \text{ is a ground instance an operator in } O,$
and $\text{precond}(a)$ is true in $s\}$

if $E = \emptyset$ then return failure

nondeterministically choose an action $a \in E$

$s \leftarrow \gamma(s, a)$

$\pi \leftarrow \pi.a$





Algoritmos de Planejamento

Busca em espaço de estados para a frente (progressão)

Os problemas de planejamento muitas vezes têm espaço de estados grandes

Forward-search(O, s_0, g)

$s \leftarrow s_0$

$\pi \leftarrow$ the empty plan

loop

if s satisfies g then return π

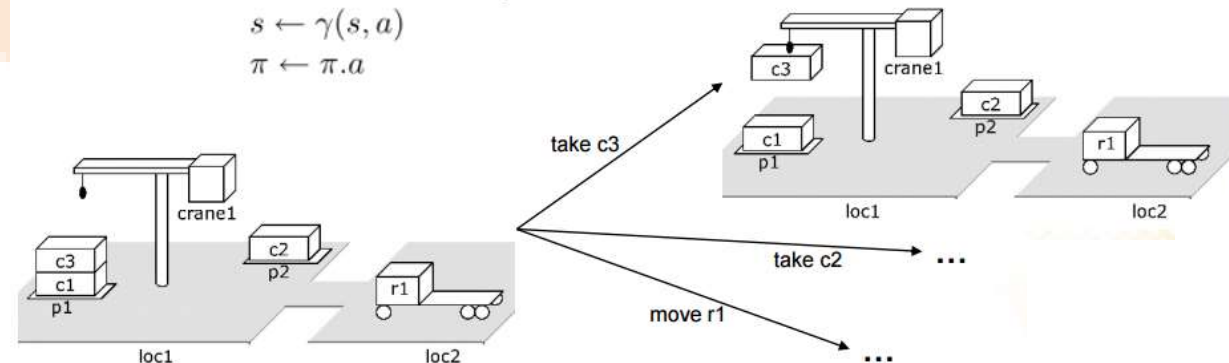
$E \leftarrow \{a \mid a \text{ is a ground instance an operator in } O, \text{ and } \text{precond}(a) \text{ is true in } s\}$

if $E = \emptyset$ then return failure

nondeterministically choose an action $a \in E$

$s \leftarrow \gamma(s, a)$

$\pi \leftarrow \pi.a$





Algoritmos de Planejamento

Busca em espaço de estados para a frente (progressão)

Algoritmos de busca clássicos:

- Busca em profundidade
- Busca em largura
- Busca de custo uniforme

Forward-search(O, s_0, g)

$s \leftarrow s_0$

$\pi \leftarrow$ the empty plan

loop

if s satisfies g then return π

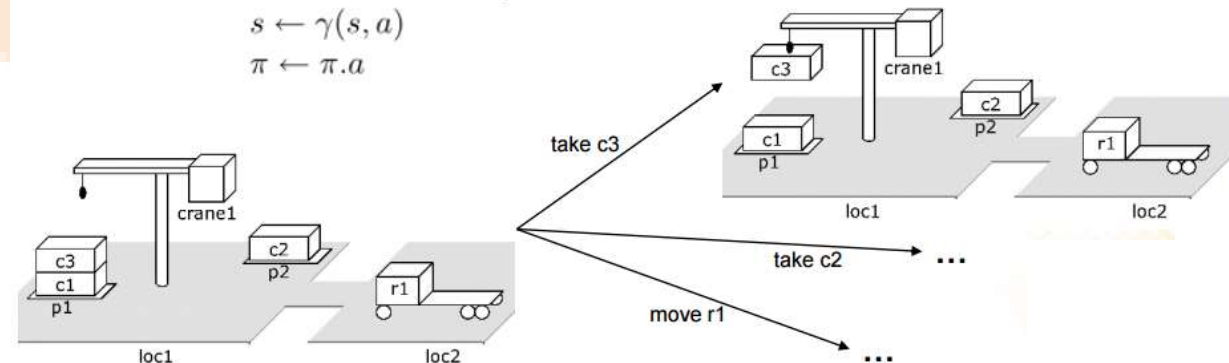
$E \leftarrow \{a \mid a \text{ is a ground instance an operator in } O,$
and $\text{precond}(a)$ is true in $s\}$

if $E = \emptyset$ then return failure

nondeterministically choose an action $a \in E$

$s \leftarrow \gamma(s, a)$

$\pi \leftarrow \pi.a$





Algoritmos de Planejamento

Busca para trás (regressão) de estados relevantes

Chama-se busca de **estados relevantes** porque consideramos apenas as ações que são relevantes ao objetivo (ou estado atual)

Backward-search(O, s_0, g)

