

2. Formas de raciocínio

O **raciocínio** é um processo de construção de novas sentenças a partir de sentenças existentes (fatos). Para tanto, existem categorias de raciocínio, descritas na sequência. Na Figura 4 são apresentadas os principais sistemas de raciocínio declarativo/dedutivo.

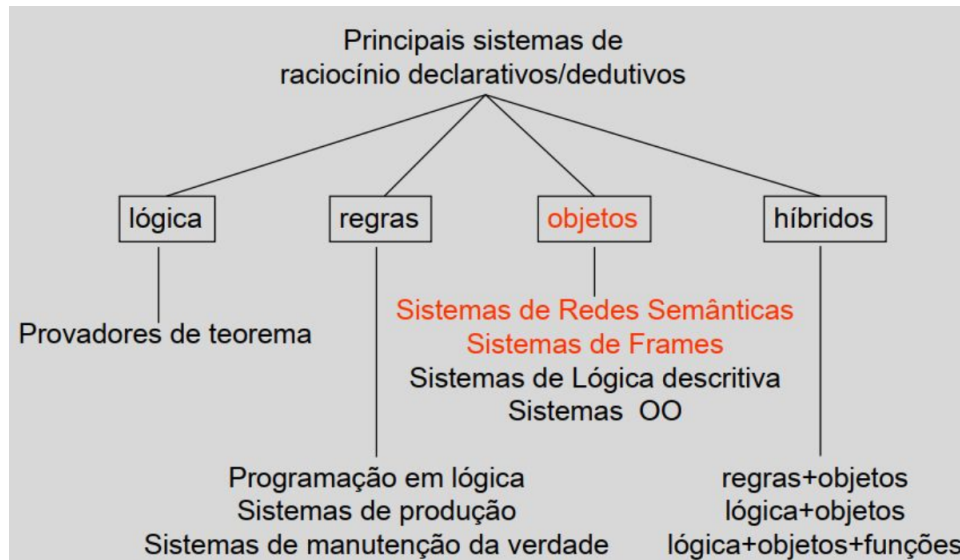


Figura 4: Exemplos dos principais sistemas de raciocínio declarativo/dedutivo

2.1. Dedução

É o processo de raciocínio no qual uma conclusão segue necessariamente das premissas supostas. Baseia-se na criação de novas sentenças a partir de premissas dadas como verdadeiras. A sentença criada é necessariamente verdadeira.

Uma das regras básicas da inferência da **Lógica Dedutiva**: regra do modus ponens (do latim: modo que afirma)

Se, **X** é verdade, e se, **X** sendo verdade implica que **Y** é verdade, então **Y** é verdade

Assim temos que:

- Fatos + regras de inferência => novos fatos
- causa → efeito

Exemplo 1:

Se há fogo (causa), há fumaça (efeito). $X \rightarrow Y$ (X implica em Y)

- Como conclusão, temos: aqui tem fogo, logo, aqui tem fumaça (novo fato)

Uma maneira usual de entender a tabela-verdade de uma condicional é pensar em uma obrigação ou um contrato. Por exemplo, uma promessa que muitos políticos fazem quando são candidatos é:

Se eu for eleito, então vou diminuir os impostos.

Se o político for eleito, os eleitores devem esperar que esse político diminua os impostos. No entanto, se o político não for eleito, os eleitores não terão nenhuma expectativa sobre o que tal político fará com os impostos, mesmo que a pessoa tenha influência suficiente para baixá-los (ROSEN, 2009).

Será apenas quando o político for eleito, mas não baixar os impostos, que os eleitores poderão dizer que o político quebrou sua promessa de campanha. Esse último cenário corresponde ao caso em que p é verdadeira e q é falsa em $p \rightarrow q$ (ROSEN, 2009)

Exemplo 2:

Premissa 1: Todo homem é mortal

Premissa 2: João é homem

Conclusão: João é mortal

2.2. Indução

Nessa técnica, uma conclusão sobre todos os membros de uma classe por meio do exame de apenas uns poucos membros da classe. De maneira geral, raciocínio do particular para o geral. Formalmente temos:

Para um conjunto de objetos, $X = \{a, b, c, d, \dots\}$, se a propriedade P é verdade para a , e se P é verdade para b , e se P é verdade para c , ... então P é verdade para todo X

Assim, parte dos fatos para gerar regras. Temos que:

- **fato1 + fato2 + fato 3 \rightarrow regra!**

Exemplo 1:

Se Sr. Antônio, assim como D. Maria, tem dor de cabeça e dengue, então todo mundo que tem dengue, tem dor de cabeça

Se **A** é verdade, e **B** é verdade, então **X** todo é verdade.

No raciocínio indutivo a conclusão contém alguma informação que não está contida nas premissas, ficando em aberto a possibilidade de que essa informação a mais cause a falsidade da conclusão apesar das premissas verdadeiras.

Uma conclusão sobre todos os membros de uma classe por meio do exame de apenas uns poucos membros da classe.

Exemplo 2:

Caso 1:

- Rose é médica
- Rose tem um ótimo salário

Caso 2:

- Márcio é médico
- Márcio tem um ótimo salário

Caso 3:

- João é médico
- João tem um ótimo salário

Lei geral: Médico tem um ótimo salário

2.3. Abdução

Essa técnica consiste em, dada uma premissa do tipo $P \rightarrow Q$, e sabendo-se que Q é verdadeira, admite-se que, talvez, P seja verdade, ou seja, supõe-se, sem certeza, que P é verdade. Assim, é uma heurística para fazer “inferências plausíveis”.

Desta forma, propicia uma conclusão plausível consistente com a informação disponível, a qual pode de fato estar errada. Formalmente temos:

Ou seja, é o oposto da dedução: do efeito para a causa

Exemplo 1:

- Se há fumaça, há fogo.
- Eu vi fumaça (efeito), logo aqui tem fogo (causa)

Exemplo 2:

- Se há febre e dor, a doença é dengue
- Este tipo de inferência preserva a falsidade

Exemplo 3:

- Se eu leio que fumar causa câncer de pulmão
- José morreu de câncer de pulmão

- **Lei Geral:** posso inferir que José era um fumante

2.4. Analogia

Essa técnica baseia-se na experiência de casos anteriores, dos quais há verdades conhecidas. Se o caso que está sendo analisado assemelha-se ao(s) caso(s) anterior(es), então supõe-se, sem certeza absoluta, que as mesmas verdades são verdadeiras também para esse caso.

Parte do particular para o particular, não possui, do ponto de vista formal, uma força de prova, mas somente é verossímil ou provável/possível.

- **Fatos + similaridades + regras de adaptação +...**

Desta forma, a partir de fatos (conhecimento em extensão) e da similaridade entre eles, resolve o problema sem gerar regras.

Exemplo 1

- Naquele caso de dengue, eu passei aspirina e não deu certo, logo, vou evitar receitar aspirina nesse caso semelhante

Exemplo 2:

- Caso anterior:
 - João ingeriu bebida alcoólica em demasia.
 - João teve amnésia.
- Caso analisado:
 - Maria ingeriu bebida alcoólica em demasia.
- Inferência por analogia:
 - Maria teve amnésia.

2.5. Exercícios

1) Determine o tipo de raciocínio utilizado a seguir:

Todos os alunos gostam de inteligência artificial.

Francisco é aluno.

Portanto, Francisco gosta de inteligência artificial.

2) Determine o tipo de raciocínio utilizado a seguir:

O ferro conduz eletricidade

O ferro é metal

O ouro conduz eletricidade

O ouro é metal

O cobre conduz eletricidade
O cobre é metal
Logo os metais conduzem eletricidade.

3) Determine o tipo de raciocínio utilizado a seguir:

Quando chove, a grama fica molhada.
A grama está molhada, então pode ter chovido

4) Determine o tipo de raciocínio utilizado a seguir:

A Terra e Marte são planetas, giram em torno do sol e têm atmosfera.
A Terra é habitada.
Marte também deve ser habitado

2.6. Representação em lógica

A representação de conhecimentos em IA clássica é um campo multidisciplinar que utiliza teorias e técnicas de três outros campos (SOWA, 2000):

- Lógica, que provê a estrutura formal e regras de inferência (REGRAS DE PRODUÇÃO).
- Ontologia, que define os tipos e as coisas existentes no domínio da aplicação (FRAMES e ONTOLOGIAS).
- Computação, que dá suporte a aplicações que a diferenciam da filosofia pura.