$\pi \leftarrow$ the empty plan

loop

if s_0 satisfies g then return π

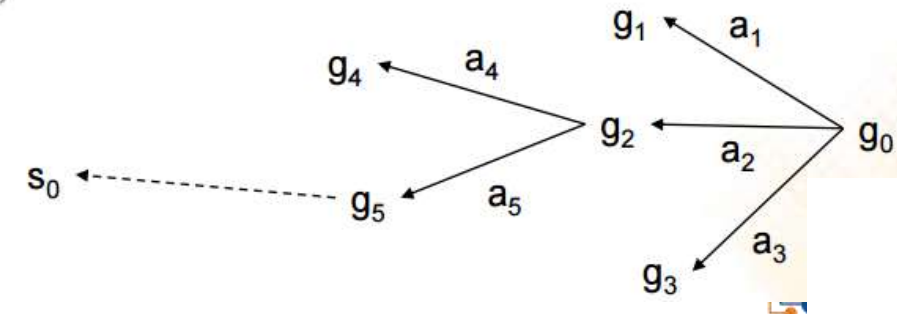
$A \leftarrow \{a \mid a \text{ is a ground instance of an operator in } O$
and $\gamma^{-1}(g, a)$ is defined $\}$

if $A = \emptyset$ then return failure

nondeterministically choose an action $a \in A$

$\pi \leftarrow a.\pi$

$g \leftarrow \gamma^{-1}(g, a)$





Algoritmos de Planejamento

Busca para trás (regressão) de estados relevantes

$$g' = (g - \text{ADD}(a)) \text{ Precond}(a)$$

Backward-search(O, s_0, g)

$\pi \leftarrow$ the empty plan

loop

if s_0 satisfies g then return π

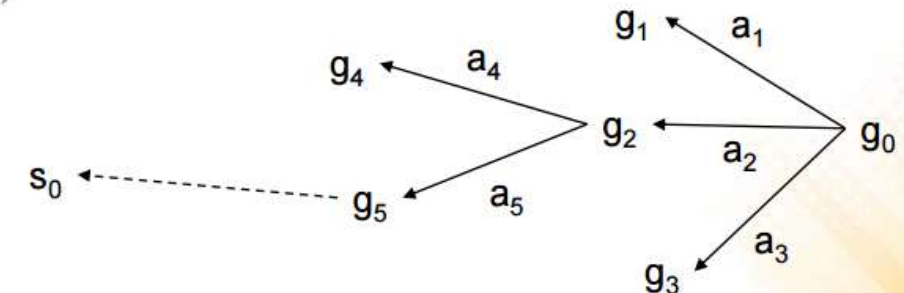
$A \leftarrow \{a \mid a \text{ is a ground instance of an operator in } O$
and $\gamma^{-1}(g, a)$ is defined $\}$

if $A = \emptyset$ then return failure

nondeterministically choose an action $a \in A$

$\pi \leftarrow a.\pi$

$g \leftarrow \gamma^{-1}(g, a)$





Algoritmos de Planejamento

Heurísticas de Planejamento

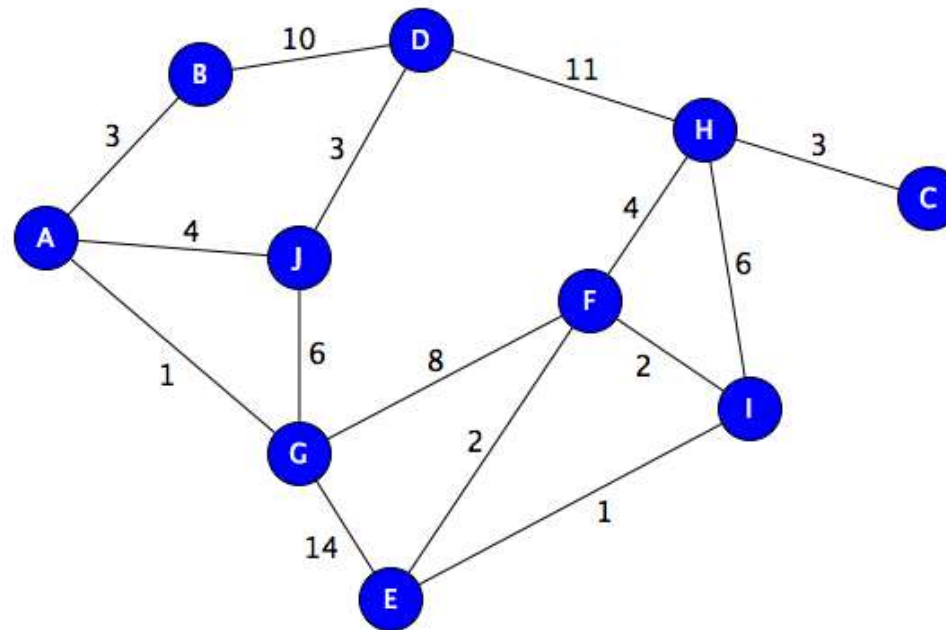
Uma função heurística $h(s)$ estima a distância de um estado s para o objetivo e onde se pode derivar uma heurística **admissível** para essa distância — uma que não superestime.

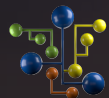




Algoritmos de Planejamento

Heurísticas de Planejamento





Data Science
Academy

Data Science Academy raphaelbsfontenelle@gmail.com 615c1fdde32fc361b30c9ec2

Obrigado



Data Science Academy



Data Science Academy