A Programação em Lógica é método de representação de conhecimento fundamentado na lógica simbólica. A linguagem mais representativa da Programação em Lógica é o PROLOG (PROgramming in LOGic), criada em 1972 na Universidade de Marselha.

É uma linguagem declarativa, em que ao invés do programa estipular a maneira de chegar à solução, passo a passo, limita-se a fornecer uma descrição do problema que se pretende computar.

O PROLOG baseia-se em:

- Fatos; e
- Regras de produção (relações lógicas).

Outro fato que difere o PROLOG de outras linguagens é não possuir estruturas de controle (if-else, do-while, switch). Para isso é utilizado métodos lógicos para declarar como o programa deverá encontrar a solução.

Um programa em PROLOG pode rodar em um mundo interativo: o usuário poderá formular queries utilizando fatos e regras para produzir a solução.

Como ferramenta, vamos utilizar o Interpretador online: <https://swish.swi-prolog.org> para resolver e praticar as atividades contendo representação lógica.

Para este tipo de representação, existem algumas definições descritas na sequência.

2.6.1. Fatos

Determina uma relação existente entre objetos desconhecidos. Expressa a verdade sobre um relacionamento. Com os fatos podemos montar uma base de conhecimento.

- os **nomes** de constantes e predicados (fatos) iniciam sempre com letra **minúscula**. Ex: ciclano, fulano, gosta(), filho() ...
- o **predicado** (relação unária, n-ária ou função) é escrito **primeiro** e os **objetos** relacionados (constantes ou variáveis) são escritos **depois** dentro dos parênteses.
- já as **variáveis** sempre começam por letra maiúscula. Além disso, toda sentença termina com ponto “.”.

Exemplo 1

```
gosta(maria, jose).  
homem(jose).
```

Onde:

- **homem/gosta** é o predicado ou relação
- **jose, maria** - argumento do predicado ou objeto

Exemplo 2

- O fato que **Abraão** é um progenitor de **Isaque** pode ser escrito em Prolog como:

```
progenitor(abraão,isaque).
```

- Neste caso definiu-se progenitor como o nome de uma relação: **abraão** e **isaque** são seus argumentos.

Vamos ver um exemplo de uma árvore familiar (ao lado) completa em Prolog:

```
progenitor(sara,isaque).  
progenitor(abraão,isaque).  
progenitor(abraão,ismael).  
progenitor(isaque,esau).  
progenitor(isaque,jacó).  
progenitor(jacó,josé).
```

- Cada cláusula (fato) declara um fato sobre a relação progenitor. Veja como fica a representação destes fatos:

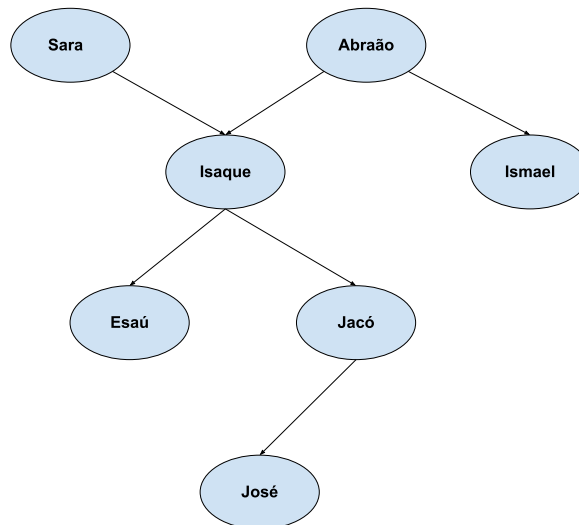


Figura 4: Exemplo de representação dos fatos da árvore familiar. Fonte: A autora

Para um programa ser interpretado, precisa-se realizar uma consulta na base de dados, procurando por células que se unifiquem à consulta.

Assim, quando o programa é interpretado, pode-se questionar o Prolog sobre a relação progenitor, por exemplo:

Isaque é o pai de Jacó?

?- progenitor(isaque,jacó).

- Como o Prolog encontra essa pergunta como um fato inserido em sua base, ele responde:

R: true

Ele responderá **true** se localizar um fato que se unifique à consulta; se o fato consultado não existir na base de dados, responderá **false** (fail).

Veja o exemplo executado na ferramenta SWISH, apresentado na Figura 5.

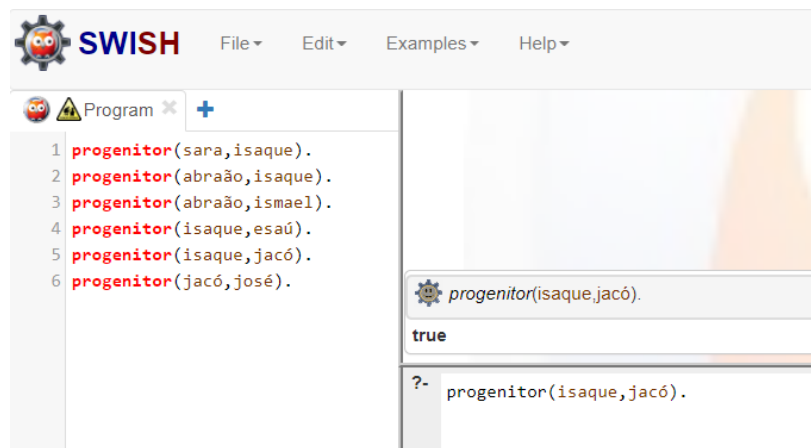


Figura 5: Exemplo de árvore familiar executada no SWISH. Fonte: A autora

Exemplo 3:

Fatos:

- **homem**(joao). -- Esta é uma relação unária
- **homem**(antonio).
- **homem**(luis).
- **mulher**(ana).
- **mulher**(maria).
- **mulher**(rita).
- **filho**(ana, joao). --Esta é uma relação n-ária
- **filho**(ana, maria).

2.6.2. Regras

As regras expressam um relacionamento entre fatos. Um relacionamento em uma regra é verdadeiro se os outros relacionamentos nesta regra também o são.

As regras são utilizadas para expressar dependência entre um fato e outro fato:

luz(acesa) :- **interruptor**(ligado).

- O “:-” significa “se”; ou seja, essa regra significa que **luz**(acesa) é verdadeiro se o **interruptor**(ligado) é verdadeiro.

Pode-se criar também grupo de fatos:

avó(X,Z) :- (**mãe**(X,Y), **mãe**(Y,Z)) ; (**mãe**(X,Y), **pai**(Y,Z)).

- Essa regra significa: X é avó de Z **SE** X é mãe de Y **E** Y é mãe de Z **OU** X é mãe de Y **E** Y é pai de Z.
- X, Y e Z são variáveis cujo valor é desconhecido a princípio e, após descoberto, não sofre mais mudanças. São sempre escritas em maiúsculas.
- **Outros conceitos:**

- **Conjunção:** O operador “E” lógico é representado por vírgula “,” entre os fatos

mãe(X,Y), **mãe**(Y,Z).

- **Disjunção:** o operador “OU” lógico é utilizado quando se declara mais de uma regra para determinar uma mesma relação. É representado por ponto-e-vírgula entre as regras “;”.

mãe(X,Y), **mãe**(Y,Z) ; (**mãe**(X,Y), **pai**(Y,Z)).

- **Negação:** é avaliada quando falsa no caso de não estar presente nenhuma regra positiva. Utiliza-se o operador “+” ou **not**().

legal(X) :- **not ilegal**(X).

criança(X) :- **not odeia**(X, sorvete).

- **Diferente:** quando duas variáveis não podem ter o mesmo valor.

vizinho(X,Y) :- **rua**(X,Y), **rua**(Y,Z), X \== Y.

Podem conter listas:

`compra(ana, [roupa, comida, brinquedo])`

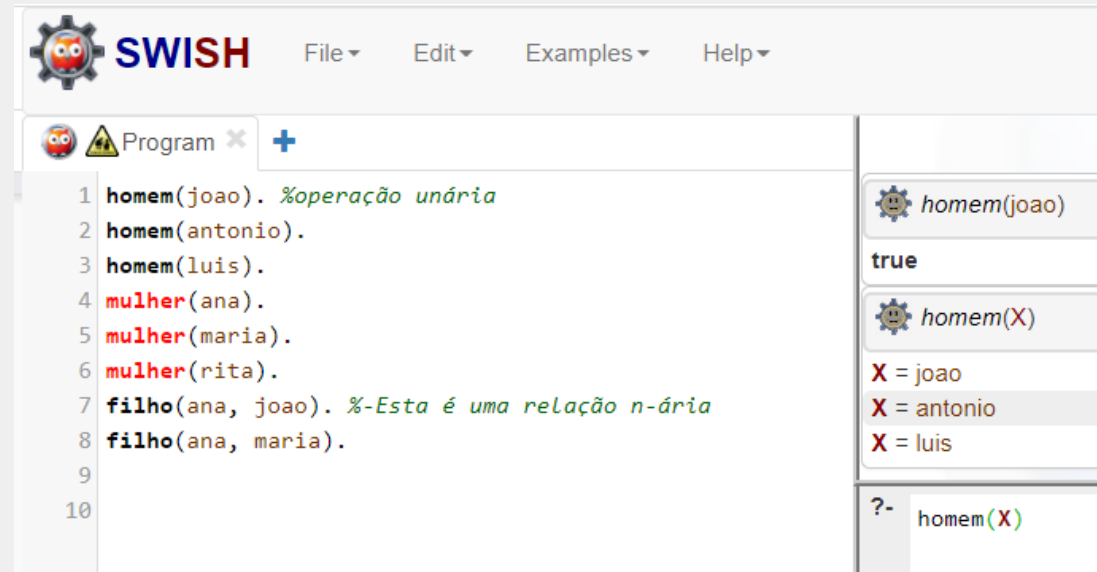
Exemplo 2:

- `X` é pai, se `X` for um homem e existir pelo menos um filho de `X`
`pai(X) :- homem(X), filho(Y, X). %ou ainda filho(_,X)`

Para testar os fatos e regras:

?- `homem(joao)` -> Qual será a resposta do interpretador?

?- `homem(X)` %%`X` é uma variável, irá retornar todos os possíveis fatos que são verdades na BC



SWISH

File Edit Examples Help

Program

```
1 homem(joao). %operação unária
2 homem(antonio).
3 homem(luis).
4 mulher(ana).
5 mulher(maria).
6 mulher(rita).
7 filho(ana, joao). %-Esta é uma relação n-ária
8 filho(ana, maria).
9
10
```

homem(joao)
true

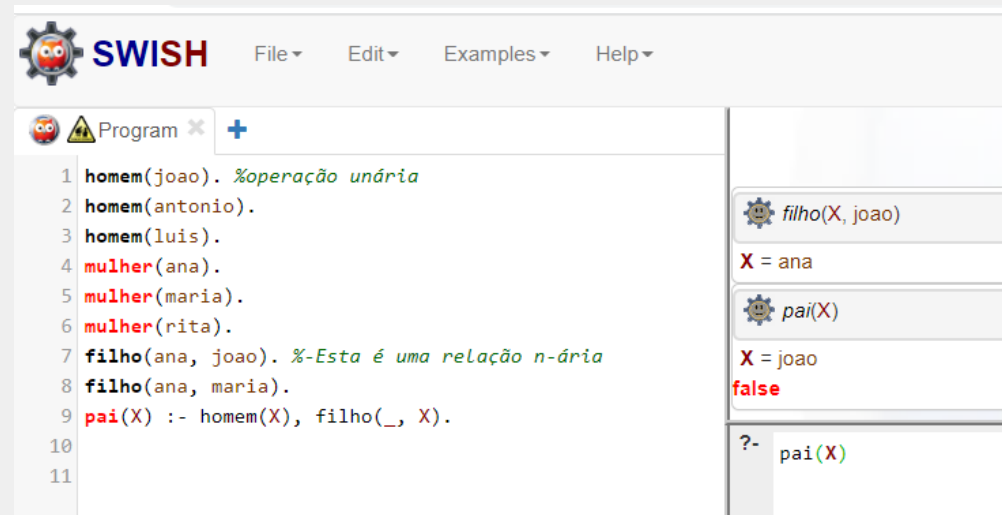
homem(X)
X = joao
X = antonio
X = luis

?- homem(X)

Quais serão os valores retornados?

?- `filho(X, joao)`

?- `pai(X)`



SWISH

File Edit Examples Help

Program

```
1 homem(joao). %operação unária
2 homem(antonio).
3 homem(luis).
4 mulher(ana).
5 mulher(maria).
6 mulher(rita).
7 filho(ana, joao). %-Esta é uma relação n-ária
8 filho(ana, maria).
9 pai(X) :- homem(X), filho(_, X).
10
11
```

filho(X, joao)
X = ana

pai(X)
X = joao
false

?- pai(X)